GPL Reference Guide for IBM SPSS Statistics

Note

Before using this information and the product it supports, read the information in “Notices” on page 387.

Product Information

This edition applies to version 28, release 0, modification 0 of IBM® SPSS® Statistics and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation .
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Introduction to GPL.....1

The Basics 1
GPL Syntax Rules..... 2
GPL Concepts..... 3
Brief Overview of GPL Algebra..... 3
How Coordinates and the GPL Algebra Interact..... 5
Common Tasks 12
How to Add Stacking to a Graph 12
How to Add Faceting (Paneling) to a Graph 14
How to Add Clustering to a Graph 15
How to Use Aesthetics 16

Chapter 2. GPL Statement and Function Reference..... 19

GPL Statements..... 19
COMMENT Statement 19
PAGE Statement 20
GRAPH Statement 20
SOURCE Statement 21
DATA Statement 21
TRANS Statement 22
COORD Statement 22
SCALE Statement 27
GUIDE Statement 38
ELEMENT Statement 43
GPL Functions..... 53
aestheticMaximum Function 59
aestheticMinimum Function 60
aestheticMissing Function 61
alpha Function 62
base Function 62
base.aesthetic Function 63
base.all Function 63
base.coordinate Function 64
begin Function (For GPL Graphs)..... 65
begin Function (For GPL Pages)..... 65
beta Function 66
bin.dot Function 66
bin.hex Function 68
bin.quantile.letter Function 70
bin.rect Function 72
binCount Function 75
binStart Function 75
binWidth Function 76
chiSquare Function 77
closed Function 77
cluster Function 77
col Function 79
collapse Function 80
color Function (For GPL Graphic Elements)..... 81
color Function (For GPL Guides)..... 82

color.brightness Function (For GPL Graphic Elements)..... 82
color.brightness Function (For GPL Guides)..... 83
color.hue Function (For GPL Graphic Elements)..... 84
color.hue Function (For GPL Guides)..... 84
color.saturation Function (For GPL Graphic Elements)..... 85
color.saturation Function (For GPL Guides)..... 86
csvSource Function 86
dataMaximum Function 87
dataMinimum Function 87
delta Function 88
density.beta Function 88
density.chiSquare Function 90
density.exponential Function 92
density.f Function 94
density.gamma Function 96
density.kernel Function 98
density.logistic Function 101
density.normal Function 103
density.poisson Function 105
density.studentizedRange Function 107
density.t Function 109
density.uniform Function 111
density.weibull Function 113
dim Function 115
end Function 117
eval Function 118
exclude Function 122
exponent Function 123
exponential Function 123
f Function 123
format Function 124
format.date Function 124
format.dateTime Function 124
format.time Function 125
from Function 125
gamma Function 126
gap Function 126
gridlines Function 126
in Function 127
include Function 127
index Function 128
iter Function 128
jump Function 128
label Function (For GPL Graphic Elements)..... 129
label Function (For GPL Guides)..... 130
layout.circle Function 131
layout.dag Function 133
layout.data Function 135
layout.grid Function 137
layout.network Function 139
layout.random Function 141
layout.tree Function 143
link.alpha Function 145
link.complete Function 147
link.delaunay Function 149
link.distance Function 151
link.gabriel Function 153
link.hull Function 155

link.influence Function 157
link.join Function 159
link.mst Function 162
link.neighbor Function 164
link.relativeNeighborhood Function 166
link.sequence Function 168
link.tsp Function 170
logistic Function 172
map Function 172
marron Function 173
max Function 173
min Function 174
mirror Function 174
missing.gap Function 175
missing.interpolate Function 175
missing.listwise Function 175
missing.pairwise Function 176
missing.wings Function 176
multiple Function 177
noConstant Function 177
node Function 177
notIn Function 178
normal Function 178
opposite Function 178
origin Function (For GPL Graphs) 179
origin Function (For GPL Scales) 179
poisson Function 180
position Function (For GPL Graphic Elements) 180
position Function (For GPL Guides) 181
preserveStraightLines Function 182
project Function 182
proportion Function 183
reflect Function 183
region.confi.count Function 184
region.confi.mean Function 186
region.confi.percent.count Function 188
region.confi.proportion.count Function 190
region.confi.smooth Function 192
region.spread.range Function 194
region.spread.sd Function 196
region.spread.se Function 198
reverse Function 201
root Function 201
sameRatio Function 202
savSource Function 202
scale Function (For GPL Axes) 202
scale Function (For GPL Graphs) 203
scale Function (For GPL Graphic Elements and form.line) 203
scale Function (For GPL Pages) 204
scaledToData Function 204
segments Function 205
shape Function (For GPL Graphic Elements) 205
shape Function (For GPL Guides) 206
showAll Function 207
size Function (For GPL Graphic Elements) 207
size Function (For GPL Guides) 208
smooth.cubic Function 208
smooth.linear Function 211

smooth.loess Function 213
smooth.mean Function 215
smooth.median Function 217
smooth.quadratic Function 220
smooth.spline Function 222
smooth.step Function 224
sort.data Function 225
sort.natural Function 226
sort.statistic Function 226
sort.values Function 226
split Function 227
sqlSource Function 227
start Function 228
startAngle Function 228
studentizedRange Function 229
summary.count Function 229
summary.count.cumulative Function 231
summary.countTrue Function 234
summary.first Function 236
summary.kurtosis Function 238
summary.last Function 240
summary.max Function 242
summary.mean Function 244
summary.median Function 246
summary.min Function 248
summary.mode Function 250
summary.percent Function 252
summary.percent.count Function 254
summary.percent.count.cumulative Function 256
summary.percent.cumulative Function 258
summary.percent.sum Function 260
summary.percent.sum.cumulative Function 262
summary.percentile Function 264
summary.percentTrue Function 267
summary.proportion Function 269
summary.proportion.count Function 270
summary.proportion.count.cumulative Function 272
summary.proportion.cumulative Function 275
summary.proportion.sum Function 275
summary.proportion.sum.cumulative Function 277
summary.proportionTrue Function 279
summary.range Function 282
summary.sd Function 284
summary.se Function 286
summary.se.kurtosis Function 288
summary.se.skewness Function 290
summary.sum Function 292
summary.sum.cumulative Function 294
summary.variance Function 296
t Function 298
texture.pattern Function 298
ticks Function 299
to Function 300
transparency Function (For GPL Graphic Elements) 300
transparency Function (For GPL Guides) 301
transpose Function 301
uniform Function 302
unit.percent Function 302

userSource Function ... 303
values Function ... 303
visible Function ... 304
weibull Function ... 304
weight Function ... 305
wrap Function ... 305

Chapter 3. GPL Examples ... 307
Using the Examples in Your Application ... 307
Summary Bar Chart Examples ... 308
Simple Bar Chart ... 308
Simple Bar Chart of Counts ... 308
Simple Horizontal Bar Chart ... 309
Simple Bar Chart With Error Bars ... 310
Simple Bar Chart with Bar for All Categories ... 311
Stacked Bar Chart ... 312
Clustered Bar Chart ... 313
Clustered and Stacked Bar Chart ... 315
Bar Chart Using an Evaluation Function ... 316
Bar Chart with Mapped Aesthetics ... 318
Faceted (Paneled) Bar Chart ... 319
3-D Bar Chart ... 322
Error Bar Chart ... 323
Histogram Examples ... 324
Histogram ... 324
Histogram with Distribution Curve ... 325
Percentage Histogram ... 327
Frequency Polygon ... 328
Stacked Histogram ... 329
Faceted (Paneled) Histogram ... 330
Population Pyramid ... 331
Cumulative Histogram ... 332
3-D Histogram ... 333
High-Low Chart Examples ... 334
Simple Range Bar for One Variable ... 334
Simple Range Bar for Two Variables ... 335
High-Low-Close Chart ... 336
Scatter/Dot Examples ... 337
Simple 1-D Scatterplot ... 337
Simple 2-D Scatterplot ... 338
Simple 2-D Scatterplot with Fit Line ... 339
Grouped Scatterplot ... 340
Grouped Scatterplot with Convex Hull ... 341
Scatterplot Matrix (SPLOM) ... 342
Bubble Plot ... 343
Binned Scatterplot ... 344
Binned Scatterplot with Polygons ... 345
Scatterplot with Border Histograms ... 346
Scatterplot with Border Boxplots ... 347
Dot Plot ... 348
2-D Dot Plot ... 350
Jittered Categorical Scatterplot ... 352
Line Chart Examples ... 353
Simple Line Chart ... 353
Simple Line Chart with Points ... 354
Line Chart of Date Data ... 355
Line Chart With Step Interpolation ... 356

Fit Line 357
Line Chart from Equation 358
Line Chart with Separate Scales 360
Pie Chart Examples 361
Pie Chart 361
Paneled Pie Chart 363
Stacked Pie Chart 364
Boxplot Examples 365
1-D Boxplot 365
Boxplot 366
Clustered Boxplot 368
Boxplot With Overlaid Dot Plot 369
Multi-Graph Examples 370
Scatterplot with Border Histograms 370
Scatterplot with Border Boxplots 371
Stocks Line Chart with Volume Bar Chart 372
Dual Axis Graph 373
Histogram with Dot Plot 374
Other Examples 375
Collapsing Small Categories 375
Mapping Aesthetics 376
Faceting by Separate Variables 377
Grouping by Separate Variables 378
Clustering Separate Variables 379
Binning over Categorical Values 380
Categorical Heat Map 381
Creating Categories Using the eval Function 383

Chapter 4. GPL Constants 385
Color Constants 385
Shape Constants 385
Size Constants 385
Pattern Constants 385

Notices 387
Trademarks 388

Index 389

Chapter 1. Introduction to GPL

The Graphics Production Language (GPL) is a language for creating graphs. It is a concise and flexible language based on the grammar described in *The Grammar of Graphics*. Rather than requiring you to learn commands that are specific to different graph types, GPL provides a basic grammar with which you can build any graph. For more information about the theory that supports GPL, see *The Grammar of Graphics, 2nd Edition* [1].

The Basics

The GPL example below creates a simple bar chart. A summary of the GPL follows the bar chart.

Note: To run the examples that appear in the GPL documentation, they must be incorporated into the syntax specific to your application. For more information, see "Using the Examples in Your Application " on page 307.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary)))
```

Figure 1. GPL for a simple bar chart

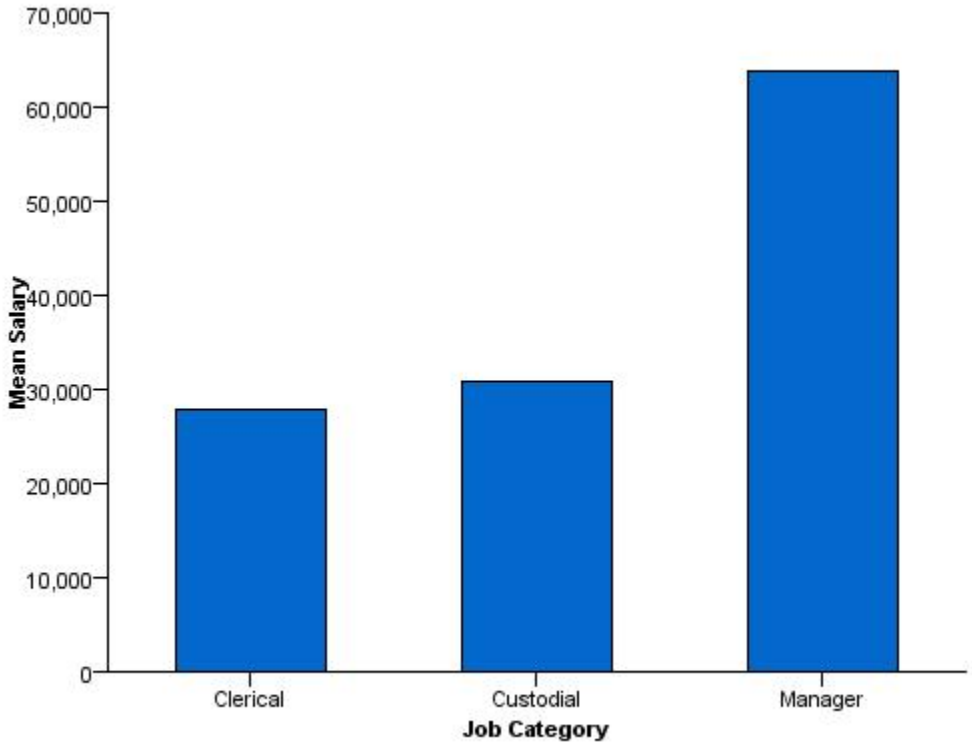


Figure 2. Simple bar chart

Each line in the example is a **statement**. One or more statements make up a block of GPL. Each statement specifies an aspect of the graph, such as the source data, relevant data transformations, coordinate

[1] Wilkinson, L. 2005. *The Grammar of Graphics*, 2nd ed. New York: Springer-Verlag.

systems, guides (for example, axis labels), graphic elements (for example, points and lines), and statistics.

Statements begin with a **label** that identifies the statement type. The label and the colon (:) that follows the label are the only items that delineate the statement.

Consider the statements in the example:

- **SOURCE**. This statement specifies the file or dataset that contains the data for the graph. In the example, it identifies `userSource`, which is a data source defined by the application that is calling the GPL. The data source could also have been a comma-separated values (CSV) file.
- **DATA**. This statement assigns a variable to a column or field in the data source. In the example, the DATA statements assign *jobcat* and *salary* to two columns in the data source. The statement identifies the appropriate columns in the data source by using the `name` function. The strings passed to the `name` function correspond to variable names in the `userSource`. These could also be the column header strings that appear in the first line of a CSV file. Note that *jobcat* is defined as a categorical variable. If a measurement level is not specified, it is assumed to be continuous.
- **SCALE**. This statement specifies the type of scale used for the graph dimensions and the range for the scale, among other options. In the example, it specifies a linear scale on the second dimension (the *y* axis in this case) and indicates that the scale must include 0. Linear scales do not necessarily include 0, but many bar charts do. Therefore, it's explicitly defined to ensure the bars start at 0. You need to include a SCALE statement only when you want to modify the scale. In this example, no SCALE statement is specified for the first dimension. We are using the default scale, which is categorical because the underlying data are categorical.
- **GUIDE**. This statement handles all of the aspects of the graph that aren't directly tied to the data but help to interpret the data, such as axis labels and reference lines. In the example, the GUIDE statements specify labels for the *x* and *y* axes. A specific axis is identified by a `dim` function. The first two dimensions of any graph are the *x* and *y* axes. The GUIDE statement is not required. Like the SCALE statement, it is needed only when you want to modify a particular guide. In this case, we are adding labels to the guides. The axis guides would still be created if the GUIDE statements were omitted, but the axes would not have labels.
- **ELEMENT**. This statement identifies the graphic element type, variables, and statistics. The example specifies *interval*. An interval element is commonly known as a bar element. It creates the bars in the example. `position()` specifies the location of the bars. One bar appears at each category in the *jobcat*. Because statistics are calculated on the second dimension in a 2-D graph, the height of the bars is the mean of *salary* for each job category. The contents of `position()` use GPL algebra. See the topic "Brief Overview of GPL Algebra" on page 3 for more information.

Details about all of the statements and functions appear in Chapter 2, "GPL Statement and Function Reference," on page 19.

GPL Syntax Rules

When writing GPL, it is important to keep the following rules in mind.

- Except in quoted strings, whitespace is irrelevant, including line breaks. Although it is possible to write a complete GPL block on one line, line breaks are used for readability.
- All quoted strings must be enclosed in quotation marks/double-quotes (for example, `"text"`). You cannot use single quotes to enclose strings.
- To add a quotation mark within a quoted string, precede the quotation mark with an escape character (\) (for example, `"Respondents Answering \"Yes\""`).
- To add a line break within a quoted string, use `\n` (for example, `"Employment\nCategory"`).
- GPL is case sensitive. Statement labels and function names must appear in the case as documented. Other names (like variable names) are also case sensitive.
- Functions are separated by commas. For example:

```
ELEMENT: point(position(x*y), color(z), size(size."5px"))
```

- GPL names must begin with an alpha character and can contain alphanumeric characters and underscores (_), including those in international character sets. GPL names are used in the SOURCE, DATA, TRANS, and SCALE statements to assign the result of a function to the name. For example, `gendervar` in the following example is a GPL name:

  ```
  DATA: gendervar=col(source(s), name("gender"), unit.category())
  ```

GPL Concepts

This section contains conceptual information about GPL. Although the information is useful for understanding GPL, it may not be easy to grasp unless you first review some examples. You can find examples in Chapter 3, "GPL Examples," on page 307.

Brief Overview of GPL Algebra

Before you can use all of the functions and statements in GPL, it is important to understand its algebra. The algebra determines how data are combined to specify the position of graphic elements in the graph. That is, the algebra defines the graph dimensions or the data frame in which the graph is drawn. For example, the frame of a basic scatterplot is specified by the values of one variable crossed with the values of another variable. Another way of thinking about the algebra is that it identifies the variables you want to analyze in the graph.

The GPL algebra can specify one or more variables. If it includes more than one variable, you must use one of the following operators:

- **Cross (*)**. The cross operator crosses all of the values of one variable with all of the values of another variable. A result exists for every case (row) in the data. The cross operator is the most commonly used operator. It is used whenever the graph includes more than one axis, with a different variable on each axis. Each variable on each axis is crossed with each variable on the other axes (for example, A*B results in A on the *x* axis and B on the *y* axis when the coordinate system is 2-D). Crossing can also be used for paneling (faceting) when there are more crossed variables than there are dimensions in a coordinate system. That is, if the coordinate system were 2-D rectangular and three variables were crossed, the last variable would be used for paneling (for example, with A*B*C, C is used for paneling when the coordinate system is 2-D).
- **Nest (/)**. The nest operator nests all of the values of one variable in all of the values of another variable. The difference between crossing and nesting is that a result exists only when there is a corresponding value in the variable that nests the other variable. For example, `city/state` nests the *city* variable in the *state* variable. A result will exist for each city and its appropriate state, not for every combination of *city* and *state*. Therefore, there will not be a result for *Chicago* and *Montana*. Nesting always results in paneling, regardless of the coordinate system.
- **Blend (+)**. The blend operator combines all of the values of one variable with all of the values of another variable. For example, you may want to combine two salary variables on one axis. Blending is often used for repeated measures, as in `salary2004+salary2005`.

Crossing and nesting add dimensions to the graph specification. Blending combines the values into one dimension. How the dimensions are interpreted and drawn depends on the coordinate system. See "How Coordinates and the GPL Algebra Interact" on page 5 for details about the interaction between the coordinate system and the algebra.

Rules

Like elementary mathematical algebra, GPL algebra has associative, distributive, and commutative rules. All operators are associative:

```
(X*Y)*Z = X*(Y*Z)
(X/Y)/Z = X/(Y/Z)
(X+Y)+Z = X+(Y+Z)
```

The cross and nest operators are also distributive:

```
X*(Y+Z) = X*Y+X*Z
X/(Y+Z) = X/Y+X/Z
```

However, GPL algebra operators are *not* commutative. That is,

```
X*Y ≠ Y*X
X/Y ≠ Y/X
```

Operator Precedence

The nest operator takes precedence over the other operators, and the cross operator takes precedence over the blend operator. Like mathematical algebra, the precedence can be changed by using parentheses. You will almost always use parentheses with the blend operator because the blend operator has the lowest precedence. For example, to blend variables before crossing or nesting the result with other variables, you would do the following:

```
(A+B)*C
```

However, note that there are some cases in which you will cross *then* blend. For example, consider the following.

```
(A*C)+(B*D)
```

In this case, the variables are crossed first because there is no way to untangle the variable values after they are blended. A needs to be crossed with C and B needs to be crossed with D. Therefore, using `(A+B)*(C+D)` won't work. `(A*C)+(B*D)` crosses the correct variables and then blends the results together.

Note: In this last example, the parentheses are superfluous, because the cross operator's higher precedence ensures that the crossing occurs before the blending. The parentheses are used for readability.

Analysis Variable

Statistics other than count-based statistics require an analysis variable. The analysis variable is the variable on which a statistic is calculated. In a 1-D graph, this is the first variable in the algebra. In a 2-D graph, this is the second variable. Finally, in a 3-D graph, it is the third variable.

In all of the following, *salary* is the analysis variable:

- **1-D.** `summary.sum(salary)`
- **2-D.** `summary.mean(jobcat*salary)`
- **3-D.** `summary.mean(jobcat*gender*salary)`

The previous rules apply only to algebra used in the `position` function. Algebra can be used elsewhere (as in the `color` and `label` functions), in which case the only variable in the algebra is the analysis variable. For example, in the following ELEMENT statement for a 2-D graph, the analysis variable is *salary* in the `position` function and the `label` function.

```
ELEMENT: interval(position(summary.mean(jobcat*salary)), label(summary.mean(salary)))
```

Unity Variable

The unity variable (indicated by 1) is a placeholder in the algebra. It is not the same as the numeric value 1. When a scale is created for the unity variable, unity is located in the middle of the scale but no other values exist on the scale. The unity variable is needed only when there is no explicit variable in a specific dimension and you need to include the dimension in the algebra.

For example, assume a 2-D rectangular coordinate system. If you are creating a graph showing the count in each *jobcat* category, `summary.count(jobcat)` appears in the GPL specification. Counts are shown

along the *y* axis, but there is no explicit variable in that dimension. If you want to panel the graph, you need to specify something in the second dimension before you can include the paneling variable. Thus, if you want to panel the graph by columns using `gender`, you need to change the specification to `summary.count(jobcat*1*gender)`. If you want to panel by rows instead, there would be another unity variable to indicate the missing third dimension. The specification would change to `summary.count(jobcat*1*1*gender)`.

You can't use the unity variable to compute statistics that require an analysis variable (like `summary.mean`). However, you can use it with count-based statistics (like `summary.count` and `summary.percent.count`).

User Constants

The algebra can also include user constants, which are quoted string values (for example, `"2005"`). When a user constant is included in the algebra, it is like adding a new variable, with the variable's value equal to the constant for all cases. The effect of this depends on the algebra operators and the function in which the user constant appears.

In the `position` function, the constants can be used to create separate scales. For example, in the following GPL, two separate scales are created for the paneled graph. By nesting the values of each variable in a different string and blending the results, two different groups of cases with different scale ranges are created.

```
ELEMENT: line(position(date*(calls/"Calls"+orders/"Orders")))
```

For a full example, see “Line Chart with Separate Scales ” on page 360 .

If the cross operator is used instead of the nest operator, both categories will have the same scale range. The panel structures will also differ.

```
ELEMENT: line(position(date*calls*"Calls"+date*orders*"Orders"))
```

Constants can also be used in the `position` function to create a category of all cases when the constant is blended with a categorical variable. Remember that the value of the user constant is applied to all cases, so that's why the following works:

```
ELEMENT: interval(position(summary.mean((jobcat+"All")*salary)))
```

For a full example, see “Simple Bar Chart with Bar for All Categories ” on page 311 .

Aesthetic functions can also take advantage of user constants. Blending variables creates multiple graphic elements for the same case. To distinguish each group, you can mimic the blending in the aesthetic function—this time with user constants.

```
ELEMENT: point(position(jobcat*(salbegin+salary), color("Beginning"+"Current")))
```

User constants are not required to create most charts, so you can ignore them in the beginning. However, as you become more proficient with GPL, you may want to return to them to create custom graphs.

How Coordinates and the GPL Algebra Interact

The algebra defines the dimensions of the graph. Each crossing results in an additional dimension. Thus, `gender*jobcat*salary` specifies three dimensions. How these dimensions are drawn depends on the coordinate system and any functions that may modify the coordinate system.

Some examples may clarify these concepts. The relevant GPL statements are extracted from the full specification.

1-D Graph

```
COORD: rect(dim(1))
ELEMENT: point(position(salary))
```

Full Specification

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
COORD: rect(dim(1))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: point(position(salary))
```

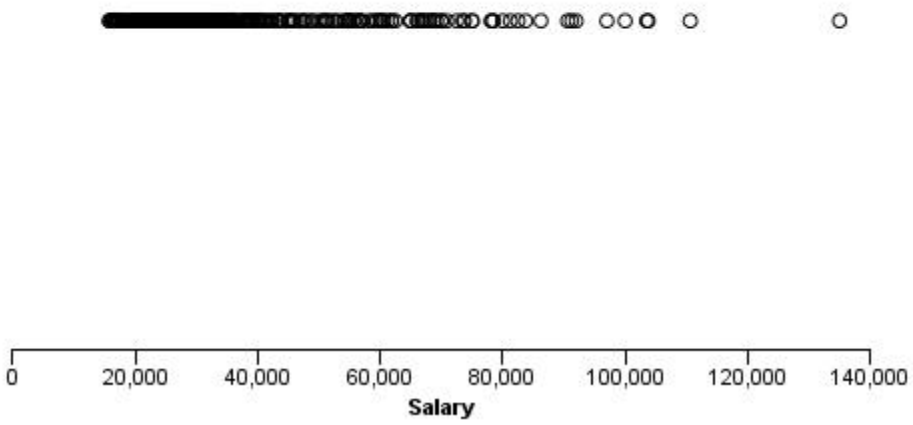


Figure 3. Simple 1-D scatterplot

- The coordinate system is explicitly set to one-dimensional, and only one variable appears in the algebra.
- The variable is plotted on one dimension.

2-D Graph

```
ELEMENT: point(position(salbegin*salary))
```

Full Specification

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: point(position(salbegin*salary))
```

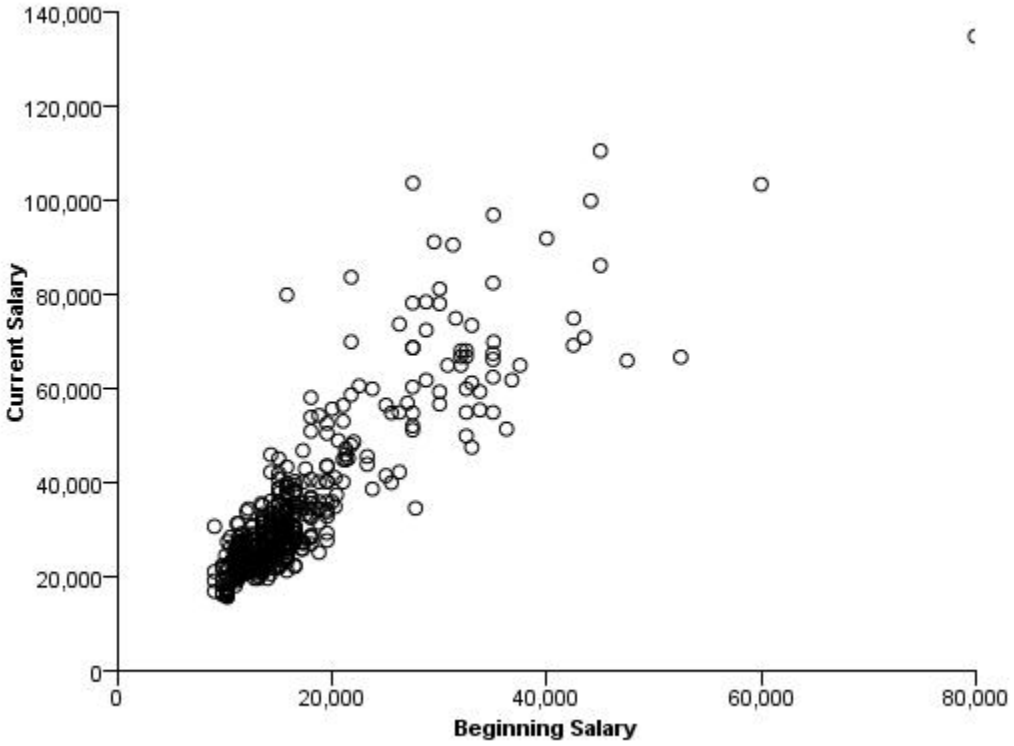


Figure 4. Simple 2-D scatterplot

- No coordinate system is specified, so it is assumed to be 2-D rectangular.
- The two crossed variables are plotted against each other.

Another 2-D Graph

```
ELEMENT: interval(position(summary.count(jobcat)))
```

Full Specification

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Count"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.count(jobcat)))
```

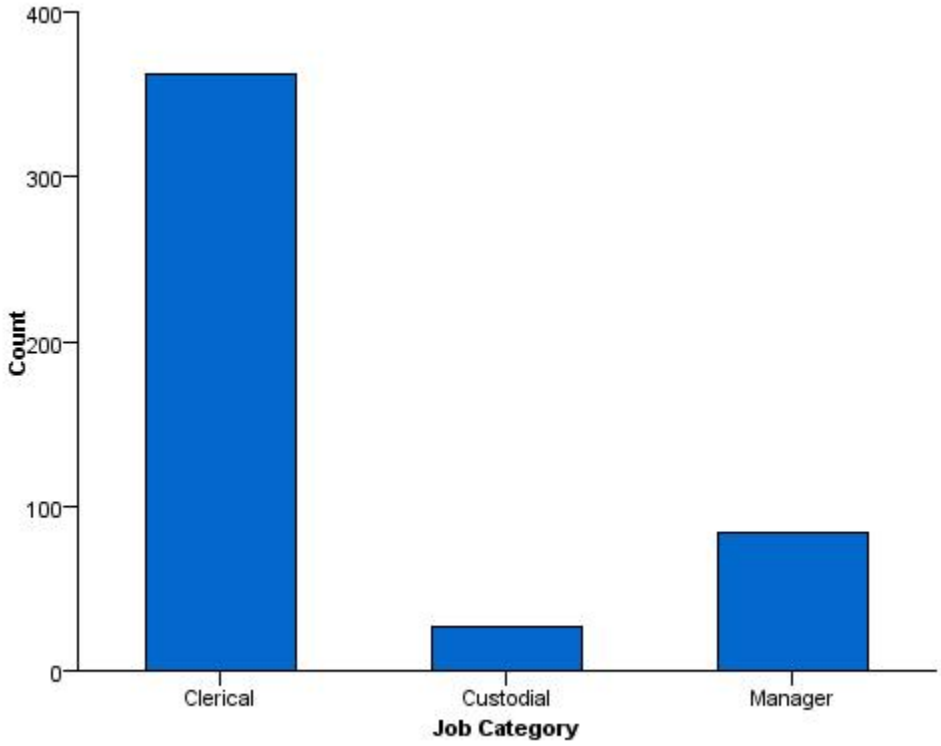


Figure 5. Simple 2-D bar chart of counts

- No coordinate system is specified, so it is assumed to be 2-D rectangular.
- Although there is only one variable in the specification, another for the result of the count statistic is implied (percent statistics behave similarly). The algebra could have been written as `jobcat*1`.
- The variable and the result of the statistic are plotted.

A Faceted (Paneled) 2-D Graph

```
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)))
```

Full Specification

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: salary = col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(3), label("Gender"))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)))
```

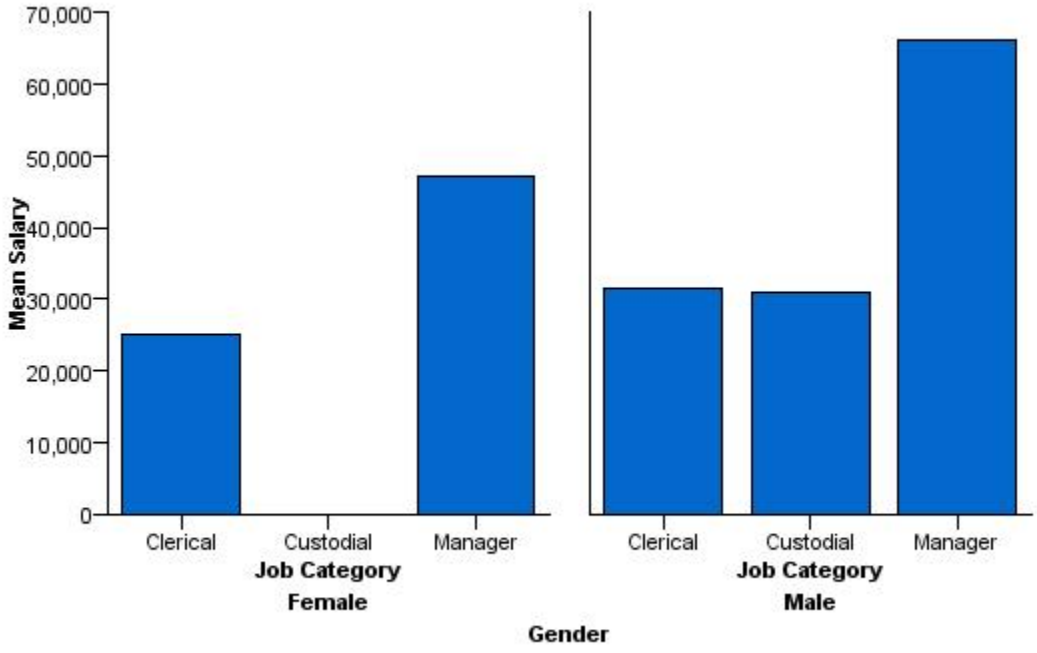


Figure 6. Faceted 2-D bar chart

- No coordinate system is specified, so it is assumed to be 2-D rectangular.
- There are three variables in the algebra, but only two dimensions. The last variable is used for faceting (also known as paneling).
- The second dimension variable in a 2-D chart is the analysis variable. That is, it is the variable on which the statistic is calculated.
- The first variable is plotted against the result of the summary statistic calculated on the second variable for each category in the faceting variable.

A Faceted (Paneled) 2-D Graph with Nested Categories

```
ELEMENT: interval(position(summary.mean(jobcat/gender*salary)))
```

Full Specification

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: salary = col(source(s), name("salary"))
SCALE: linear(dim(2), include(0.0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1.1), label("Job Category"))
GUIDE: axis(dim(1), label("Gender"))
ELEMENT: interval(position(summary.mean(jobcat/gender*salary)))
```

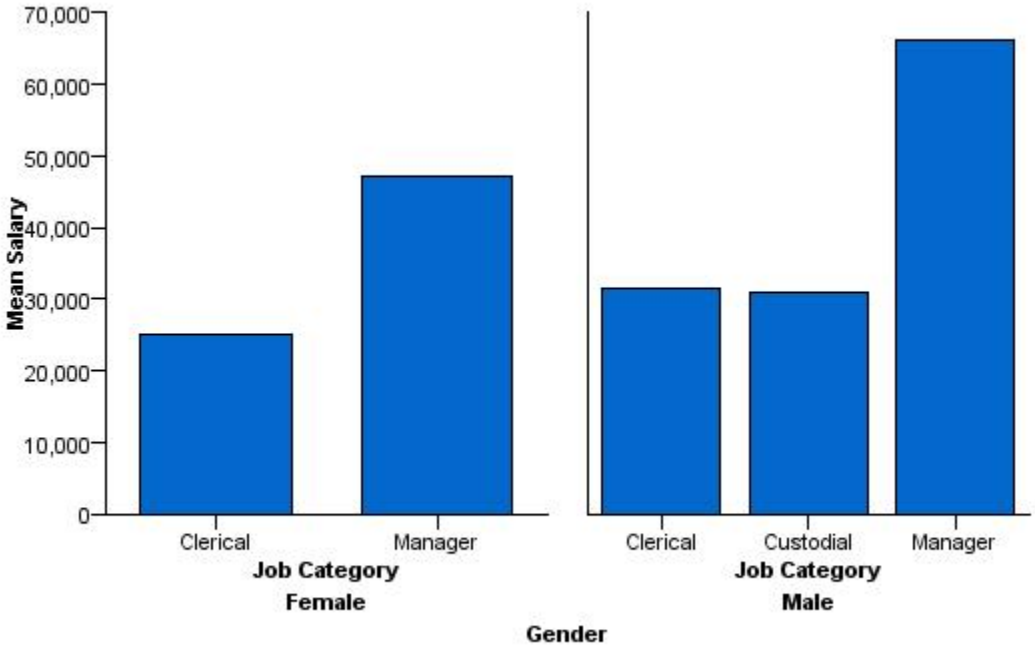


Figure 7. Faceted 2-D bar chart with nested categories

- This example is the same as the previous paneled example, except for the algebra.
- The second dimension variable is the same as in the previous example. Therefore, it is the variable on which the statistic is calculated.
- *jobcat* is nested in *gender*. Nesting always results in faceting, regardless of the available dimensions.
- With nested categories, only those combinations of categories that occur in the data are shown in the graph. In this case, there is no bar for *Female* and *Custodial* in the graph, because there is no case with this combination of categories in the data. Compare this result to the previous example that created facets by crossing categorical variables.

A 3-D Graph

```
COORD: rect(dim(1,2,3))
ELEMENT: interval(position(summary.mean(jobcat*gender*salary)))
```

Full Specification

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
DATA: salary=col(source(s), name("salary"))
COORD: rect(dim(1,2,3))
SCALE: linear(dim(3), include(0))
GUIDE: axis(dim(3), label("Mean Salary"))
GUIDE: axis(dim(2), label("Gender"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*gender*salary)))
```

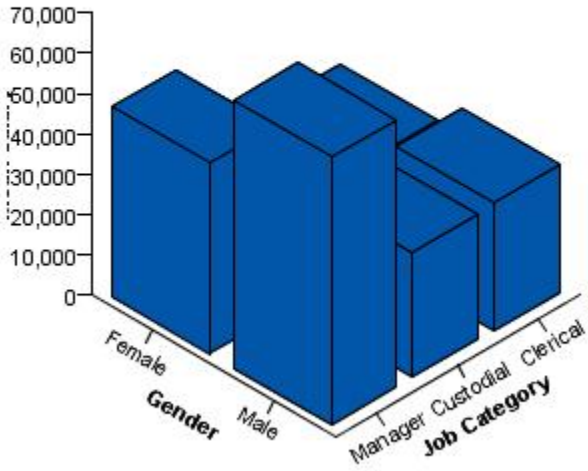


Figure 8. 3-D bar chart

- The coordinate system is explicitly set to three-dimensional, and there are three variables in the algebra.
- The three variables are plotted on the available dimensions.
- The *third* dimension variable in a 3-D chart is the analysis variable. This differs from the 2-D chart in which the second dimension variable is the analysis variable.

A Clustered Graph

```
COORD: rect(dim(1,2), cluster(3))
ELEMENT: interval(position(summary.mean(gender*salary*jobcat)), color(gender))
```

Full Specification

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
DATA: salary=col(source(s), name("salary"))
COORD: rect(dim(1,2), cluster(3))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(3), label("Gender"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)), color(jobcat))
```

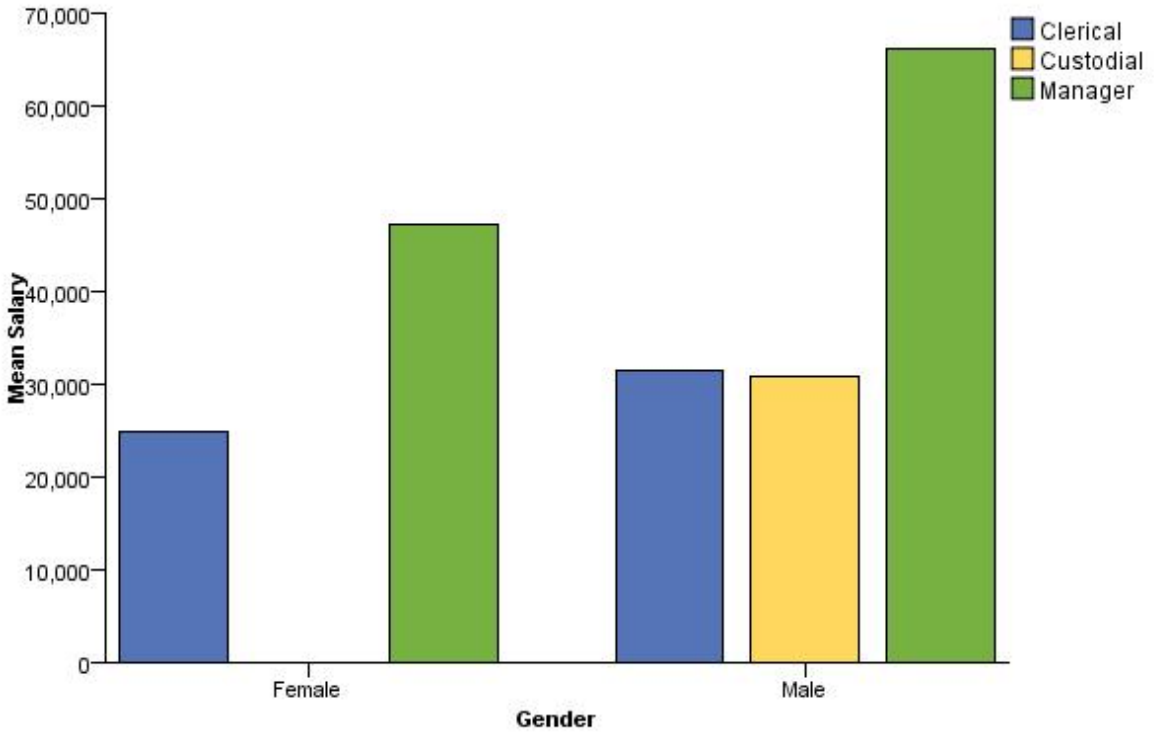


Figure 9. Clustered 2-D bar chart

- The coordinate system is explicitly set to two-dimensional, but it is modified by the `cluster` function.
- The `cluster` function indicates that clustering occurs along dim(3), which is the dimension associated with *jobcat* because it is the third variable in the algebra.
- The variable in dim(1) identifies the variable whose values determine the bars in each cluster. This is *gender*.
- Although the coordinate system was modified, this is still a 2-D chart. Therefore, the analysis variable is still the second dimension variable.
- The variables are plotted using the modified coordinate system. Note that the graph would be a paneled graph if you removed the `cluster` function. The charts would look similar and show the same results, but their coordinate systems would differ. Refer back to the paneled 2-D graph to see the difference.

Common Tasks

This section provides information for adding common graph features. This GPL creates a simple 2-D bar chart. You can apply the steps to any graph, but the examples use the GPL in “The Basics ” on page 1 as a "baseline."

How to Add Stacking to a Graph

Stacking involves a couple of changes to the ELEMENT statement. The following steps use the GPL shown in “The Basics ” on page 1 as a "baseline" for the changes.

1. Before modifying the ELEMENT statement, you need to define an additional *categorical* variable that will be used for stacking. This is specified by a DATA statement (note the `unit.category()` function):

```
DATA: gender=col(source(s), name("gender"), unit.category())
```

2. The first change to the ELEMENT statement will split the graphic element into color groups for each *gender* category. This splitting results from using the `color` function:

```
ELEMENT: interval(position(summary.mean(jobcat*salary)), color(gender))
```

3. Because there is no collision modifier for the interval element, the groups of bars are overlaid on each other, and there's no way to distinguish them. In fact, you may not even see graphic elements for one of the groups because the other graphic elements obscure them. You need to add the stacking collision modifier to re-position the groups (we also changed the statistic because stacking summed values makes more sense than stacking the mean values):

```
ELEMENT: interval.stack(position(summary.sum(jobcat*salary)), color(gender))
```

The complete GPL is shown below:

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: salary = col(source(s), name("salary"))
SCALE: linear(dim(2), include(0.0))
GUIDE: axis(dim(2), label("Sum Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval.stack(position(summary.sum(jobcat*salary)), color(gender))
```

Following is the graph created from the GPL.

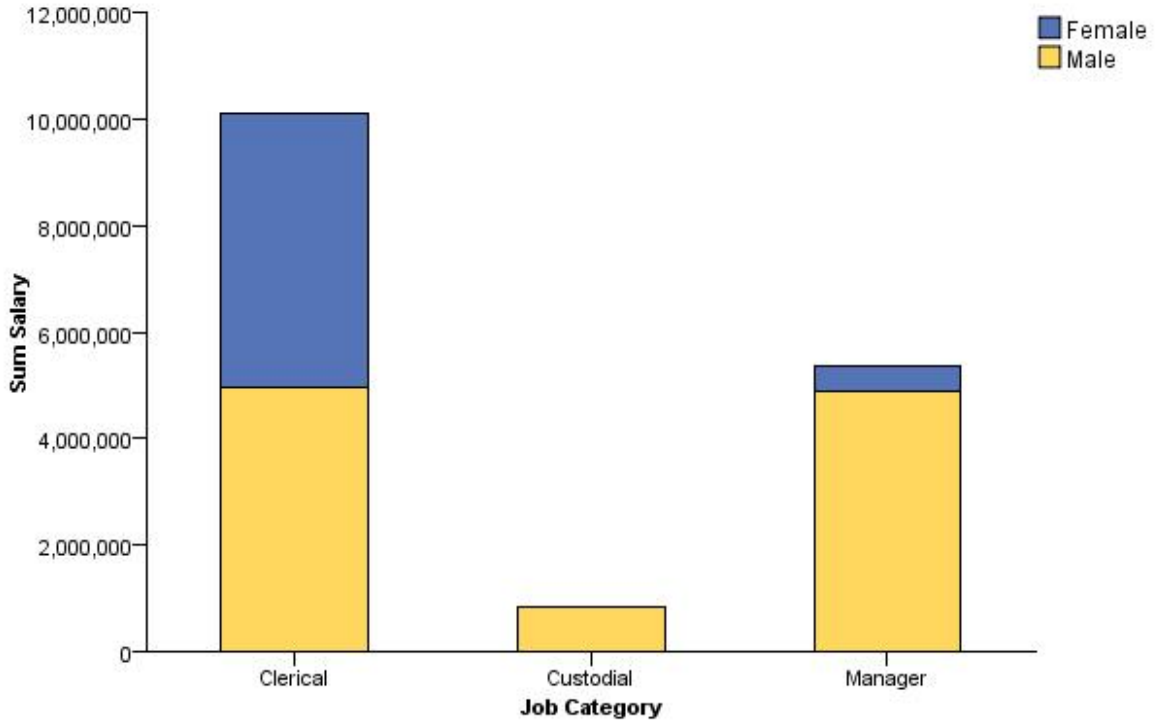


Figure 10. Stacked bar chart

Legend Label

The graph includes a legend, but it has no label by default. To add or change the label for the legend, you use a GUIDE statement:

```
GUIDE: legend(aesthetic(aesthetic.color), label("Gender"))
```

How to Add Faceting (Paneling) to a Graph

Faceted variables are added to the algebra in the ELEMENT statement. The following steps use the GPL shown in “The Basics ” on page 1 as a "baseline" for the changes.

1. Before modifying the ELEMENT statement, we need to define an additional *categorical* variable that will be used for faceting. This is specified by a DATA statement (note the `unit.category()` function):

```
DATA: gender=col(source(s), name("gender"), unit.category())
```

2. Now we add the variable to the algebra. We will cross the variable with the other variables in the algebra:

```
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)))
```

Those are the only necessary steps. The final GPL is shown below.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: salary = col(source(s), name("salary"))
SCALE: linear(dim(2), include(0.0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)))
```

Following is the graph created from the GPL.

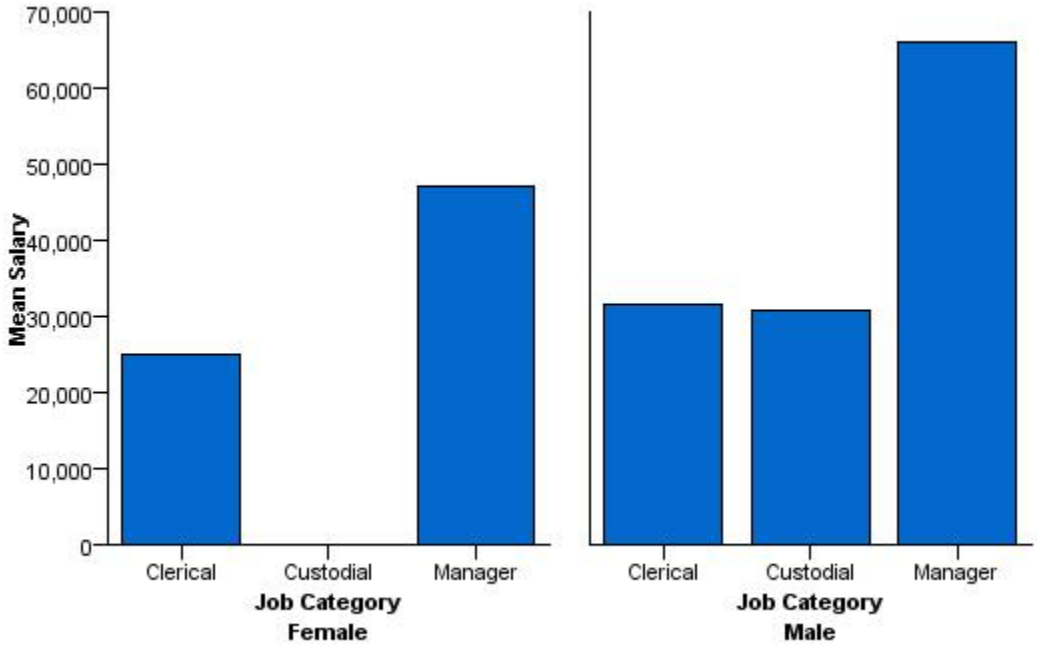


Figure 11. Faceted bar chart

Additional Features

Labeling. If you want to label the faceted dimension, you treat it like the other dimensions in the graph by adding a GUIDE statement for its axis:

```
GUIDE: axis(dim(3), label("Gender"))
```

In this case, it is specified as the 3rd dimension. You can determine the dimension number by counting the crossed variables in the algebra. *gender* is the 3rd variable.

Nesting. Faceted variables can be nested as well as crossed. Unlike crossed variables, the nested variable is positioned next to the variable in which it is nested. So, to nest *gender* in *jobcat*, you would do the following:

```
ELEMENT: interval(position(summary.mean(jobcat/gender*salary)))
```

Because *gender* is used for nesting, it is not the 3rd dimension as it was when crossing to create facets. You can't use the same simple counting method to determine the dimension number. You still count the crossings, but you count each crossing *as a single factor*. The number that you obtain by counting each crossed factor is used for the nested variable (in this case, *1*). The other dimension is indicated by the nested variable dimension followed by a dot and the number 1 (in this case, *1.1*). So, you would use the following convention to refer to the *gender* and *jobcat* dimensions in the GUIDE statement:

```
GUIDE: axis(dim(1), label("Gender"))
GUIDE: axis(dim(1.1), label("Job Category"))
GUIDE: axis(dim(2), label("Mean Salary"))
```

How to Add Clustering to a Graph

Clustering involves changes to the COORD statement and the ELEMENT statement. The following steps use the GPL shown in “The Basics ” on page 1 as a "baseline" for the changes.

1. Before modifying the COORD and ELEMENT statements, you need to define an additional *categorical* variable that will be used for clustering. This is specified by a DATA statement (note the `unit.category()` function):

   ```
   DATA: gender=col(source(s), name("gender"), unit.category())
   ```

2. Now you will modify the COORD statement. If, like the baseline graph, the GPL does not already include a COORD statement, you first need to add one:

   ```
   COORD: rect(dim(1,2))
   ```

   In this case, the default coordinate system is now explicit.

3. Next add the `cluster` function to the coordinate system and specify the clustering dimension. In a 2-D coordinate system, this is the third dimension:

   ```
   COORD: rect(dim(1,2), cluster(3))
   ```

4. Now we add the clustering dimension variable to the algebra. This variable is in the 3rd position, corresponding to the clustering dimension specified by the `cluster` function in the COORD statement:

   ```
   ELEMENT: interval(position(summary.mean(jobcat*salary*gender)))
   ```

   Note that this algebra looks similar to the algebra for faceting. Without the `cluster` function added in the previous step, the resulting graph would be faceted. The `cluster` function essentially collapses the faceting into one axis. Instead of a facet for each *gender* category, there is a cluster on the *x* axis for each category.

5. Because clustering changes the dimensions, we update the GUIDE statement so that it corresponds to the clustering dimension.

   ```
   GUIDE: axis(dim(3), label("Gender"))
   ```

6. With these changes, the chart is clustered, but there is no way to distinguish the bars in each cluster. You need to add an aesthetic to distinguish the bars:

```
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)), color(jobcat))
```

The complete GPL looks like the following.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
DATA: salary=col(source(s), name("salary"))
COORD: rect(dim(1,2), cluster(3))
SCALE: linear(dim(2), include(0))
```

```
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(3), label("Gender"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)), color(jobcat))
```

Following is the graph created from the GPL.

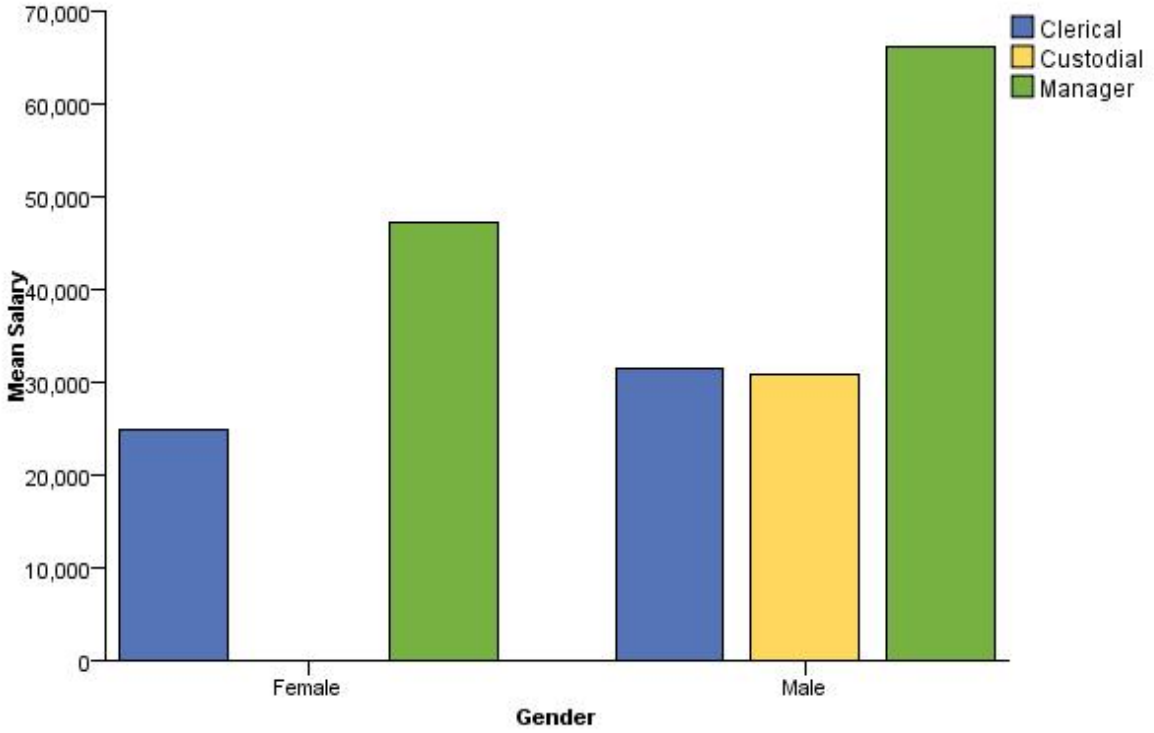


Figure 12. Clustered bar chart

Legend Label

The graph includes a legend, but it has no label by default. To change the label for the legend, you use a `GUIDE` statement:

```
GUIDE: legend(aesthetic(aesthetic.color), label("Gender"))
```

How to Use Aesthetics

GPL includes several different aesthetic functions for controlling the appearance of a graphic element. The simplest use of an aesthetic function is to define a uniform aesthetic for every instance of a graphic element. For example, you can use the `color` function to assign a color constant (like `color.red`) to the point element, thereby making *all* of the points in the graph red.

A more interesting use of an aesthetic function is to change the value of the aesthetic based on the value of another variable. For example, instead of a uniform color for the scatterplot points, the color could vary based on the value of the categorical variable *gender*. All of the points in the *Male* category will be one color, and all of the points in the *Female* category will be another. Using a categorical variable for an aesthetic creates groups of cases. In addition to identifying the graphic elements for the groups of cases, the grouping allows you to evaluate statistics for the individual groups, if needed.

An aesthetic may also vary based on a set of continuous values. Using continuous values for the aesthetic does not result in distinct groups of graphic elements. Instead, the aesthetic varies along the same continuous scale. There are no distinct groups on the scale, so the color varies gradually, just as the continuous values do.

The steps below use the following GPL as a "baseline" for adding the aesthetics. This GPL creates a simple scatterplot.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: point(position(salbegin*salary))
```

Figure 13. Baseline GPL for example

1. First, you need to define an additional *categorical* variable that will be used for one of the aesthetics. This is specified by a DATA statement (note the `unit.category()` function):

   ```
   DATA: gender=col(source(s), name("gender"), unit.category())
   ```

2. Next you need to define another variable, this one being *continuous*. It will be used for the other aesthetic.

   ```
   DATA: prevexp=col(source(s), name("prevexp"))
   ```

3. Now you will add the aesthetics to the graphic element in the ELEMENT statement. First add the aesthetic for the categorical variable:

   ```
   ELEMENT: point(position(salbegin*salary), shape(gender))
   ```

   Shape is a good aesthetic for the categorical variable. It has distinct values that correspond well to categorical values.

4. Finally add the aesthetic for the continuous variable:

   ```
   ELEMENT: point(position(salbegin*salary), shape(gender), color(prevexp))
   ```

   Not all aesthetics are available for continuous variables. That's another reason why shape was a good aesthetic for the categorical variable. Shape is not available for continuous variables because there aren't enough shapes to cover a continuous spectrum. On the other hand, color gradually changes in the graph. It can capture the full spectrum of continuous values. Transparency or brightness would also work well.

The complete GPL looks like the following.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin = col(source(s), name("salbegin"))
DATA: salary = col(source(s), name("salary"))
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: prevexp = col(source(s), name("prevexp"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: point(position(salbegin*salary), shape(gender), color(prevexp))
```

Following is the graph created from the GPL.

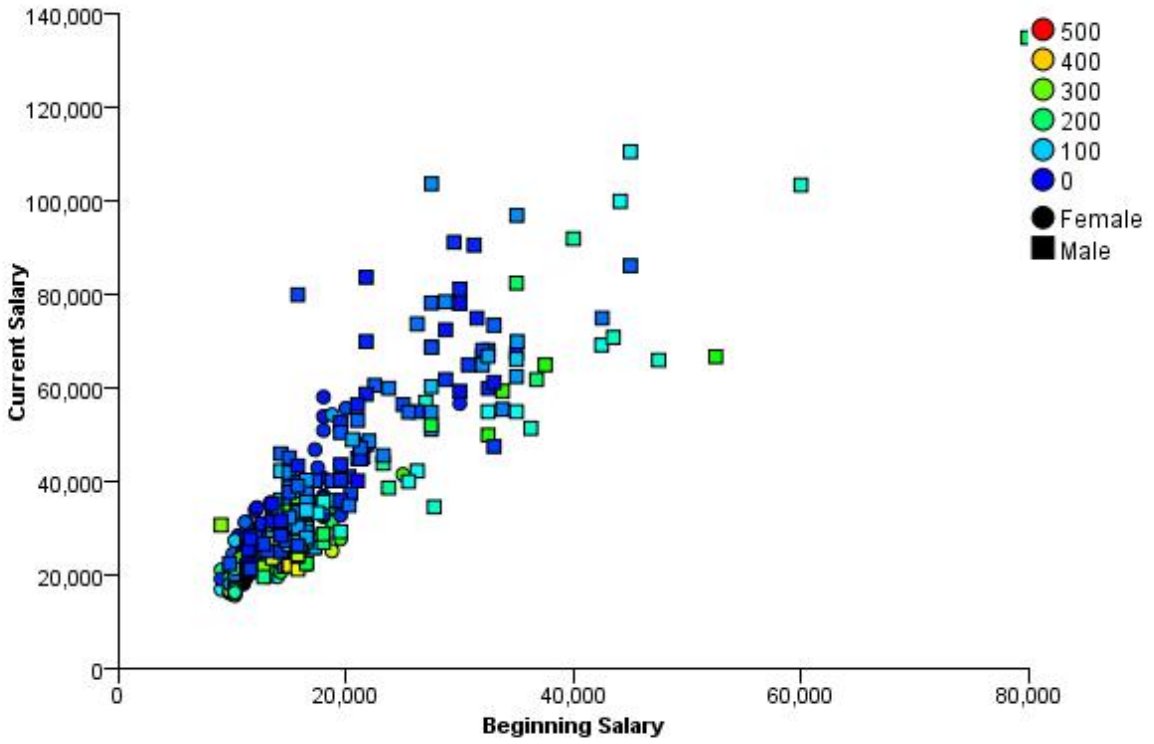


Figure 14. Scatterplot with aesthetics

Legend Label

The graph includes legends, but the legends have no labels by default. To change the labels, you use GUIDE statements that reference each aesthetic:

```
GUIDE: legend(aesthetic(aesthetic.shape), label("Gender"))
GUIDE: legend(aesthetic(aesthetic.color), label("Previous Experience"))
```

When interpreting the color legend in the example, it's important to realize that the color aesthetic corresponds to a continuous variable. Only a handful of colors may be shown in the legend, and these colors do not reflect the whole spectrum of colors that could appear in the graph itself. They are more like mileposts at major divisions.

Chapter 2. GPL Statement and Function Reference

This section provides detailed information about the various statements that make up GPL and the functions that you can use in each of the statements.

GPL Statements

There are general categories of GPL statements.

Data definition statements. Data definition statements specify the data sources, variables, and optional variable transformations. All GPL code blocks include at least two data definition statements: one to define the actual data source and one to specify the variable extracted from the data source.

Specification statements. Specification statements define the graph. They define the axis scales, coordinate systems, text, graphic elements (for example, bars and points), and statistics. All GPL code blocks require at least one ELEMENT statement, but the other specification statements are optional. GPL uses a default value when the SCALE, COORD, and GUIDE statements are not included in the GPL code block.

Control statements. Control statements specify the layout for graphs. The GRAPH statement allows you to group multiple graphs in a single page display. For example, you may want to add histograms to the borders on a scatterplot. The PAGE statement allows you to set the size of the overall visualization. Control statements are optional.

Comment statement. The COMMENT statement is used for adding comments to the GPL. These are optional.

Data Definition Statements
“SOURCE Statement ” on page 21
“DATA Statement ” on page 21
“TRANS Statement ” on page 22
Specification Statements
“COORD Statement ” on page 22
“SCALE Statement ” on page 27
“GUIDE Statement ” on page 38
“ELEMENT Statement ” on page 43
Control Statements
“PAGE Statement ” on page 20
“GRAPH Statement ” on page 20
Comment Statements
“COMMENT Statement ” on page 19

COMMENT Statement

Syntax

```
COMMENT: <text>
```

<text>. The comment text. This can consist of any string of characters except a statement label followed by a colon (:), unless the statement label and colon are enclosed in quotes (for example, `COMMENT: With "SCALE:" statement`).

Description

This statement is optional. You can use it to add comments to your GPL or to comment out a statement by converting it to a comment. The comment does not appear in the resulting graph.

Examples

```
COMMENT: This graph shows counts for each job category.
```

Figure 15. Defining a comment

PAGE Statement

Syntax

```
PAGE: <function>
```

<function>. A function for specifying the PAGE statements that mark the beginning and end of the visualization.

Description

This statement is optional. It's needed only when you specify a size for the page display or visualization. The current release of GPL supports only one PAGE block.

If you are using GRAPH blocks in a PAGE block, note the following:

- SOURCE and DATA statements must appear directly in the PAGE block, followed by the GRAPH blocks. (See the second example, "Defining a page with multiple graphs.")
- TRANS, COORD, SCALE, GUIDE, and ELEMENT statements cannot appear directly in the PAGE block. These statements apply to individual graphs and must appear in the GRAPH block to which they apply.

Examples

```
PAGE: begin(scale(400px,300px))
SOURCE: s=csvSource(file("mydata.csv"))
DATA: x=col(source(s), name("x"))
DATA: y=col(source(s), name("y"))
ELEMENT: line(position(x*y))
PAGE: end()
```

Figure 16. Example: Defining a page

```
PAGE: begin(scale(400px,300px))
SOURCE: s=csvSource(file("mydata.csv"))
DATA: a=col(source(s), name("a"))
DATA: b=col(source(s), name("b"))
DATA: c=col(source(s), name("c"))
GRAPH: begin(scale(90%, 45%), origin(10%, 50%))
ELEMENT: line(position(a*c))
GRAPH: end()
GRAPH: begin(scale(90%, 45%), origin(10%, 0%))
ELEMENT: line(position(b*c))
GRAPH: end()
PAGE: end()
```

Figure 17. Example: Defining a page with multiple graphs

Valid Functions

“begin Function (For GPL Pages)” on page 65
“end Function ” on page 117

GRAPH Statement

Syntax

```
GRAPH: <function>
```

<function>. A function for specifying the GRAPH statements that mark the beginning and end of the individual graph.

Description

This statement is optional. It's needed only when you want to group multiple graphs in a single page display or you want to customize a graph's size. The GRAPH statement is essentially a wrapper around the

GPL that defines a particular graph. There is no limit to the number of graphs that can appear in a GPL block.

Grouping graphs is useful for related graphs, like graphs on the borders of histograms. However, the graphs do not have to be related. You may simply want to group the graphs for presentation.

For information about organization of statements when there are multiple GRAPH statements, see “PAGE Statement ” on page 20 .

Examples

```
GRAPH: begin(scale(50%,50%))
```

Figure 18. Scaling a graph

```
GRAPH: begin(origin(10.0%, 20.0%), scale(80.0%, 80.0%))
ELEMENT: point(position(salbegin*salary))
GRAPH: end()
GRAPH: begin(origin(10.0%, 100.0%), scale(80.0%, 10.0%))
ELEMENT: interval(position(summary.count(bin.rect(salbegin))))
GRAPH: end()
GRAPH: begin(origin(90.0%, 20.0%), scale(10.0%, 80.0%))
COORD: transpose()
ELEMENT: interval(position(summary.count(bin.rect(salary))))
GRAPH: end()
```

Figure 19. Example: Scatterplot with border histograms

Valid Functions

“begin Function (For GPL Graphs)” on page 65
“end Function ” on page 117

SOURCE Statement

Syntax

```
SOURCE: <source name> = <function>
```

<source name>. User-defined name for the data source. Refer to “GPL Syntax Rules” on page 2 for information about which characters you can use in the name.

<function>. A function for extracting data from various data sources.

Description

Defines a data source for the graph. There can be multiple data sources, each specified by a different SOURCE statement.

Examples

```
SOURCE: mydata = csvSource(path("/Data/demo.csv"))
```

Figure 20. Example: Reading a CSV file

Valid Functions

“csvSource Function ” on page 86
“savSource Function ” on page 202
“sqlSource Function ” on page 227
“userSource Function ” on page 303

DATA Statement

Syntax

```
DATA: <variable name> = <function>
```

<variable name>. User-defined name for the variable. Refer to “GPL Syntax Rules” on page 2 for information about which characters you can use in the name.

<function>. A function indicating the data sources.

Description

Defines a variable from a specific data source. The GPL statement must also include a SOURCE statement. The name identified by the SOURCE statement is used in the DATA statement to indicate the data source from which a particular variable is extracted.

Examples

```
DATA: age = col(source(mydata), name("age"))
```

Figure 21. Example: Specifying a variable from a data source

age is an arbitrary name. In this example, the variable name is the same as the name that appears in the data source. Using the same name avoids confusion. The *col* function takes a data source and data source variable name as its arguments. Note that the data source name was previously defined by a SOURCE statement and is not enclosed in quotes.

Valid Functions

“col Function ” on page 79
“iter Function ” on page 128

TRANS Statement

Syntax

```
TRANS: <variable name> = <function>
```

<variable name>. A string that specifies a name for the variable that is created as a result of the transformation. Refer to “GPL Syntax Rules” on page 2 for information about which characters you can use in the name.

<function>. A valid function.

Description

Defines a new variable whose value is the result of a data transformation function.

Examples

```
TRANS: saldiff = eval(((salary-salbegin)/salary)*100)
```

Figure 22. Example: Creating a transformation variable from other variables

```
TRANS: casenum = index()
```

Figure 23. Example: Creating an index variable

Valid Functions

“collapse Function ” on page 80
“eval Function ” on page 118
“index Function ” on page 128

COORD Statement

Syntax

```
COORD: <coord>
```

<coord>. A valid coordinate type or transformation function.

Description

Specifies a coordinate system for the graph. You can also embed coordinate systems or wrap a coordinate system in a transformation. When transformations and coordinate systems are embedded in each other, they are applied in order, with the innermost being applied first. Thus, `mirror(transpose(rect(1,2)))` specifies that a 2-D rectangular coordinate system is transposed and then mirrored.

Examples

```
COORD: polar.theta()
```

Figure 24. Example: Polar coordinates for pie charts

```
COORD: rect(dim(1,2,3))
```

Figure 25. Example: 3-D rectangular coordinates

```
COORD: rect(dim(2), polar.theta(dim(1)))
```

Figure 26. Example: Embedded coordinate systems for paneled pie chart

```
COORD: transpose()
```

Figure 27. Example: Transposed coordinate system

Coordinate Types and Transformations

“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“mirror Function ” on page 174
“project Function ” on page 182
“reflect Function ” on page 183
“transpose Function ” on page 301
“wrap Function ” on page 305

GPL Coordinate Types

There are several coordinate types available in GPL.

Coordinate Types

“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26

parallel Coordinate Type

Syntax

```
parallel(dim(<numeric>), <coord>)
```

<numeric>. One or more numeric values (separated by commas) indicating the graph dimension or dimensions to which the parallel coordinate system applies. The number of values equals the number of dimensions for the coordinate system's frame, and the values are always in sequential order. For example, `parallel(dim(1,2,3,4))` indicates that the first four variables in the algebra are used for the parallel coordinates. Any others are used for faceting. If no dimensions are specified, all variables are used for the parallel coordinates. See the topic “dim Function ” on page 115 for more information.

<coord>. A valid coordinate type or transformation function. This is optional.

Description

Creates a parallel coordinate system. A graph using this coordinate system consists of multiple, parallel axes showing data across multiple variables, resulting in a plot that is similar to a profile plot.

When you use a parallel coordinate system, you cross each continuous variable in the algebra. A line graphic element is the most common element type for this graph. The graphic element is always distinguished by some aesthetic so that any patterns are readily apparent.

Examples

```
TRANS: caseid = index()
COORD: parallel()
ELEMENT: line(position(var1*var2*var3*var4), split(caseid), color(catvar1))
```

The example includes the `split` function to create a separate line for each case in the data. Otherwise, there would be only one line that crossed back through the coordinate system to connect all the cases.

Figure 28. Example: Parallel coordinates graph

Coordinate Types and Transformations

“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“mirror Function ” on page 174
“project Function ” on page 182
“reflect Function ” on page 183
“transpose Function ” on page 301
“wrap Function ” on page 305

Applies To

“COORD Statement ” on page 22
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“project Function ” on page 182

polar Coordinate Type

Syntax

```
polar(dim(<numeric>), <function>, <coord>)
```

<numeric>. Numeric values (separated by commas) indicating the dimensions to which the polar coordinates apply. This is optional and is assumed to be the first two dimensions. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<coord>. A valid coordinate type or transformation function. This is optional.

Description

Creates a polar coordinate system. This differs from the polar.theta coordinate system in that it is two dimensional. One dimension is associated with the radius, and the other is associated with the theta angle.

Examples

```
COORD: polar()
ELEMENT: line(position(date*close), closed(), preserveStraightLines())
```

Figure 29. Example: Polar line chart

Valid Functions
“reverse Function ” on page 201
“startAngle Function ” on page 228
Coordinate Types and Transformations
“parallel Coordinate Type ” on page 23
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“mirror Function ” on page 174
“project Function ” on page 182
“reflect Function ” on page 183
“transpose Function ” on page 301
“wrap Function ” on page 305
Applies To
“COORD Statement ” on page 22
“parallel Coordinate Type ” on page 23
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“project Function ” on page 182

polar.theta Coordinate Type

Syntax

```
polar.theta(<function>, <coord>)
```

<numeric>. A numeric value indicating the dimension. This is optional and required only when `polar.theta` is not the first or innermost dimension. Otherwise, it is assumed to be the first dimension. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<coord>. A valid coordinate type or transformation function. This is optional.

Description

Creates a polar.theta coordinate system, which is the coordinate system for creating pie charts. polar.theta differs from the polar coordinate system in that it is one dimensional. This is the dimension for the theta angle.

Examples

```
COORD: polar.theta()
ELEMENT: interval.stack(position(summary.count()), color(jobcat))
```

Figure 30. Example: Pie chart

Valid Functions
“reverse Function ” on page 201
“startAngle Function ” on page 228
Coordinate Types and Transformations
“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“rect Coordinate Type ” on page 26
“mirror Function ” on page 174
“project Function ” on page 182
“reflect Function ” on page 183
“transpose Function ” on page 301

“wrap Function ” on page 305

Applies To

“COORD Statement ” on page 22
“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“rect Coordinate Type ” on page 26
“project Function ” on page 182

rect Coordinate Type

Syntax

```
rect(dim(<numeric>), <function>, <coord>)
```

<numeric>. One or more numeric values (separated by commas) indicating the graph dimension or dimensions to which the rectangular coordinate system applies. The number of values equals the number of dimensions for the coordinate system's frame, and the values are always in sequential order (for example, `dim(1,2,3)` and `dim(4,5)`). See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<coord>. A valid coordinate type or transformation function. This is optional.

Description

Creates a rectangular coordinate system. By default, a rectangular coordinate system is 2-D, which is the equivalent of specifying `rect(dim(1,2))`. To create a 3-D coordinate system, use `rect(dim(1,2,3))`. Similarly, use `rect(dim(1))` to specify a 1-D coordinate system. Changing the coordinate system also changes which variable in the algebra is summarized for a statistic. The statistic function is calculated on the second crossed variable in a 2-D coordinate system and the third crossed variable in a 3-D coordinate system.

Examples

```
COORD: rect(dim(1,2))
ELEMENT: interval(position(summary.mean(jobcat*salary)))
```

Figure 31. Example: 2-D bar chart

```
COORD: rect(dim(1,2,3))
ELEMENT: interval(position(summary.mean(jobcat*gender*salary)))
```

Figure 32. Example: 3-D bar chart

Valid Functions

“cluster Function ” on page 77
“sameRatio Function ” on page 202

Coordinate Types and Transformations

“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“mirror Function ” on page 174
“project Function ” on page 182
“reflect Function ” on page 183
“transpose Function ” on page 301
“wrap Function ” on page 305

Applies To

“COORD Statement ” on page 22
“parallel Coordinate Type ” on page 23

“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“project Function ” on page 182

SCALE Statement

Syntax

```
SCALE: <scale type>
```

or

```
SCALE: <scale name> = <scale type>
```

<scale type>. A valid scale type.

<scale name>. A user-defined name for the scale. This is required only when you are creating a graph with dual scales. An example is a graph that shows the mean of a variable on one axis and the count on the other. The scale name is referenced by an axis and a graphic element to indicate which scale is associated with the axis and graphic element. Refer to “GPL Syntax Rules” on page 2 for information about which characters you can use in the name.

Description

Defines the scale for a particular dimension or aesthetic (such as color).

Examples

```
SCALE: linear(dim(2), max(50000))
```

Figure 33. Example: Specifying a linear dimension scale

```
SCALE: log(aesthetic(aesthetic.color))
ELEMENT: point(position(x*y), color(z))
```

Figure 34. Example: Specifying a log aesthetic scale

```
SCALE: y1 = linear(dim(2))
SCALE: y2 = linear(dim(2))
GUIDE: axis(dim(1), label("Employment Category"))
GUIDE: axis(scale(y1), label("Mean Salary"))
GUIDE: axis(scale(y2), label("Count"), opposite(), color(color.red))
ELEMENT: interval(scale(y1), position(summary.mean(jobcat*salary)))
ELEMENT: line(scale(y2), position(summary.count(jobcat)), color(color.red))
```

Figure 35. Example: Creating a graph with dual scales

Scale Types

“asn Scale ” on page 28
“atanh Scale ” on page 28
“cat Scale ” on page 29
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“logit Scale ” on page 33
“pow Scale ” on page 33
“prob Scale ” on page 34
“probit Scale ” on page 35
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

GPL Scale Types

There are several scale types available in GPL.

Scale Types

“asn Scale ” on page 28
“atanh Scale ” on page 28
“cat Scale ” on page 29
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“logit Scale ” on page 33
“pow Scale ” on page 33
“prob Scale ” on page 34
“probit Scale ” on page 35
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

asn Scale

Syntax

```
asn(dim(<numeric>), <function>)
```

or

```
asn(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the `ELEMENT` statement.

Description

Creates an arcsine scale. Data values for this scale must fall in the closed interval [0, 1]. That is, for any data value x, $0 \leq x \leq 1$.

Example

```
SCALE: asn(dim(1))
```

Figure 36. Example: Specifying an arcsine scale

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61

Applies To

“SCALE Statement ” on page 27

atanh Scale

Syntax

```
atanh(dim(<numeric>), <function>)
```

or

```
atanh(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

Description

Creates a Fisher's *z* scale (also called the hyberbolic arctangent scale). Data values for this scale must fall in the open interval (-1, 1). That is, for any data value *x*, -1 < *x* < 1.

Example

```
SCALE: atanh(dim(1))
```

Figure 37. Example: Specifying a Fisher's z scale

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61

Applies To

“SCALE Statement ” on page 27

cat Scale

Syntax

```
cat(dim(<numeric>), <function>)
```

or

```
cat(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function in the ELEMENT statement, often used for distinguishing groups of graphic elements, as in clusters and stacks.

Description

Creates a categorical scale that can be associated with a dimension (such as an axis or panel facet) or an aesthetic (as in the legend). A categorical scale is created for a categorical variable by default. You usually don't need to specify the scale unless you want to use a function to modify it (for example, to sort the categories).

Examples

```
SCALE: cat(dim(1), sort.natural())
```

Figure 38. Example: Specifying the scale for a dimension and sorting categories

```
SCALE: cat(aesthetic(aesthetic.color), include("IL"))
```

Figure 39. Example: Specifying the scale for an aesthetic and including categories

Note: The `exclude` function is not supported for aesthetic scales.

Valid Functions

“aestheticMissing Function ” on page 61
“exclude Function ” on page 122
“include Function ” on page 127
“map Function ” on page 172
“reverse Function ” on page 201
“sort.data Function ” on page 225
“sort.natural Function ” on page 226
“sort.statistic Function ” on page 226
“sort.values Function ” on page 226

Applies To

“SCALE Statement ” on page 27

cLoglog Scale

Syntax

```
cLoglog(dim(<numeric>), <function>)
```

or

```
cLoglog(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

Description

Creates a complementary log-log-transformed scale (also called a Weibull scale). The formula for the transformation is `log(log(1/(1-x))`. Data values for this scale must fall in the open interval (0, 1). That is, for any data value *x*, 0 < *x* < 1.

Example

```
SCALE: cLoglog(dim(2))
```

Figure 40. Example: Specifying a Weibull scale

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61
“dataMaximum Function ” on page 87
“dataMinimum Function ” on page 87
“include Function ” on page 127
“max Function ” on page 173
“min Function ” on page 174
“origin Function (For GPL Scales)” on page 179
“reverse Function ” on page 201
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“asn Scale ” on page 28

“atanh Scale ” on page 28
“cat Scale ” on page 29
“linear Scale ” on page 31
“log Scale ” on page 32
“logit Scale ” on page 33
“pow Scale ” on page 33
“prob Scale ” on page 34
“probit Scale ” on page 35
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

Applies To

“SCALE Statement ” on page 27
“asn Scale ” on page 28
“atanh Scale ” on page 28
“cat Scale ” on page 29
“linear Scale ” on page 31
“log Scale ” on page 32
“logit Scale ” on page 33
“pow Scale ” on page 33
“prob Scale ” on page 34
“probit Scale ” on page 35
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

linear Scale

Syntax

```
linear(dim(<numeric>), <function>)
```

or

```
linear(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

Description

Creates a linear scale. A linear scale is created for a continuous variable by default. You usually don't need to specify the scale unless you want to add a function to modify it (for example, to specify the range).

Example

```
SCALE: linear(dim(2), max(50000))
```

Figure 41. Example: Specifying a maximum value for the linear scale

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60

“aestheticMissing Function ” on page 61
“dataMaximum Function ” on page 87
“dataMinimum Function ” on page 87
“include Function ” on page 127
“max Function ” on page 173
“min Function ” on page 174
“origin Function (For GPL Scales)” on page 179
“reverse Function ” on page 201

Applies To

“SCALE Statement ” on page 27

log Scale

Syntax

```
log(dim(<numeric>), <function>)
```

or

```
log(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

Description

Creates a logarithmic-transformed scale. The formula for the transformation is `log<base>(x)`, where <base> is defined by the `base` function. If a base is not explicitly specified, the default is base 10. Data values for this scale must be greater than 0.

Example

```
SCALE: log(dim(2), base(2))
```

Figure 42. Example: Specifying the scale and a base

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61
“dataMaximum Function ” on page 87
“dataMinimum Function ” on page 87
“include Function ” on page 127
“base Function ” on page 62
“max Function ” on page 173
“min Function ” on page 174
“origin Function (For GPL Scales)” on page 179
“reverse Function ” on page 201

Applies To

“SCALE Statement ” on page 27

logit Scale

Syntax

```
logit(dim(<numeric>), <function>)
```

or

```
logit(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

Description

Creates a logit-transformed scale. The formula for the transformation is `log(1/(1-x))`. Data values for this scale must fall in the open interval (0, 1). That is, for any data value *x*, 0 < *x* < 1.

Example

```
SCALE: logit(dim(2))
```

Figure 43. Example: Specifying a logit scale

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61

Applies To

“SCALE Statement ” on page 27

pow Scale

Syntax

```
pow(dim(<numeric>), <function>)
```

or

```
pow(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

Description

Creates a power scale. an exponent-transformed scale. The formula for the transformation is `power(x, <exponent>)`, where `<exponent>` is defined by the `exponent` function. If an exponent is not explicitly specified, the default is 0.5.

Example

```
SCALE: pow(dim(2), exponent(2))
```

Figure 44. Example: Specifying the scale and an exponent

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61
“dataMaximum Function ” on page 87
“dataMinimum Function ” on page 87
“exponent Function ” on page 123
“include Function ” on page 127
“max Function ” on page 173
“min Function ” on page 174
“origin Function (For GPL Scales)” on page 179
“reverse Function ” on page 201

Applies To

“SCALE Statement ” on page 27

prob Scale

Syntax

```
prob(dim(<numeric>), <distribution function>, <function>)
```

or

```
prob(aesthetic(aesthetic.<aesthetic type>), <distribution function>, <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<distribution function>. A distribution function. This is required.

<function>. One or more valid functions. These are optional.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the `ELEMENT` statement.

Description

Creates a probability scale based on the inverse cumulative distribution function (CDF) for the specified distribution. Data values for this scale must fall in the open interval (0, 1). That is, for any data value x, $0 < x < 1$.

Example

```
SCALE: prob(dim(2), beta(2, 5))
```

Figure 45. Example: Specifying a beta distribution for the probability scale

Distribution Functions

“beta Function ” on page 66
“chiSquare Function ” on page 77
“exponential Function ” on page 123
“f Function ” on page 123
“gamma Function ” on page 126
“logistic Function ” on page 172
“normal Function ” on page 178
“poisson Function ” on page 180
“studentizedRange Function ” on page 229
“t Function ” on page 298
“uniform Function ” on page 302

“weibull Function ” on page 304

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61

Applies To

“SCALE Statement ” on page 27

probit Scale

Syntax

```
probit(dim(<numeric>), <function>)
```

or

```
probit(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

Description

Creates a probit scale. The formula for the transformation is the inverse cumulative distribution function (CDF) of the normal distribution. Data values for this scale must fall in the closed interval [0, 1]. That is, for any data value x, $0 \leq x \leq 1$.

Note that `probit(dim(<numeric>))` is equivalent to `prob(dim(<numeric>), prob(normal(0, 1))`.

Example

```
probit(dim(2))
```

Figure 46. Example: Specifying a probit scale

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61

Applies To

“SCALE Statement ” on page 27

safeLog Scale

Syntax

```
safeLog(dim(<numeric>), <function>)
```

or

```
safeLog(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

Description

Creates a "safe" logarithmic-transformed scale. Unlike a regular log scale, the safe log scale uses a modified function to handle 0 and negative values. If a base is not explicitly specified, the default is base 10.

Formula for Safe Log Transformation

The safe log formula is:

```
sign(x) * log(1 + abs(x))
```

So if you assume that the axis value is –99, the result of the transformation is:

```
sign(-99) * log(1 + abs(-99)) = -1 * log(1 + 99) = -1 * 2 = -2
```

Example

```
SCALE: safeLog(dim(2), base(2))
```

Figure 47. Example: Specifying the scale and a base

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61
“dataMaximum Function ” on page 87
“dataMinimum Function ” on page 87
“include Function ” on page 127
“base Function ” on page 62
“max Function ” on page 173
“min Function ” on page 174
“origin Function (For GPL Scales)” on page 179
“reverse Function ” on page 201

Applies To

“SCALE Statement ” on page 27

safePower Scale

Syntax

```
safePower(dim(<numeric>), <function>)
```

or

```
safePower(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

Description

Creates a "safe" power-transformed scale. Unlike a regular power scale, the safe power scale uses a modified function to handle negative values. If an exponent is not explicitly specified, the default is 0.5.

Formulas for Safe Power Transformation

When the exponent is a *positive* number, the safe power formulas are:

```
If (x>=0):
pow(1+x, exponent)-1
If (x<0):
1-pow(1-x, exponent)
```

When the exponent is a *negative* number, the safe power formulas are:

```
If (x>=0):
1-pow(1+x, exponent)
If (x<0):
pow(1-x, exponent)-1
```

So, if you assume the axis value is –99 and the exponent is 0.5, the result of the transformation is:

```
1 - pow(1 - (-99), 0.5) = 1 - pow(100, 0.5) = 1 - 10 = -9
```

So, if you assume the axis value is –99 and the exponent is –2, the result of the transformation is:

```
pow(1 - (-99), -2) - 1 = pow(100, -2) - 1 = 0.0001 - 1 = -0.999
```

Example

```
SCALE: safePower(dim(2), exponent(10))
```

Figure 48. Example: Specifying the scale and an exponent

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61
“dataMaximum Function ” on page 87
“dataMinimum Function ” on page 87
“exponent Function ” on page 123
“include Function ” on page 127
“max Function ” on page 173
“min Function ” on page 174
“origin Function (For GPL Scales)” on page 179
“reverse Function ” on page 201

Applies To

“SCALE Statement ” on page 27

time Scale

Syntax

```
time(dim(<numeric>), <function>)
```

or

```
time(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<aesthetic type>. An aesthetic type indicating the aesthetic to which the scale applies. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the `ELEMENT` statement.

Description

Creates a time scale.

Example

```
SCALE: time(dim(1), max("12/31/2005"))
```

Figure 49. Example: Specifying a maximum value for the time scale

Valid Functions

“aestheticMaximum Function ” on page 59
“aestheticMinimum Function ” on page 60
“aestheticMissing Function ” on page 61
“dataMaximum Function ” on page 87
“dataMinimum Function ” on page 87
“include Function ” on page 127
“max Function ” on page 173
“min Function ” on page 174
“origin Function (For GPL Scales)” on page 179
“reverse Function ” on page 201

Applies To

“SCALE Statement ” on page 27

GUIDE Statement

Syntax

```
GUIDE: <guide type>
```

<guide type>. A valid guide type.

Description

Specifies a guide for the graph. Guides provide additional information that can help a viewer interpret the graph. An axis is a guide because it provides labels for the categories and values in a graph (but is separate from the scale on which the data are drawn). A title is a guide because it describes the graph. A legend is a guide because it provides color swatches to match a category name to specific instances of a graphic element in the graph.

Examples

```
GUIDE: axis(dim(2), label("Mean Current Salary"))
```

Figure 50. Example: Axis

Guide Types

“axis Guide Type ” on page 39
“form.line Guide Type ” on page 40
“legend Guide Type ” on page 40
“text.footnote Guide Type ” on page 41
“text.subfootnote Guide Type ” on page 41
“text.subsubfootnote Guide Type ” on page 42
“text.subtitle Guide Type ” on page 42
“text.subsubtitle Guide Type ” on page 42
“text.title Guide Type ” on page 43

GPL Guide Types

There are several different types of guides.

Guide Types

“axis Guide Type ” on page 39
“form.line Guide Type ” on page 40
“legend Guide Type ” on page 40
“text.footnote Guide Type ” on page 41
“text.subfootnote Guide Type ” on page 41
“text.subsubfootnote Guide Type ” on page 42
“text.subtitle Guide Type ” on page 42
“text.subsubtitle Guide Type ” on page 42
“text.title Guide Type ” on page 43

axis Guide Type

Syntax

```
axis(dim(<numeric>), <function>)
```

or

```
axis(scale(<scale name>), <function>)
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. Use the `null()` function to hide the axis.

<scale name>. The name of the scale to which the axis applies. See the topic “scale Function (For GPL Axes)” on page 202 for more information.

Description

Specifies the axis for a particular dimension. Do not confuse the axis with the scale. They are separate parts of the graph and are handled with separate GPL statements. The axis helps interpret the scale with labels and tick marks, but the axis does not change the positioning of graphic elements as changing the scale would. For information about scales, see “SCALE Statement ” on page 27.

Examples

```
GUIDE: axis(dim(2), label("Mean Current Salary"))
```

Figure 51. Example: Specifying an axis label for a dimension with one scale

```
SCALE: y1 = linear(dim(2))
SCALE: y2 = linear(dim(2))
GUIDE: axis(scale(y1), label("Mean Salary"))
GUIDE: axis(scale(y2), label("Count"), opposite(), color(color.red))
```

Figure 52. Example: Specifying axis labels for a dimension with two scales

Valid Functions

“color Function (For GPL Guides)” on page 82
“delta Function ” on page 88
“format.date Function ” on page 124
“format.dateTime Function ” on page 124
“format.time Function ” on page 125
“gap Function ” on page 126
“gridlines Function ” on page 126
“label Function (For GPL Guides)” on page 130

“opposite Function ” on page 178
“start Function ” on page 228
“ticks Function ” on page 299
“unit.percent Function ” on page 302

Applies To

“GUIDE Statement ” on page 38

form.line Guide Type

Syntax

```
form.line(<function>)
```

<function>. One or more valid functions. The `position` function is required.

Description

Specifies a vertical or horizontal reference line. For information about specifying the location of the line, see “position Function (For GPL Guides)” on page 181 .

Examples

```
GUIDE: form.line(position(*, 5000))
```

Figure 53. Example: Horizontal reference line

```
GUIDE: form.line(position(5000, *))
```

Figure 54. Example: Vertical reference line

Valid Functions

“color.brightness Function (For GPL Guides)” on page 83
“color Function (For GPL Guides)” on page 82
“color.hue Function (For GPL Guides)” on page 84
“label Function (For GPL Guides)” on page 130
“position Function (For GPL Guides)” on page 181
“color.saturation Function (For GPL Guides)” on page 86
“scale Function (For GPL Graphic Elements and form.line)” on page 203
“shape Function (For GPL Guides)” on page 206
“size Function (For GPL Guides)” on page 208
“transparency Function (For GPL Guides)” on page 301

Applies To

“GUIDE Statement ” on page 38

legend Guide Type

Syntax

```
legend(aesthetic(aesthetic.<aesthetic type>), <function>)
```

<aesthetic type>. An aesthetic associated with the legend. The aesthetic identifies the legend, which was created as the result of an aesthetic function in the `ELEMENT` statement.

<function>. One or more valid functions. Use the `null()` function to hide the legend.

Description

Specifies properties of the legend associated with a specific aesthetic, which is defined by an aesthetic function in the `ELEMENT` statement. The legend provides a visual representation of the scale created by the aesthetic function. Thus, a legend guide is related to the aesthetic scale in the same way an axis guide

is related to a dimension scale. Note that using a uniform aesthetic value in the aesthetic function (for example, `color(color.blue)`) does not create a scale. Therefore, the legend is not used in that case.

Examples

```
GUIDE: legend(aesthetic(aesthetic.color), label("Gender"))
ELEMENT: interval(position(summary.count(jobcat)), color(gender))
```

Figure 55. Example: Specifying a legend title

Valid Functions
"label Function (For GPL Guides)" on page 130
Applies To
"GUIDE Statement " on page 38

text.footnote Guide Type

Syntax

```
text.footnote(<function>)
```

<function>. One or more valid functions.

Description

Specifies the footnote for the graph.

Examples

```
GUIDE: text.footnote(label("Some Text"))
```

Figure 56. Example: Specifying the footnote text

Valid Functions
"label Function (For GPL Guides)" on page 130
Applies To
"GUIDE Statement " on page 38

text.subfootnote Guide Type

Syntax

```
text.subfootnote(<function>)
```

<function>. One or more valid functions.

Description

Specifies the second-level footnote for the graph. That is, the subfootnote appears below the footnote.

Examples

```
GUIDE: text.subfootnote(label("Some Text"))
```

Figure 57. Example: Specifying the second-level footnote text

Valid Functions
"label Function (For GPL Guides)" on page 130
Applies To
"GUIDE Statement " on page 38

text.subsubfootnote Guide Type

Syntax

```
text.subsubfootnote(<function>)
```

<function>. One or more valid functions.

Description

Specifies the third-level footnote for the graph. That is, the subsubfootnote appears below the subfootnote.

Examples

```
GUIDE: text.subsubfootnote(label("Some Text"))
```

Figure 58. Example: Specifying the third-level footnote text

Valid Functions
“label Function (For GPL Guides)” on page 130
Applies To
“GUIDE Statement ” on page 38

text.subtitle Guide Type

Syntax

```
text.subtitle(<function>)
```

<function>. One or more valid functions.

Description

Specifies the subtitle for the graph.

Examples

```
GUIDE: text.subtitle(label("Some Text"))
```

Figure 59. Example: Specifying the subtitle text

Valid Functions
“label Function (For GPL Guides)” on page 130
Applies To
“GUIDE Statement ” on page 38

text.subsubtitle Guide Type

Syntax

```
text.subsubtitle(<function>)
```

<function>. One or more valid functions.

Description

Specifies the second-level subtitle for the graph.

Examples

```
GUIDE: text.subsubtitle(label("Some Text"))
```

Figure 60. Example: Specifying the second-level subtitle text

Valid Functions
“label Function (For GPL Guides)” on page 130
Applies To
“GUIDE Statement ” on page 38

text.title Guide Type

Syntax

```
text.title(<function>)
```

<function>. One or more valid functions.

Description

Specifies the title for the graph.

Examples

```
GUIDE: text.title(label("Salary by Gender"))
```

Figure 61. Example: Specifying the title text

Valid Functions
“label Function (For GPL Guides)” on page 130
Applies To
“GUIDE Statement ” on page 38

ELEMENT Statement

Syntax

```
ELEMENT: <element type>
```

<element type>. A valid element type.

Description

Specifies a graphic element used to draw data on the graph. Graphic elements are the bars, points, lines, etc., that make up a graph.

There can be multiple ELEMENT statements in the same block of GPL to create multiple graphic elements. In this case, the variables in multiple ELEMENT statements share the same dimension and aesthetic scales.

For example, assume the GPL includes the following ELEMENT statements:

```
ELEMENT: point(position(x*y))
ELEMENT: point(position(x2*y2))
```

In the resulting graph, dimension 1 uses x+x2, and dimension 2 uses y+y2.

Note: This behavior is sometimes called an **implied blend** because we could have written the GPL as follows:

```
ELEMENT: point(position(x*y+x2*y2))
```

Examples

```
ELEMENT: point(position(x*y))
```

Figure 62. Example: Scatterplot

```
ELEMENT: line(position(x*y))
```

Figure 63. Example: Line chart

```
ELEMENT: interval(position(summary.mean(x*y)))
```

Figure 64. Example: Bar chart of means

Graphic Element Types
“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

GPL Graphic Element Types

There are several different types of elements for drawing data on a graph.

area Element

Syntax

```
area.<collision modifier>(<function>)
```

<collision modifier>. A method that specifies what should happen when two area graphic elements overlap each other. The collision modifier is optional.

<function>. One or more valid functions. The `position` function is required. The `scale` function is required when there are multiple scales in a single dimension (as in a "dual axis" graph).

Description

Specifies an area graphic element.

Examples

```
ELEMENT: area(position(summary.mean(jobcat*gender)))
```

Figure 65. Example: Area chart

Collision Modifiers
“difference Collision Modifier ” on page 51
“stack Collision Modifier ” on page 53
Valid Functions
“closed Function ” on page 77
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“jump Function ” on page 128
“label Function (For GPL Graphic Elements)” on page 129
“missing.gap Function ” on page 175
“missing.interpolate Function ” on page 175
“missing.wings Function ” on page 176

“position Function (For GPL Graphic Elements)” on page 180
“preserveStraightLines Function ” on page 182
“scale Function (For GPL Graphic Elements and form.line)” on page 203
“split Function ” on page 227
“texture.pattern Function ” on page 298
“transparency Function (For GPL Graphic Elements)” on page 300
“visible Function ” on page 304

Applies To

“ELEMENT Statement ” on page 43

bar Element

Description

`bar` is an alias for `interval`. For the syntax and other examples, see “interval Element ” on page 46 .

Examples

```
ELEMENT: bar(position(summary.mean(jobcat*gender)))
```

Figure 66. Example: Bar chart

edge Element

Syntax

```
edge(<function>)
```

<function>. One or more valid functions. The `position` function is required. The `scale` function is required when there are multiple scales in a single dimension (as in a "dual axis" graph).

Description

Specifies a vertex-edge graphic element. The edge element is typically used in conjunction with a link or layout function, which calculates the actual links among the vertices. The edge element is a graphical representation of these links.

Examples

```
ELEMENT: edge(position(link.mst(x*y))
```

Figure 67. Example: Minimum spanning tree

```
ELEMENT: edge(position(link.hull(x*y))
```

Figure 68. Example: Convex hull

Valid Functions

“closed Function ” on page 77
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“label Function (For GPL Graphic Elements)” on page 129
“missing.gap Function ” on page 175
“missing.interpolate Function ” on page 175
“missing.wings Function ” on page 176
“position Function (For GPL Graphic Elements)” on page 180
“preserveStraightLines Function ” on page 182
“scale Function (For GPL Graphic Elements and form.line)” on page 203

“shape Function (For GPL Graphic Elements)” on page 205
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“transparency Function (For GPL Graphic Elements)” on page 300
“visible Function ” on page 304

Applies To

“ELEMENT Statement ” on page 43

interval Element

Syntax

```
interval.<collision modifier>(<function>)
```

<collision modifier>. A method that specifies what should happen when two interval graphic elements overlap each other. The collision modifier is optional.

<function>. One or more valid functions. The `position` function is required. The `scale` function is required when there are multiple scales in a single dimension (as in a "dual axis" graph).

Description

Specifies an interval (bar) graphic element, as would be used in a bar chart, histogram, or error bar chart.

Examples

```
ELEMENT: interval(position(summary.mean(jobcat*salary)))
```

Figure 69. Example: Bar chart

```
ELEMENT: interval(position(bin.rect(summary.count(salary))))
```

Figure 70. Example: Histogram

```
ELEMENT: interval(position(region.confi.mean(jobcat*salary)), shape(shape.ibeam))
```

Figure 71. Example: Error bar chart

```
ELEMENT: interval.stack(position(summary.sum(jobcat*salary)), color(gender))
```

Figure 72. Example: Stacked bar chart

Collision Modifiers

“dodge Collision Modifier ” on page 51
“stack Collision Modifier ” on page 53

Valid Functions

“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“label Function (For GPL Graphic Elements)” on page 129
“position Function (For GPL Graphic Elements)” on page 180
“scale Function (For GPL Graphic Elements and form.line)” on page 203
“shape Function (For GPL Graphic Elements)” on page 205
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“texture.pattern Function ” on page 298
“transparency Function (For GPL Graphic Elements)” on page 300

“visible Function ” on page 304

Applies To

“ELEMENT Statement ” on page 43

line Element

Syntax

```
line.<collision modifier>(<function>)
```

<collision modifier>. A method that specifies what should happen when two line graphic elements overlap each other. The collision modifier is optional.

<function>. One or more valid functions. The `position` function is required. The `scale` function is required when there are multiple scales in a single dimension (as in a "dual axis" graph).

Description

Specifies a line graphic element. The line is drawn through values in the order in which they appear in the *x*-axis domain. This is one of the features that distinguishes a line graphic element from the path graphic element. See “path Element ” on page 48 for more information about the path graphic element.

Examples

```
ELEMENT: line(position(salbegin*salary))
```

Figure 73. Example: Line chart of continuous variables

```
ELEMENT: line(position(summary.mean(jobcat*salary)))
```

Figure 74. Example: Line chart of a summary statistic

Collision Modifiers

“stack Collision Modifier ” on page 53

Valid Functions

“closed Function ” on page 77
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“jump Function ” on page 128
“label Function (For GPL Graphic Elements)” on page 129
“missing.gap Function ” on page 175
“missing.interpolate Function ” on page 175
“missing.wings Function ” on page 176
“position Function (For GPL Graphic Elements)” on page 180
“preserveStraightLines Function ” on page 182
“scale Function (For GPL Graphic Elements and form.line)” on page 203
“shape Function (For GPL Graphic Elements)” on page 205
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“transparency Function (For GPL Graphic Elements)” on page 300
“visible Function ” on page 304

Applies To

“ELEMENT Statement ” on page 43

path Element

Syntax

```
path(<function>)
```

<function>. One or more valid functions. The `position` function is required. The `scale` function is required when there are multiple scales in a single dimension (as in a "dual axis" graph).

Description

Specifies a path graphic element. The path graphic element connects the data values in the order in which their associated cases appear in the dataset. Therefore, it can cross back on itself. This is one of the features that distinguishes the path graphic element from the line graphic element. Additionally, paths can have variable sizes, so another variable can control the thickness of the path at any particular point.

Examples

```
ELEMENT: path(position(salbegin*salary))
```

Figure 75. Example: Line chart drawn through all values

```
ELEMENT: path(position(salbegin*salary), size(educ))
```

Figure 76. Example: Creating a line chart with variable widths

Valid Functions

“closed Function ” on page 77
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“label Function (For GPL Graphic Elements)” on page 129
“missing.gap Function ” on page 175
“missing.interpolate Function ” on page 175
“missing.wings Function ” on page 176
“position Function (For GPL Graphic Elements)” on page 180
“preserveStraightLines Function ” on page 182
“scale Function (For GPL Graphic Elements and form.line)” on page 203
“shape Function (For GPL Graphic Elements)” on page 205
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“transparency Function (For GPL Graphic Elements)” on page 300
“visible Function ” on page 304

Applies To

“ELEMENT Statement ” on page 43

point Element

Syntax

```
point.<collision modifier>(<function>)
```

<collision modifier>. A method that specifies what should happen when two points overlap each other. The collision modifier is optional.

<function>. One or more valid functions. The `position` function is required. The `scale` function is required when there are multiple scales in a single dimension (as in a "dual axis" graph).

Description

Specifies a point graphic element, as would be used in a scatterplot.

Examples

```
ELEMENT: point(position(salbegin*salary))
```

Figure 77. Example: Scatterplot

```
ELEMENT: point.dodge.symmetric(position(bin.dot(salary)))
```

Figure 78. Example: Dot plot

Collision Modifiers

“dodge.asymmetric Collision Modifier ” on page 51
“dodge.symmetric Collision Modifier ” on page 52
“jitter Collision Modifier ” on page 52
“stack Collision Modifier ” on page 53

Valid Functions

“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“label Function (For GPL Graphic Elements)” on page 129
“position Function (For GPL Graphic Elements)” on page 180
“scale Function (For GPL Graphic Elements and form.line)” on page 203
“shape Function (For GPL Graphic Elements)” on page 205
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“texture.pattern Function ” on page 298
“transparency Function (For GPL Graphic Elements)” on page 300
“visible Function ” on page 304

Applies To

“ELEMENT Statement ” on page 43

polygon Element

Syntax

```
polygon(<function>)
```

<function>. One or more valid functions. The `position` function is required. The `scale` function is required when there are multiple scales in a single dimension (as in a "dual axis" graph).

Description

Specifies a polygonal graphic element. A polygon connects multiple points to create an enclosed space. For example, it may be used to draw a state or country in a map, or it may be used to draw the outline of a two-dimensional bin.

Examples

```
ELEMENT: polygon(position(bin.hex(x*y, dim(1,2)), color(summary.count()))
```

Figure 79. Example: Hexagonal binning

Valid Functions

“color Function (For GPL Graphic Elements)” on page 81

“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“label Function (For GPL Graphic Elements)” on page 129
“position Function (For GPL Graphic Elements)” on page 180
“preserveStraightLines Function ” on page 182
“scale Function (For GPL Graphic Elements and form.line)” on page 203
“shape Function (For GPL Graphic Elements)” on page 205
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“transparency Function (For GPL Graphic Elements)” on page 300
“visible Function ” on page 304

Applies To

“ELEMENT Statement ” on page 43

schema Element

Syntax

```
schema(<function>)
```

<function>. One or more valid functions. The `position` function is required. The `scale` function is required when there are multiple scales in a single dimension (as in a "dual axis" graph).

Description

Specifies a schema (boxplot) graphic element.

Examples

```
ELEMENT: schema(position(bin.quantile.letter(jobcat*salary)))
```

Figure 80. Example: Boxplot

Valid Functions

“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“label Function (For GPL Graphic Elements)” on page 129
“position Function (For GPL Graphic Elements)” on page 180
“scale Function (For GPL Graphic Elements and form.line)” on page 203
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“texture.pattern Function ” on page 298
“transparency Function (For GPL Graphic Elements)” on page 300
“visible Function ” on page 304

Applies To

“ELEMENT Statement ” on page 43

GPL Collision Modifiers

Collision modifiers specify what happens when two graphic elements overlap.

difference Collision Modifier

Syntax

```
<element type>.difference
```

<element type>. A valid element type.

Description

Clips graphic elements and draws the difference between dichotomous values. The aesthetic value associated with the greater value in a group determines the aesthetic for the result. This function is used to create a difference area graph. It is useful to compare the result to the one obtained when using the `stack` position modifier.

Examples

```
ELEMENT: area.difference(position(summary.sum(jobcat*salary)), color(gender))
```

Figure 81. Example: Difference area chart

Valid Element Types
“area Element ” on page 44

dodge Collision Modifier

Syntax

```
<element type>.dodge
```

<element type>. A valid element type.

Description

Moves graphic elements next to other graphic elements that appear at the same value, rather than superimposing them. The graphic elements are arranged symmetrically. That is, the graphic elements are moved to opposite sides of a central position. For point elements, this function is the same as `dodge.symmetric`. (See “dodge.symmetric Collision Modifier ” on page 52 .)

Although dodged charts can look similar to clustered charts, they are not the same. Clustering changes the coordinates. Dodging only repositions graphic elements to avoid collisions. Therefore, a clustered chart allocates space for missing categories while a dodged chart does not.

Examples

```
ELEMENT: point.dodge(position(bin.dot(salary*jobcat)))
```

Figure 82. Example: 2-D Dot plot

```
ELEMENT: interval.dodge(position(summary.mean(jobcat*salary)), color(gender))
```

Figure 83. Example: Dodged bar chart

Valid Element Types
“interval Element ” on page 46
“point Element ” on page 48

dodge.asymmetric Collision Modifier

Syntax

```
<element type>.dodge.asymmetric
```

<element type>. A valid element type.

Description

Moves graphic elements next to other graphic elements that appear at the same value, rather than superimposing them. The graphic elements are arranged asymmetrically. That is, the graphic elements are stacked on top of one another, with the graphic element on the bottom positioned at a specific value on the scale. `dodge.asymmetric` is typically used for 1-D dot plots.

Examples

```
ELEMENT: point.dodge.asymmetric(position(bin.dot(salary)))
```

Figure 84. Example: 1-D Dot plot

Valid Element Types
“point Element ” on page 48

dodge.symmetric Collision Modifier

Syntax

```
<element type>.dodge.symmetric
```

<element type>. A valid element type.

Description

Moves graphic elements next to other graphic elements that appear at the same value, rather than superimposing them. The graphic elements are arranged symmetrically. That is, the graphic elements extend in two directions from a central position. `dodge.asymmetric` is typically used for 2-D dot plots.

Examples

```
ELEMENT: point.dodge.symmetric(position(bin.dot(salary*jobcat)))
```

Figure 85. Example: 2-D Dot plot

Valid Element Types
“point Element ” on page 48

jitter Collision Modifier

Syntax

```
<element type>.jitter.<jitter type>
```

<element type>. A valid element type.

<jitter type>. A valid jitter type. This is optional. If no type is specified, `jitter.joint.uniform` is used.

Description

Repositions graphic elements randomly using a normal or uniform distribution.

Examples

```
ELEMENT: point.jitter(position(jobcat*gender))
```

Figure 86. Example: Jittered categorical point chart

Table 1. Jitter types

Type	**Meaning**
`joint`	Same as `joint.uniform`
`conditional`	Same as `conditional.uniform`

Table 1. Jitter types (continued)	
Type	**Meaning**
`normal`	Same as `joint.normal`
`uniform`	Same as `joint.uniform`
`joint.normal`	Jitter points in all dimensions using a normal distribution
`joint.uniform`	Jitter points in all dimensions using a uniform distribution
`conditional.normal`	Jitter points in the analysis dimension using a normal distribution
`conditional.uniform`	Jitter points in the analysis dimension using a uniform distribution

Valid Element Types

“point Element ” on page 48

stack Collision Modifier

Syntax

```
<element type>.stack
```

<element type>. A valid element type.

Description

Stacks graphic elements that would normally be superimposed when they have the same data values.

Examples

```
ELEMENT: interval.stack(position(summary.mean(jobcat*salary)), color(gender))
```

Figure 87. Example: Stacked bar chart

Valid Element Types

“area Element ” on page 44
“interval Element ” on page 46
“line Element ” on page 47
“point Element ” on page 48

GPL Functions

Functions are used within GPL statements. Functions can also be embedded in other functions.

Aesthetic Functions

“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“shape Function (For GPL Graphic Elements)” on page 205
“size Function (For GPL Graphic Elements)” on page 207
“texture.pattern Function ” on page 298
“transparency Function (For GPL Graphic Elements)” on page 300
“visible Function ” on page 304

Aesthetic Mapping Functions

"aestheticMaximum Function " on page 59
"aestheticMinimum Function " on page 60
"aestheticMissing Function " on page 61
"map Function " on page 172

Guide Aesthetic Functions

"color Function (For GPL Guides)" on page 82
"color.brightness Function (For GPL Guides)" on page 83
"color.hue Function (For GPL Guides)" on page 84
"color.saturation Function (For GPL Guides)" on page 86
"shape Function (For GPL Guides)" on page 206
"size Function (For GPL Guides)" on page 208
"transparency Function (For GPL Guides)" on page 301

Data Functions

"DATA Statement " on page 21
"col Function " on page 79
"iter Function " on page 128

Data Source Functions

"SOURCE Statement " on page 21
"csvSource Function " on page 86
"savSource Function " on page 202
"sqlSource Function " on page 227
"userSource Function " on page 303

Binning Functions

"bin.dot Function " on page 66
"bin.hex Function " on page 68
"bin.quantile.letter Function " on page 70
"bin.rect Function " on page 72
"density.beta Function " on page 88
"density.chiSquare Function " on page 90
"density.exponential Function " on page 92
"density.f Function " on page 94
"density.gamma Function " on page 96
"density.logistic Function " on page 101
"density.normal Function " on page 103
"density.poisson Function " on page 105
"density.studentizedRange Function " on page 107
"density.t Function " on page 109
"density.uniform Function " on page 111
"density.weibull Function " on page 113
"link.alpha Function " on page 145
"link.complete Function " on page 147
"link.delaunay Function " on page 149
"link.distance Function " on page 151
"link.gabriel Function " on page 153
"link.hull Function " on page 155
"link.influence Function " on page 157
"link.join Function " on page 159
"link.mst Function " on page 162

“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296

Graph Control Functions

“GRAPH Statement ” on page 20
“begin Function (For GPL Graphs)” on page 65
“end Function ” on page 117

Missing Value Functions for Lines and Areas

“area Element ” on page 44
“edge Element ” on page 45
“line Element ” on page 47
“path Element ” on page 48
“missing.gap Function ” on page 175
“missing.interpolate Function ” on page 175
“missing.wings Function ” on page 176

Percentage Base Functions

"base.aesthetic Function " on page 63
"base.all Function " on page 63
"base.coordinate Function " on page 64
"region.confi.percent.count Function " on page 188
"region.confi.proportion.count Function " on page 190
"summary.percent.count Function " on page 254
"summary.percent.count.cumulative Function " on page 256
"summary.percent.sum Function " on page 260
"summary.percent.sum.cumulative Function " on page 262
"summary.proportion.count Function " on page 270
"summary.proportion.count.cumulative Function " on page 272
"summary.proportion.sum Function " on page 275
"summary.proportion.sum.cumulative Function " on page 277

Probability Scale Distribution Functions

"prob Scale " on page 34
"beta Function " on page 66
"chiSquare Function " on page 77
"exponential Function " on page 123
"f Function " on page 123
"gamma Function " on page 126
"logistic Function " on page 172
"normal Function " on page 178
"poisson Function " on page 180
"studentizedRange Function " on page 229
"t Function " on page 298
"uniform Function " on page 302
"weibull Function " on page 304

Sort Functions

"cat Scale " on page 29
"sort.data Function " on page 225
"sort.natural Function " on page 226
"sort.statistic Function " on page 226
"sort.values Function " on page 226

Statistic Functions

"density.beta Function " on page 88
"density.chiSquare Function " on page 90
"density.exponential Function " on page 92
"density.f Function " on page 94
"density.gamma Function " on page 96
"density.kernel Function " on page 98
"density.logistic Function " on page 101
"density.normal Function " on page 103
"density.poisson Function " on page 105
"density.studentizedRange Function " on page 107
"density.t Function " on page 109
"density.uniform Function " on page 111
"density.weibull Function " on page 113
"layout.circle Function " on page 131

“layout.dag Function ” on page 133
“layout.data Function ” on page 135
“layout.grid Function ” on page 137
“layout.network Function ” on page 139
“layout.random Function ” on page 141
“layout.tree Function ” on page 143
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“smooth.cubic Function ” on page 208
“smooth.linear Function ” on page 211
“smooth.loess Function ” on page 213
“smooth.mean Function ” on page 215
“smooth.median Function ” on page 217
“smooth.quadratic Function ” on page 220
“smooth.spline Function ” on page 222
“smooth.step Function ” on page 224
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256

“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296

Transformation Functions

“TRANS Statement ” on page 22
“collapse Function ” on page 80
“eval Function ” on page 118
“index Function ” on page 128

Other Functions

“alpha Function ” on page 62
“base Function ” on page 62
“begin Function (For GPL Graphs)” on page 65
“begin Function (For GPL Pages)” on page 65
“binCount Function ” on page 75
“binStart Function ” on page 75
“binWidth Function ” on page 76
“closed Function ” on page 77
“cluster Function ” on page 77
“dataMaximum Function ” on page 87
“dataMinimum Function ” on page 87
“delta Function ” on page 88
“dim Function ” on page 115
“end Function ” on page 117
“exclude Function ” on page 122
“exponent Function ” on page 123
“format Function ” on page 124
“format.date Function ” on page 124
“format.dateTime Function ” on page 124
“format.time Function ” on page 125
“from Function ” on page 125
“gap Function ” on page 126
“gridlines Function ” on page 126

“in Function ” on page 127
“include Function ” on page 127
“jump Function ” on page 128
“label Function (For GPL Graphic Elements)” on page 129
“label Function (For GPL Guides)” on page 130
“marron Function ” on page 173
“max Function ” on page 173
“min Function ” on page 174
“missing.listwise Function ” on page 175
“missing.pairwise Function ” on page 176
“multiple Function ” on page 177
“noConstant Function ” on page 177
“node Function ” on page 177
“notIn Function ” on page 178
“opposite Function ” on page 178
“origin Function (For GPL Graphs)” on page 179
“origin Function (For GPL Scales)” on page 179
“position Function (For GPL Graphic Elements)” on page 180
“position Function (For GPL Guides)” on page 181
“preserveStraightLines Function ” on page 182
“proportion Function ” on page 183
“reverse Function ” on page 201
“root Function ” on page 201
“sameRatio Function ” on page 202
“scale Function (For GPL Axes)” on page 202
“scale Function (For GPL Graphs)” on page 203
“scale Function (For GPL Graphic Elements and form.line)” on page 203
“scale Function (For GPL Pages)” on page 204
“scaledToData Function ” on page 204
“segments Function ” on page 205
“showAll Function ” on page 207
“split Function ” on page 227
“start Function ” on page 228
“startAngle Function ” on page 228
“ticks Function ” on page 299
“to Function ” on page 300
“unit.percent Function ” on page 302
“values Function ” on page 303
“weight Function ” on page 305

aestheticMaximum Function

Syntax

```
aestheticMinimum(<aesthetic type>.<aesthetic constant>)
```

or

```
aestheticMinimum(<aesthetic type>."aesthetic value")
```

<aesthetic type>. An aesthetic type indicating the specific aesthetic for which a maximum value is being specified. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

<aesthetic constant>. A constant for the aesthetic (for example, `size.huge`). Valid constants depend on the aesthetic.

"aesthetic value". A specific value for the aesthetic (for example, `size."10px"`. Valid values depend on the aesthetic.

Description

Specifies an aesthetic value that is mapped to the maximum data value for a scale. The maximum data value may be a "nice value" based on the data (the default), the exact data maximum (if using the `dataMaximum` function on the scale), or a specified value (if using the `max` function on the scale). For example, a graphic element may be sized by a continuous value. By default, the continuous scale has a "nice value" maximum. The `aestheticMaximum` function can map a size to this maximum value.

Examples

```
SCALE: linear(aesthetic(aesthetic.size), aestheticMinimum(size."1px"),
              aestheticMaximum(size."5px"))
ELEMENT: point(position(x*y), size(z))
```

Figure 88. Example: Specifying a minimum and maximum size for points in a bubble plot

Applies To

“asn Scale ” on page 28
“atanh Scale ” on page 28
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“logit Scale ” on page 33
“pow Scale ” on page 33
“prob Scale ” on page 34
“probit Scale ” on page 35
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

aestheticMinimum Function

Syntax

```
aestheticMinimum(<aesthetic type>.<aesthetic constant>)
```

or

```
aestheticMinimum(<aesthetic type>."<aesthetic value>")
```

<aesthetic type>. An aesthetic type indicating the specific aesthetic for which a minimum value is being specified. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the ELEMENT statement.

<aesthetic constant>. A constant for the aesthetic (for example, `size.tiny`). Valid constants depend on the aesthetic.

"aesthetic value". A specific value for the aesthetic (for example, `size."1px"`. Valid values depend on the aesthetic.

Description

Specifies an aesthetic value that is mapped to the minimum data value for a scale. The minimum data value may be a "nice value" based on the data (the default), the exact data minimum (if using the `dataMinimum` function on the scale), or a specified value (if using the `min` function on the scale). For example, a graphic element may be sized by a continuous value. By default, the continuous scale has a "nice value" minimum. The `aestheticMinimum` function can map a size to this minimum value.

Examples

```
SCALE: linear(aesthetic(aesthetic.size), aestheticMinimum(size."1px"),
              aestheticMaximum(size."5px"))
ELEMENT: point(position(x*y), size(z))
```

Figure 89. Example: Specifying a minimum and maximum size for points in a bubble plot

Applies To

“asn Scale ” on page 28
“atanh Scale ” on page 28
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“logit Scale ” on page 33
“pow Scale ” on page 33
“prob Scale ” on page 34
“probit Scale ” on page 35
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

aestheticMissing Function

Syntax

```
aestheticMissing(<aesthetic type>.<aesthetic constant>)
```

or

```
aestheticMissing(<aesthetic type>."<aesthetic value>")
```

<aesthetic type>. An aesthetic type indicating the specific aesthetic for which a missing value aesthetic is being specified. This is an aesthetic created as the result of an aesthetic function (such as `size`) in the `ELEMENT` statement.

<aesthetic constant>. A constant for the aesthetic (for example, `size.tiny`). Valid constants depend on the aesthetic.

"aesthetic value". A specific value for the aesthetic (for example, `size."1px"`. Valid values depend on the aesthetic.

Description

Specifies an aesthetic value that is mapped to missing values for a scale.

Examples

```
SCALE: linear(aesthetic(aesthetic.color), aestheticMissing(color.black))
ELEMENT: point(position(x*y), color(z))
```

Figure 90. Example: Specifying a missing value aesthetic for a scale

Applies To

“asn Scale ” on page 28
“atanh Scale ” on page 28

“cat Scale ” on page 29
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“logit Scale ” on page 33
“pow Scale ” on page 33
“prob Scale ” on page 34
“probit Scale ” on page 35
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

alpha Function

Syntax

```
alpha(<numeric>)
```

<numeric>. A numeric value between 0 and 1.

Description

Specifies a percentage value used to calculate a percentile value or confidence interval.

Examples

```
ELEMENT: interval(position(region.confi.mean(jobcat*salary, alpha(0.99))))
```

Figure 91. Example: Specifying a 99% confidence interval

Applies To

“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“summary.percentile Function ” on page 264

base Function

Syntax

```
base(<numeric>)
```

<numeric>. A numeric value indicating the base for the logarithmic scale.

Description

Specifies a base for the logarithmic scale.

Examples

```
SCALE: log(dim(2), base(2))
```

Figure 92. Example: Specifying a different base for a logarithmic scale

Applies To

“log Scale ” on page 32
“safeLog Scale ” on page 35

base.aesthetic Function

Syntax

```
base.aesthetic(aesthetic(aesthetic.<aesthetic type>))
```

<aesthetic type>. An aesthetic whose associated variable is used as the percentage base.

Description

Specifies that the percentage is based on the count across the result of an aesthetic function. Summing the percentages of all of the cases in a specific aesthetic group equals 100%. For example, all blue bars sum to 100%, and all red bars sum to 100%.

Examples

```
COORD: rect(dim(1,2), cluster(3))
ELEMENT: interval(position(summary.percent(summary.count(jobcat*1*gender),
                  base.aesthetic(aesthetic(aesthetic.color)))), color(jobcat))
```

Figure 93. Example: Using a variable across clusters as the percentage base

Applies To

“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277

base.all Function

Syntax

```
base.all()
```

or

```
base.all(acrossPanels())
```

Description

Specifies that the percentage is based on the total count. Summing the percentages of all of the graphic elements in the chart or in each panel equals 100%. If you are using paneling and want to specify the total count across all panels as the percentage base, use the `acrossPanels` function.

Examples

```
COORD: rect(dim(1,2))
ELEMENT: interval(position(summary.percent(summary.count(jobcat),
                  base.all())))
```

Figure 94. Example: Specifying the total count as the percentage base

```
COORD: rect(dim(1,2))
ELEMENT: interval(position(summary.percent(summary.count(jobcat*1*gender),
                 base.all())))
```

Figure 95. Example: Specifying the total count in each panel as the percentage base

```
COORD: rect(dim(1,2))
ELEMENT: interval(position(summary.percent(summary.count(jobcat*1*gender),
                 base.all(acrossPanels()))))
```

Figure 96. Example: Specifying the total count across all panels as the percentage base

Applies To

“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277

base.coordinate Function

Syntax

```
base.coordinate(dim(<numeric>))
```

<numeric>. A numeric value indicating the dimension to which the scale applies. See the topic “dim Function ” on page 115 for more information.

Description

Specifies that the percentage is based on the individual values along a specific dimension. Summing the percentages of all of the graphic elements with a particular value on the specified dimension equals 100%. For example, you may do this to specify that the segments in each stacked bar sum to 100%.

Examples

```
ELEMENT: interval.stack(position(summary.percent(summary.count(jobcat*1*gender),
         base.coordinate(dim(1)))), color(gender))
```

Figure 97. Example: Making each stack equal 100%

```
COORD: rect(dim(1,2), cluster(3))
ELEMENT: interval(position(summary.percent(summary.count(gender*1*jobcat),
         base.coordinate(dim(3)))), color(gender))
```

Figure 98. Example: Making each cluster equal 100%

Applies To

“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254

“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277

begin Function (For GPL Graphs)

Syntax

```
begin(<function>)
```

<function>. One or more valid functions. These are optional.

Specifies the start of the GPL block that defines a particular graph.

Examples

```
GRAPH: begin(scale(50%,50%))
```

Figure 99. Example: Scaling a graph by 50%

```
GRAPH: begin()
ELEMENT: line(position(x*y))
GRAPH: end()
```

Figure 100. Example: Defining a particular graph

Valid Functions

“origin Function (For GPL Graphs)” on page 179
“scale Function (For GPL Graphs)” on page 203

Applies To

“GRAPH Statement ” on page 20

begin Function (For GPL Pages)

Syntax

```
begin(<function>)
```

<function>. One or more valid functions. These are optional.

Specifies the start of the GPL block that defines the page display or visualization.

Examples

```
PAGE: begin(scale(50%,50%))
```

Figure 101. Example: Scaling a visualization

Valid Functions

“scale Function (For GPL Pages)” on page 204

Applies To

“PAGE Statement ” on page 20

beta Function

Syntax

```
beta(<shape>, <shape>)
```

<shape>. Numeric values specifying the shape parameters for the distribution. Values must be greater than 0.

Description

Specifies a beta distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), beta(2, 5))
```

Figure 102. Example: Specifying a beta distribution for the probability scale

Applies To

"prob Scale " on page 34

bin.dot Function

Syntax

```
bin.dot.<position>(<algebra>, dim(<numeric>), <function>)
```

or

```
bin.dot.<position>(<binning function>, dim(<numeric>), <function>)
```

or

```
bin.dot.<position>(<statistic function>, dim(<numeric>), <function>)
```

<position>. The position at which a graphic element representing the bins is drawn. `center` is the graphical middle of the bin and makes it less likely that the graphic elements will overlap. `centroid` positions the graphic element at the centroid location of the values it represents. The coordinates of the centroid are the weighted means for each dimension. Specifying the position is optional. If none is specified, `center` is used.

<algebra>. Graph algebra, such as `x*y`. Refer to "Brief Overview of GPL Algebra" on page 3 for an introduction to graph algebra.

<numeric>. One or more numeric values (separated by commas) indicating the graph dimension or dimensions in which to bin the data. Using `dim()` is optional. Specifying the dimensions is necessary only when you want to bin a specific non-default dimension or multiple dimensions, for example when binning a 2-D scatterplot. See the topic "dim Function " on page 115 for more information.

<function>. One or more valid functions. These are optional.

<binning function>. A binning function.

<statistic function>. A statistic function.

Description

Creates irregularly spaced bins of graphic elements that have nearly identical values. When data are sparse, `bin.dot` centers the bins on the data. This function is typically used to create a dot plot.

Examples

```
ELEMENT: point.stack.asymmetric(position(bin.dot(salary)))
```

Figure 103. Example: Creating a 1-D dot plot

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions

“binCount Function ” on page 75
“binStart Function ” on page 75
“binWidth Function ” on page 76

Binning Functions

“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70

Applies To

“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“density.beta Function ” on page 88
“density.chiSquare Function ” on page 90
“density.exponential Function ” on page 92
“density.f Function ” on page 94
“density.gamma Function ” on page 96
“density.logistic Function ” on page 101
“density.normal Function ” on page 103
“density.poisson Function ” on page 105
“density.studentizedRange Function ” on page 107
“density.t Function ” on page 109
“density.uniform Function ” on page 111
“density.weibull Function ” on page 113
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244

“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296

bin.hex Function

Syntax

```
bin.hex.<position>(<algebra>, dim(<numeric>), <function>)
```

or

```
bin.hex.<position>(<binning function>, dim(<numeric>), <function>)
```

or

```
bin.hex.<position>(<statistic function>, dim(<numeric>), <function>)
```

<position>. The position at which a graphic element representing the bins is drawn. `center` is the graphical middle of the bin and makes it less likely that the graphic elements will overlap. `centroid` positions the graphic element at the centroid location of the values it represents. The coordinates of the centroid are the weighted means for each dimension. Specifying the position is optional. If none is specified, `center` is used.

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<numeric>. One or more numeric values (separated by commas) indicating the graph dimension or dimensions in which to bin the data. Using `dim()` is optional.Specifying the dimensions is necessary only when you want to bin a specific non-default dimension or multiple dimensions, for example when binning a 2-D scatterplot. See the topic “dim Function ” on page 115 for more information.

<function>. One or more valid functions. These are optional.

<binning function>. A binning function.

<statistic function>. A statistic function.

Description

Creates hexagonal bins for grouping similar cases. `bin.hex` is most often used when creating binned scatterplots or other binned multivariate graphs.

Examples

```
ELEMENT: point(position(bin.hex(salbegin*salary, dim(1,2))), size(summary.count()))
```

Figure 104. Example: Binned scatterplot

```
ELEMENT: polygon(position(bin.hex(salbegin*salary, dim(1,2))), color(summary.count()))
```

Figure 105. Example: Binned polygon

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions

“binCount Function ” on page 75
“binStart Function ” on page 75
“binWidth Function ” on page 76

Binning Functions

“bin.dot Function ” on page 66
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“density.beta Function ” on page 88
“density.chiSquare Function ” on page 90
“density.exponential Function ” on page 92
“density.f Function ” on page 94
“density.gamma Function ” on page 96
“density.logistic Function ” on page 101
“density.normal Function ” on page 103
“density.poisson Function ” on page 105
“density.studentizedRange Function ” on page 107
“density.t Function ” on page 109
“density.uniform Function ” on page 111
“density.weibull Function ” on page 113
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162

“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296

bin.quantile.letter Function

Syntax

```
bin.quantile.letter(<algebra>)
```

or

```
bin.quantile.letter(<binning function>)
```

or

```
bin.quantile.letter(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. A statistic function.

Description

Calculates the five statistics (minimum, first quartile, median, third quartile, and maximum) used in box plots. The data are binned appropriately as a result of the statistical calculation. `bin.quantile.letter` is used with the schema element to generate a box plot.

Examples

```
ELEMENT: schema(position(bin.quantile.letter(jobcat*salary)))
```

Figure 106. Example: Creating a box plot

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.rect Function ” on page 72
“density.beta Function ” on page 88
“density.chiSquare Function ” on page 90
“density.exponential Function ” on page 92
“density.f Function ” on page 94
“density.gamma Function ” on page 96
“density.logistic Function ” on page 101
“density.normal Function ” on page 103
“density.poisson Function ” on page 105
“density.studentizedRange Function ” on page 107
“density.t Function ” on page 109
“density.uniform Function ” on page 111
“density.weibull Function ” on page 113
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164

“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296

bin.rect Function

Syntax

```
bin.rect.<position>(<algebra>, dim(<numeric>), <function>)
```

or

```
bin.rect.<position>(<binning function>, dim(<numeric>), <function>)
```

or

```
bin.rect.<position>(<statistic function>, dim(<numeric>), <function>)
```

<position>. The position at which a graphic element representing the bins is drawn. `center` is the graphical middle of the bin and makes it less likely that the graphic elements will overlap. `centroid` positions the graphic element at the centroid location of the values it represents. The coordinates of the centroid are the weighted means for each dimension. Specifying the position is optional. If none is specified, `center` is used.

<algebra>. Graph algebra, such as `x*y`. Refer to "Brief Overview of GPL Algebra" on page 3 for an introduction to graph algebra.

<numeric>. One or more numeric values (separated by commas) indicating the graph dimension or dimensions in which to bin the data. Using `dim()` is optional. Specifying the dimensions is necessary only when you want to bin a specific non-default dimension or multiple dimensions--for example, when binning a 2-D scatterplot. See the topic "dim Function " on page 115 for more information.

<function>. One or more valid functions. These are optional.

<binning function>. A binning function.

<statistic function>. A statistic function.

Description

Creates rectangular bins for grouping similar cases. `bin.rect` is the binning method commonly used in histograms to calculate the count in each bin.

Examples

```
ELEMENT: interval(position(summary.count(bin.rect(salary))))
```

Figure 107. Example: Histogram binning

```
ELEMENT: interval(position(summary.count(bin.rect(salary, binWidth(5000)))))
```

Figure 108. Example: Histogram binning with specified bin sizes

```
ELEMENT: point(position(summary.count(bin.rect(salbegin*salary, dim(1,2)))),
               size(summary.count()))
```

Figure 109. Example: Binned scatterplot

Statistic Functions

See "GPL Functions" on page 53 .

Valid Functions

"binCount Function " on page 75
"binStart Function " on page 75
"binWidth Function " on page 76

Binning Functions

"bin.dot Function " on page 66
"bin.hex Function " on page 68
"bin.quantile.letter Function " on page 70

Applies To

"bin.dot Function " on page 66
"bin.hex Function " on page 68
"bin.quantile.letter Function " on page 70
"density.beta Function " on page 88
"density.chiSquare Function " on page 90

“density.exponential Function ” on page 92
“density.f Function ” on page 94
“density.gamma Function ” on page 96
“density.logistic Function ” on page 101
“density.normal Function ” on page 103
“density.poisson Function ” on page 105
“density.studentizedRange Function ” on page 107
“density.t Function ” on page 109
“density.uniform Function ” on page 111
“density.weibull Function ” on page 113
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275

“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296

binCount Function

Syntax

```
binCount(<integer> ...)
```

<integer>. An integer indicating the number of bins. If there are multiple binned dimensions, you can specify the number of bins for each dimension. Use commas to separate the multiple counts. For example, `binCount(15,10)` specifies 15 bins for dimension 1 and 10 for dimension 2. `0` specifies the default for a dimension. So, `binCount(0,10)` specifies the default number of bins for dimension 1 and 10 bins for dimension 2.

Description

Specifies the number of bins.

Examples

```
ELEMENT: interval(position(summary.count(bin.rect(salary, binCount(25)))))
```

Figure 110. Example: Defining a specific number of bins

```
ELEMENT: interval(position(bin.rect(salbegin*salary, dim(1,2), binCount(25,25))))
```

Figure 111. Example: Defining a specific number of bins for multiple binned dimensions

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.rect Function ” on page 72

binStart Function

Syntax

```
binStart(<value> ...)
```

<value>. An integer or quoted date literal indicating the value of the first bin. If there are multiple binned dimensions, you can specify the first bin for each dimension. Use commas to separate the multiple first bins. For example, `binStart(0,10)` specifies 0 as the first bin on dimension 1 and 10 as the first bin on dimension 2. If you specify a value for one dimension, you have to specify a value for all dimensions.

Description

Specifies the value of the first bin. You can use the function to make sure bins begin at a specified value, regardless of the data values. Note that the first bin may not be drawn if there are no actual data values in that bin. However, the bin is still included when determining the number of bins and their widths.

If the specified value is greater than the lowest data value on the dimension, this function does not really specify the starting value of the first bin because the function can never exclude smaller values from the bins. Rather, the function specifies the starting value for some other bin, depending on the width or number of bins. There may be one or more bins that precede the specified value.

Examples

```
ELEMENT: interval(position(summary.count(bin.rect(salary, binStart(10000)))))
```

Figure 112. Example: Specifying the first bin on a continuous scale

```
ELEMENT: interval(position(summary.count(bin.rect(saledate, binStart("01/20/2003")))))
```

Figure 113. Example: Specifying the first bin on a date scale

```
ELEMENT: interval(position(bin.rect(x*y, dim(1,2), binStart(2000,1000))))
```

Figure 114. Example: Specifying the first bin for multiple binned dimensions

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.rect Function ” on page 72

binWidth Function

Syntax

```
binWidth(<numeric> ...)
```

<numeric>. A positive numeric value indicating the width of the bins. If the data being binned are dates, the value indicates days or seconds, depending on the underlying data. To specify an explicit unit, append d or s to the value to indicate days or seconds, respectively. If there are multiple binned dimensions, you can specify the bin width for each dimension. Use commas to separate the multiple widths. For example, `binWidth(100,200)` specifies 100 as the bin width for dimension 1 and 200 as the bin width for dimension 2. `0` specifies the default for a dimension. So, `binWidth(0,200)` specifies the default bin width for dimension 1 and 200 as the bin width for dimension 2.

Description

Specifies the width of the bins.

Examples

```
ELEMENT: interval(position(summary.count(bin.rect(salary, binWidth(1000)))))
```

Figure 115. Example: Defining a specific bin width

```
ELEMENT: interval(position(bin.rect(salbegin*salary, dim(1,2), binWidth(1000,2000))))
```

Figure 116. Example: Defining a specific bin width for multiple binned dimensions

```
ELEMENT: interval(position(summary.count(bin.rect(date, binWidth(30d)))))
```

Figure 117. Example: Defining a specific bin width for date data

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.rect Function ” on page 72

chiSquare Function

Syntax

```
chiSquare(<degrees of freedom>)
```

<degrees of freedom>. Numeric value specifying the degrees of freedom parameter for the distribution. This value must be greater than 0.

Description

Specifies a chi-square distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), chiSquare(5))
```

Applies To

“prob Scale ” on page 34

closed Function

Syntax

```
closed()
```

Description

Specifies that the end point of a graphic element is connected to its start point. In polar coordinates, this results in a closed loop around the center of the coordinate system.

Examples

```
ELEMENT: line(position(x*y), closed())
```

Figure 118. Example: Creating a closed line

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“line Element ” on page 47
“path Element ” on page 48

cluster Function

Syntax

```
cluster(<integer> ...)
```

<integer>. One or more integers indicating the variable or variables in the algebra along whose axis the clustering occurs.

Description

Clusters graphic elements along a specific axis. You can also cluster graphic elements on more than one axis in 3-D charts.

Examples

```
COORD: rect(dim(1,2), cluster(3))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)), color(jobcat))
```

Figure 119. Example: Clustering on the first dimension in a 2-D coordinate system

In this example, *jobcat* is clustered in *gender*. Compare the position of the numbers in `dim()` to the positions of the numbers in `cluster()`. In this case, the 1 in `dim` and 3 in `cluster` are the first numbers in their respective functions. Therefore, clustering occurs on dim(3) (*gender*), and dim(1) (*jobcat*) specifies the variable that defines the graphic elements in each cluster. If you removed the `cluster` function, the chart would look similar, but dim(3) would specify a paneling facet and dim(1) would be the *x* axis. The clustering collapses multiple panels into one, changing the way dimensions are displayed. For example, compare the following graphs.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
DATA: salary=col(source(s), name("salary"))
COORD: rect(dim(1,2), cluster(3))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(3), label("Gender"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)), color(jobcat))
```

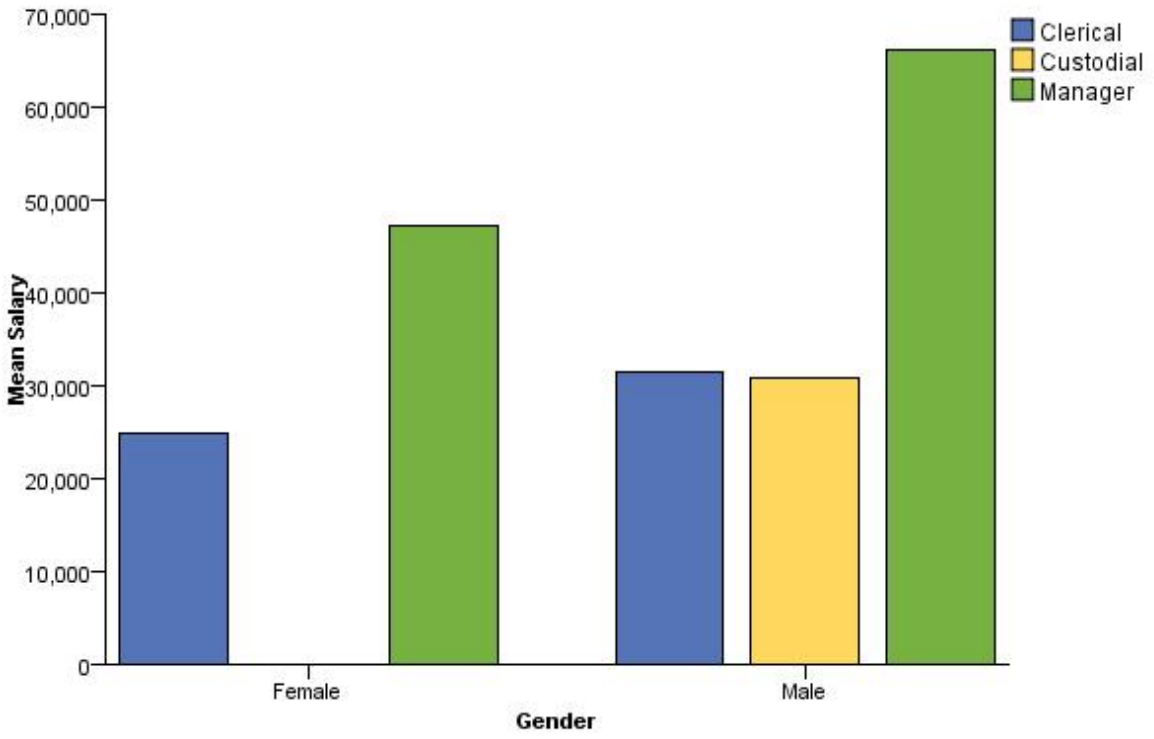


Figure 120. Clustered bar chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: salary = col(source(s), name("salary"))
SCALE: linear(dim(2), include(0.0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)))
```

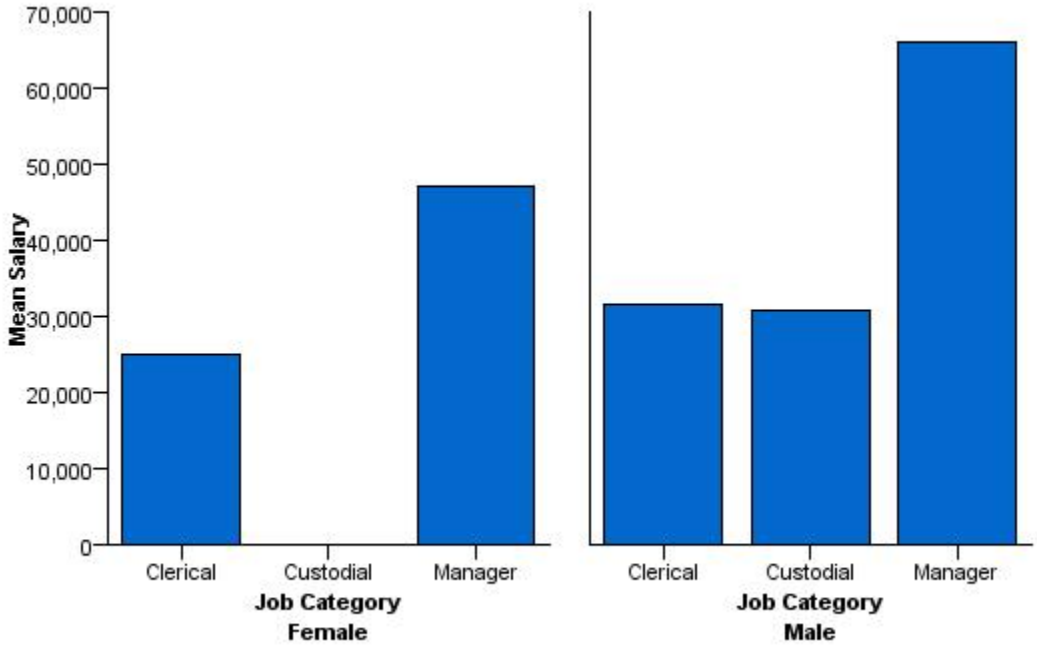


Figure 121. Faceted bar chart

```
COORD: rect(dim(1,2,3), cluster(4,0,0))
ELEMENT: interval(position(summary.mean(jobcat*gender*salary*minority)),
                  color(jobcat))
```

Figure 122. Example: Clustering on the first dimension in a 3-D coordinate system

In this example, *jobcat* is clustered in *minority*. The first parameter of `cluster` is 4. This indicates that the first variable (*jobcat*) in the algebra is clustered in the fourth variable (*minority*). As in the 2-D example, removing the `cluster` function would result in a paneled chart.

```
COORD: rect(dim(1,2,3), cluster(0,4,0))
ELEMENT: interval(position(summary.mean(minority*jobcat*salary*gender)),
                  color(gender))
```

Figure 123. Example: Clustering on the second dimension in a 3-D coordinate system

In this example, *jobcat* is clustered in *gender*. The first parameter of `cluster` is 0, which indicates that dim(1) (*minority*) is not affected by the clustering. The second parameter of `cluster` is 4. This indicates that the second variable (*jobcat*) in the algebra is clustered in the fourth variable (*gender*).

Applies To

“rect Coordinate Type ” on page 26

col Function

Syntax

```
col(source(<source name>), name("variable name"), <type>, <function>)
```

<source name>. A data source previously defined by a SOURCE statement.

"variable name". The name of the variable in the data source.

<type>. A data type or measurement level. If no type is specified, the variable is assumed to be continuous.

<function>. One or more valid functions. These are optional.

Description

Extracts a column of data from a data source. This is used for creating variables from the data.

Examples

```
DATA: age = col(source(mydata), name("age"))
```

Figure 124. Example: Specifying a continuous variable from a data source

```
DATA: gender = col(source(mydata), name("gender"), unit.category())
```

Figure 125. Example: Specifying a categorical variable from a data source

```
DATA: date = col(source(mydata), name("date"), unit.time(), format("M/d/yyyy"))
```

Figure 126. Example: Specifying a date variable from a data source

Valid Types

unit.category. Specifies a categorical variable.

unit.time. Specifies a date variable. You will often need to specify the date format if using this type. *Note*: In IBM SPSS Statistics, dates from a `userSource` are passed to GPL as numeric values and are specified as continuous variables. Therefore, you should not use this type with a `userSource`.See the topic “format Function ” on page 124 for more information.

Valid Functions

“in Function ” on page 127
“format Function ” on page 124
“notIn Function ” on page 178

Applies To

“DATA Statement ” on page 21

collapse Function

Syntax

```
collapse(category(<algebra>), minimumPercent(<numeric>), sumVariable(<algebra>),
         otherValue("label"))
```

<algebra>. Graph algebra, although in this case, the algebra should identify only one variable. The variable in `category()` identifies the variable whose categories are collapsed. The variable in `sumVariable()` identifies the variable whose sum for the total compared to the sum for a particular category determines whether the category is collapsed.

<numeric>. A numeric value between 0 and 100 indicating a percentage. A category is collapsed if its sum is less than the specified percentage of the total sum for `sumVariable`.

"label". The label for the new variable containing the collapsed categories. This is the text that identifies the variable in the graph.

Description

Collapse small categories of a categorical variable to create a new categorical variable. The function collapses the categories by recoding them to the value specified by `otherValue`.

Examples

```
TRANS: educ_collapse = collapse(category(educ), minimumPercent(10), sumVariable(salary),
                                otherValue("Other"))
ELEMENT: interval(position(summary.sum(educ_collapse*salary)))
```

Figure 127. Example: Collapsing categories whose sum is less than 10% of total

Applies To

"TRANS Statement " on page 22

color Function (For GPL Graphic Elements)

Note: If you are modifying the color for a guide, refer to "color Function (For GPL Guides)" on page 82 .

Syntax

```
color(<algebra>)
```

or

```
color(color.<color constant>)
```

or

```
color(color."color value")
```

or

```
color(<statistic function>)
```

<algebra>. Graph algebra, using one variable or a blend of variables. Each unique variable value results in a different color. For example, if you were creating a stacked bar chart, the argument of the color function would be the variable that controls the stacking. Each stack segment would be a different color.

<color constant>. A constant indicating a specific color, such as `red`. See the topic "Color Constants " on page 385 for more information. Color constants can also be blended (for example, `color.blue +color.red`).

"color value". A specific value that indicates the hexadecimal RGB color components (for example, `color."6666FF"`).

<statistic function>. A statistic function.

Description

Controls the color of the graphic elements. To specify the color explicitly for the fill or border of the graphic element, you can append `.interior` or `.exterior` to the function. Using `color` without a qualifier implies `color.interior`.

Examples

```
ELEMENT: line(position(x*y), color(color.red))
```

Figure 128. Example: Specifying a color value with a constant

```
ELEMENT: line(position(x*y), color(color."FF0000"))
```

Figure 129. Example: Specifying a color value with RGB color components

```
ELEMENT: interval(position(x*y), color.exterior(color.red))
```

Figure 130. Example: Specifying a color value for the bar border

```
ELEMENT: point(position(x*y), color(z))
```

Figure 131. Example: Using the values of a variable to control color

```
ELEMENT: interval(position(summary.mean(jobcat*salary)),
         color(summary.count()))
```

Figure 132. Example: Using a statistical function to control color

Statistic Functions

See “GPL Functions” on page 53 .

Applies To
“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

color Function (For GPL Guides)

Note: If you are modifying the color for a graphic element (like a bar or point), refer to “color Function (For GPL Graphic Elements)” on page 81 .

Syntax

```
color(color.<color constant>)
```

or

```
color(color."color value")
```

<color constant>. A constant indicating a specific color, such as `red`. See the topic “Color Constants ” on page 385 for more information.

"color value". A specific value that indicates the hexadecimal RGB color components (for example, `color."6666FF"`).

Description

Controls the color of guides, such as axes and reference lines.

Examples

```
GUIDE: axis(dim(2), color(color.blue))
```

Figure 133. Example: Specifying an axis color

Applies To
“axis Guide Type ” on page 39
“form.line Guide Type ” on page 40

color.brightness Function (For GPL Graphic Elements)

Note: If you are modifying the brightness for a guide, refer to “color.brightness Function (For GPL Guides)” on page 83 .

Syntax

```
color.brightness(<algebra>)
```

or

```
color.brightness(color.brightness."brightness value")
```

or

```
color.brightness(<statistic function>)
```

<algebra>. Graph algebra, using one variable or a blend of variables. Each variable value results in a different level of color brightness.

"brightness value". A value between 0 and 1 that indicates the level of brightness. A value of 1 indicates full brightness, while a value of 0 indicates no brightness (black).

<statistic function>. A statistic function.

Description

Controls the color brightness of the graphic elements. To specify the color brightness explicitly for the fill or border of the graphic element, you can append `.interior` or `.exterior` to the function. Using `color.brightness` without a qualifier implies `color.brightness.interior`.

Examples

```
ELEMENT: point(position(x*y), color.brightness(z), color(color.blue))
```

Figure 134. Example: Using the values of a variable to control color brightness

```
ELEMENT: interval(position(summary.mean(jobcat*salary)),
         color.brightness(summary.count()))
```

Figure 135. Example: Using a statistical function to control color brightness

Statistic Functions

See “GPL Functions” on page 53 .

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

color.brightness Function (For GPL Guides)

Note: If you are modifying the brightness for a graphic element (like a bar or point), refer to “color.brightness Function (For GPL Graphic Elements)” on page 82 .

Syntax

```
color.brightness(color.brightness."brightness value")
```

"brightness value". A value between 0 and 1 that indicates the level of brightness. A value of 1 indicates full brightness, while a value of 0 indicates no brightness (black).

Description

Controls the brightness of reference lines.

Examples

```
GUIDE: form.line(position(*,2000), color.brightness(color.brightness."0.5"))
```

Figure 136. Example: Specifying a reference line brightness

Applies To

“form.line Guide Type ” on page 40

color.hue Function (For GPL Graphic Elements)

Note: If you are modifying the hue for a guide, refer to “color.hue Function (For GPL Guides)” on page 84 .

Syntax

```
color.hue(<algebra>)
```

or

```
color.hue(color.hue."hue value")
```

or

```
color.hue(<statistic function>)
```

<algebra>. Graph algebra, using one variable or a blend of variables. Each variable value results in a different color hue.

"hue value". A value between 0 and 1 that indicates the hue level.

<statistic function>. A statistic function.

Description

Controls the color hue of the graphic elements. To specify the color hue explicitly for the fill or border of the graphic element, you can append `.interior` or `.exterior` to the function. Using `color.hue` without a qualifier implies `color.hue.interior`.

`color.hue` requires a base color other than white or black. Use `color.interior` or `color.exterior` to set the base color.

Examples

```
ELEMENT: point(position(x*y), color.hue(z), color(color.blue))
```

Figure 137. Example: Using the values of a variable to control color hue

```
ELEMENT: interval(position(summary.mean(jobcat*salary)),
         color.hue(summary.count()))
```

Figure 138. Example: Using a statistical function to control color hue

Statistic Functions

See “GPL Functions” on page 53 .

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

color.hue Function (For GPL Guides)

Note: If you are modifying the hue for a graphic element (like a bar or point), refer to “color.hue Function (For GPL Graphic Elements)” on page 84 .

Syntax

```
color.hue(color.brightness."hue value")
```

"hue value". A value between 0 and 1 that indicates the hue level.

Description

Controls the hue of reference lines.

Examples

```
GUIDE: form.line(position(*,2000), color.hue(color.hue."0.5"))
```

Figure 139. Example: Specifying a reference line hue

Applies To
“form.line Guide Type ” on page 40

color.saturation Function (For GPL Graphic Elements)

Note: If you are modifying the saturation for a guide, refer to “color.saturation Function (For GPL Guides)” on page 86 .

Syntax

```
color.saturation(<algebra>)
```

or

```
color.saturation(color.saturation."saturation value")
```

or

```
color.saturation(<statistic function>)
```

<algebra>. Graph algebra, using one variable or a blend of variables. Each variable value results in a different level of color saturation.

"saturation value". A value between 0 and 1 that indicates the saturation level. A value of 1 indicates full saturation, while a value of 0 indicates no saturation (gray).

<statistic function>. A statistic function.

Description

Controls the color saturation of the graphic elements. To specify the color saturation explicitly for the fill or border of the graphic element, you can append `.interior` or `.exterior` to the function. Using `color.saturation` without a qualifier implies `color.saturation.interior`.

Examples

```
ELEMENT: point(position(x*y), color.saturation(z), color(color.blue))
```

Figure 140. Example: Using the values of a variable to control color saturation

```
ELEMENT: interval(position(summary.mean(jobcat*salary)),
         color.saturation(summary.count()))
```

Figure 141. Example: Using a statistical function to control color

Statistic Functions

See “GPL Functions” on page 53 .

Applies To
“area Element ” on page 44

“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

color.saturation Function (For GPL Guides)

Note: If you are modifying the saturation for a graphic element (like a bar or point), refer to “color.saturation Function (For GPL Graphic Elements)” on page 85 .

Syntax

```
color.saturation(color.saturation."saturation value")
```

"saturation value". A value between 0 and 1 that indicates the saturation level. A value of 1 indicates full saturation, while a value of 0 indicates no saturation (gray).

Description

Controls the saturation of reference lines.

Examples

```
GUIDE: form.line(position(*,2000), color.saturation(color.saturation."0.5"))
```

Figure 142. Example: Specifying a reference line saturation

Applies To

“form.line Guide Type ” on page 40

csvSource Function

Syntax

```
csvSource(file("file path"), key("key name"), <function>)
```

"file path". The path to the CSV file. This can be an absolute or relative path. The path is relative to the location of the application that parses the GPL code. Backslashes must be escaped with another backslash. You can also use forward slashes.

"key name". The name of a variable in the file that acts as a key. The key is used to link multiple sources, especially a dataset and a map file.

<function>. One or more valid functions. These are optional.

Description

Reads the contents of a comma-separated values (CSV) file. This function is used to assign the contents of the file to a data source.

Examples

```
SOURCE: mydata = csvSource(file("/Data/Employee data.csv"))
```

Figure 143. Example: Reading a CSV file

Valid Functions

“missing.listwise Function ” on page 175
“missing.pairwise Function ” on page 176
“weight Function ” on page 305

Applies To
“SOURCE Statement ” on page 21

dataMaximum Function

Syntax

```
dataMaximum()
```

Description

Specifies that the maximum of the scale is exactly the same as the maximum of the data. Otherwise, a "nice" maximum is used. For example, if the last data value is 97, a "nice" maximum for the scale may be 100. `dataMaximum` forces the scale to end at 97.

Examples

```
SCALE: linear(dim(2), dataMaximum())
```

Figure 144. Example: Specifying a maximum on the 2nd dimension (y axis)

Applies To
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“pow Scale ” on page 33
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

dataMinimum Function

Syntax

```
dataMinimum()
```

Description

Specifies that minimum of the scale is exactly the same as the minimum of the data. Otherwise, a "nice" minimum is used. For example, if the first data value is 2, a "nice" minimum for the scale may be 0. `dataMinimum` forces the scale to begin at 2.

Examples

```
SCALE: linear(dim(2), dataMinimum())
```

Figure 145. Example: Specifying a minimum on the 2nd dimension (y axis)

Applies To
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“pow Scale ” on page 33
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

delta Function

Syntax

```
delta(<numeric>)
```

<numeric>. A positive numeric value indicating the difference between major ticks on the axis. If the underlying data along the axis are dates, the value indicates days.

Description

Specifies the difference between major ticks on an axis. Major ticks are the location at which labels are displayed along the axis.

Examples

```
GUIDE: axis(dim(1), delta(1000))
```

Figure 146. Example: Specifying the distance between major ticks

Applies To

“axis Guide Type ” on page 39

density.beta Function

Syntax

```
density.beta(<algebra>, shape1(<numeric>), shape2(<numeric>), <function>)
```

or

```
density.beta(<binning function>, shape1(<numeric>), shape2(<numeric>, <function>))
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<numeric>. `shape1` and `shape2` define the parameters for the distribution. These take numeric values and are **required**.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density for the beta distribution. This is often used to add a distribution curve. The distribution is defined on the closed interval [0, 1]. If you don't see the graphic element for the distribution, check the parameters for the distribution and the range for the *x* axis scale.

Because this function does not estimate parameters from the data, it can be used only for comparison and not for fitting.

Examples

```
ELEMENT: line(position(density.beta(x, shape1(2), shape2(5))))
```

Figure 147. Example: Adding a beta distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256

“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.chiSquare Function

Syntax

```
density.chiSquare(<algebra>, degreesOfFreedom(<integer>), <function>)
```

or

```
density.chiSquare(<binning function>, degreesOfFreedom(<integer>), <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<integer>. `degreesOfFreedom` defines the parameter for the distribution. This takes a positive integer and is **required**.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the chi-square distribution. This is often used to add a distribution curve. If you don't see the graphic element for the distribution, check the parameter for the distribution and the range for the *x* axis scale.

Because this function does not estimate parameters from the data, it can be used only for comparison and not for fitting.

Examples

```
ELEMENT: line(position(density.chiSquare(x, degreesoffreedom(5))))
```

Figure 148. Example: Adding a chi-square distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246

“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.exponential Function

Syntax

```
density.exponential(<algebra>, rate(<numeric>), <function>)
```

or

```
density.exponential(<binning function>, rate(<numeric>), <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<numeric>. `rate` defines the parameter for the distribution. This takes a numeric value greater than or equal to 0 and is optional. If the parameter is not specified, it is calculated from the underlying data.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the exponential distribution. This is often used to add a distribution curve. If you don't see the graphic element for the distribution, check the parameter for the distribution and the range for the *x* axis scale.

Examples

```
ELEMENT: line(position(density.exponential(x, rate(1.5))))
```

Figure 149. Example: Adding a chi-square distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240

“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.f Function

Syntax

```
density.f(<algebra>, degreesOfFreedom1(<integer>), degreesOfFreedom2(<integer>), <function>)
```

or

```
density.f(<binning function>, degreesOfFreedom1(<integer>), degreesOfFreedom2(<integer>), <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<integer>. `degreesOfFreedom1` and `degreesOfFreedom2` define the parameters for the distribution. These take positive integers and are **required**.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the F distribution. This is often used to add a distribution curve. If you don't see the graphic element for the distribution, check the parameters for the distribution and the range for the *x* axis scale.

Because this function does not estimate parameters from the data, it can be used only for comparison and not for fitting.

Examples

```
ELEMENT: line(position(density.f(x, degreesoffreedom1(5), degreesoffreedom2(2))))
```

Figure 150. Example: Adding an F distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234

“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.gamma Function

Syntax

```
density.gamma(<algebra>, rate(<numeric>), <function>)
```

or

```
density.gamma(<binning function>, rate(<numeric>), <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<numeric>. `rate` defines the parameter for the distribution. This takes a positive numeric value and is **required**.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the gamma distribution. This is often used to add a distribution curve. If you don't see the graphic element for the distribution, check the parameters for the distribution and the range for the *x* axis scale.

Because this function does not estimate parameters from the data, it can be used only for comparison and not for fitting.

Examples

```
ELEMENT: line(position(density.gamma(x, rate(2.5))))
```

Figure 151. Example: Adding a gamma distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227

“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.kernel Function

Syntax

```
density.kernel.<kernel function>(<algebra>, fixedWindow(<numeric>), <function>)
```

or

```
density.kernel.<kernel function>(<algebra>, nearestNeighbor(<integer>), <function>)
```

or

```
density.kernel.<kernel function>.joint(<algebra>, fixedWindow(<numeric>), <function>)
```

or

```
density.kernel.<kernel function>.joint(<algebra>, nearestNeighbor(<integer>), <function>)
```

<kernel function>. A kernel function. This specifies how data are weighted by the density function, depending on how close the data are to the current point.

<algebra>. Graph algebra, such as x*y. Refer to "Brief Overview of GPL Algebra" on page 3 for an introduction to graph algebra.

<numeric>. `fixedWindow` specifies the proportion of data points to include when calculating the smooth function. This takes a numeric value between 0 and 1 and is **optional**. You also have the option of using the `nearestNeighbor` function to calculate smoother's bandwidth.

<integer>. `nearestNeighbor` specifies the *k* number of nearest neighbors to includes when calculating the smooth function. This takes a positive integer and is **optional**. You also have the option of using the `fixedWindow` function to calculate the smoother's bandwidth.

<function>. One or more valid functions. These are optional. Use `scaledToData("false")` when comparing densities with very different same sizes.

joint. Used to create densities based on values in the first (*x* axis) and second (*y* axis) dimensions. Without the `joint` modifier, the density is based only on values in the first (*x* axis) dimension. You would typically use the modifier for 3-D densities.

Description

Calculates the probability density using a nonparametric kernel function. This is often used to add a distribution curve that does not assume a particular model (like normal or Poisson). You can use the `fixedWindow` function or the `nearestNeighbor` function to specify the smoother's bandwidth. If you do not specify an explicit bandwidth, the internal algorithm uses a fixed window whose size is determined by the underlying data values and the specific kernel function.

Examples

```
ELEMENT: line(position(density.kernel.epanechnikov(x)))
```

Figure 152. Example: Adding the default kernel distribution

```
ELEMENT: line(position(density.kernel.epanechnikov(x, fixedWindow(0.05))))
```

Figure 153. Example: Adding a kernel distribution using a fixed window

```
ELEMENT: line(position(density.kernel.epanechnikov(x, nearestNeighbor(100))))
```

Figure 154. Example: Adding a kernel distribution using k nearest neighbors

```
COORD: rect(dim(1,2,3))
ELEMENT: interval(position(density.kernel.epanechnikov.joint(x*y)))
```

Figure 155. Example: Creating a 3-D graph showing kernel densities

Kernel Functions

uniform. All data receive equal weights.

epanechnikov. Data near the current point receive higher weights than extreme data receive. This function weights extreme points more than the triweight, biweight, and tricube kernels but less than the Gaussian and Cauchy kernels.

biweight. Data far from the current point receive more weight than the triweight kernel allows but less weight than the Epanechnikov kernel permits.

tricube. Data close to the current point receive higher weights than both the Epanechnikov and biweight kernels allow.

triweight. Data close to the current point receive higher weights than any other kernel allows. Extreme cases get very little weight.

gaussian. Weights follow a normal distribution, resulting in higher weighting of extreme cases than the Epanechnikov, biweight, tricube, and triweight kernels.

cauchy. Extreme values receive more weight than the other kernels, with the exception of the uniform kernel, allow.

Valid Functions

“marron Function ” on page 173
“scaledToData Function ” on page 204
“segments Function ” on page 205

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244

“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.logistic Function

Syntax

```
density.logistic(<algebra>, location(<numeric>), scaleDensity(<numeric>), <function>)
```

or

```
density.logistic(<binning function>, location(<numeric>), scaleDensity(<numeric>), <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<integer>. `location` and `scaleDensity` define the parameters for the distribution. These take numeric values and are **required**.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the logistic distribution. This is often used to add a distribution curve. If you don't see the graphic element for the distribution, check the parameters for the distribution and the range for the *x* axis scale.

Because this function does not estimate parameters from the data, it can be used only for comparison and not for fitting.

Examples

```
ELEMENT: line(position(density.logistic(x, location(5), scaleDensity(2))))
```

Figure 156. Example: Adding a logistic distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238

“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.normal Function

Syntax

```
density.normal(<algebra>, mean(<numeric>), standardDeviation(<numeric>), <function>)
```

or

```
density.normal(<binning function>, mean(<numeric>), standardDeviation(<numeric>), <function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<numeric>. `mean` and `standardDeviation` define the parameters for the distribution. These take numeric values. You can use both of them or neither. If no parameters are specified, they are calculated from the underlying data.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the normal distribution. This is often used to add a distribution curve.

Examples

```
ELEMENT: interval(position(summary.count(bin.rect(x))))
ELEMENT: line(position(density.normal(x)))
```

Figure 157. Example: Adding a normal curve to a histogram

```
ELEMENT: line(position(density.normal(x, mean(50000), standardDeviation(15000))))
```

Figure 158. Example: Creating a normal curve with specific parameters

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229

“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.poisson Function

Syntax

```
density.poisson(<algebra>, rate(<numeric>), <function>)
```

or

```
density.poisson(<binning function>, rate(<numeric>), <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<numeric>. `rate` defines the parameter for the distribution. This takes a positive numeric value and is optional. If the parameter is not specified, it is calculated from the underlying data.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the Poisson distribution. This is often used to add a distribution curve. If you don't see the graphic element for the distribution, check the parameter for the distribution and the range for the *x* axis scale.

Examples

```
ELEMENT: line(position(density.poisson(x, rate(5.5))))
```

Figure 159. Example: Adding a Poisson distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229

“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.studentizedRange Function

Syntax

```
density.studentizedRange(<algebra>, degreesOfFreedom(<integer>), k(<numeric>), <function>)
```

or

```
density.studentizedRange(<binning function>, degreesOfFreedom(<integer>), k(<numeric>), <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<integer> and **<numeric>**. `degreesOfFreedom` and k define the parameters for the distribution. `degreesOfFreedom` takes an integer, and k takes a numeric value. These are **required**.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the Studentized range distribution. This is often used to add a distribution curve. If you don't see the graphic element for the distribution, check the parameters for the distribution and the range for the *x* axis scale.

Because this function does not estimate parameters from the data, it can be used only for comparison and not for fitting.

Examples

```
ELEMENT: line(position(density.studentizedRange(x, degreesOfFreedom(5), k(2.5))))
```

Figure 160. Example: Adding a Studentized range distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198

“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.t Function

Syntax

```
density.t(<algebra>, degreesOfFreedom(<integer>), <function>)
```

or

```
density.t(<binning function>, degreesOfFreedom(<integer>), <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<integer>. `degreesOfFreedom` defines the parameter for the distribution. `degreesOfFreedom` takes an integer and is **required**.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the Student's *t* distribution. This is often used to add a distribution curve. If you don't see the graphic element for the distribution, check the parameters for the distribution and the range for the *x* axis scale.

Because this function does not estimate parameters from the data, it can be used only for comparison and not for fitting.

Examples

```
ELEMENT: line(position(density.t(x, degreesOfFreedom(5))))
```

Figure 161. Example: Adding a Student's t distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190

“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.uniform Function

Syntax

```
density.uniform(<algebra>, min(<numeric>), max(<numeric>), <function>)
```

or

```
density.uniform(<binning function>, min(<numeric>), max(<numeric>), <function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function. The binning function is optional.

<numeric>. `min` and `max` define the parameters for the distribution. These take numeric values. You can use all of them or none of them. Any missing parameters are calculated from the underlying data.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the uniform distribution using the method-of-moments estimate. This is often used to add a distribution curve.

Examples

```
ELEMENT: line(position(density.uniform(x)))
```

Figure 162. Example: Adding a uniform distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190

“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

density.weibull Function

Syntax

```
density.weibull(<algebra>, rate(<numeric>), scaleDensity(<numeric>), <function>)
```

or

```
density.weibull(<binning function>, rate(<numeric>), scaleDensity(<numeric>), <function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<integer>. `rate` and `scaleDensity` define the parameters for the distribution. These take numeric values and are **required**.

<function>. A valid function. Use `scaledToData("false")` when comparing densities with very different same sizes.

Description

Calculates the probability density of the Weibull distribution. This is often used to add a distribution curve. If you don't see the graphic element for the distribution, check the parameters for the distribution and the range for the *x* axis scale.

Because this function does not estimate parameters from the data, it can be used only for comparison and not for fitting.

Examples

```
ELEMENT: line(position(density.logistic(x, location(5), scaleDensity(2))))
```

Figure 163. Example: Adding a Weibull distribution curve

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Valid Functions

“scaledToData Function ” on page 204

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184

“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

dim Function

Syntax

```
dim(<numeric> ...)
```

<numeric>. A numeric value identifying the dimension or dimensions. If you are specifying multiple dimensions, use commas to separate the numeric values.

Description

Specifies the dimension or dimensions to which a coordinate type, scale, guide, or function applies.

To figure out the numeric value associated with a dimension, look at the algebra. Counting the crossings gives the main dimension values. The coordinate system (including any clustering of the coordinate system) doesn't matter.

Consider the following algebra:

```
a*b*c*d
```

The variables in the algebra correspond to the following dimensions:

Table 2. Variables and dimensions

Variable	**Dimension**
a	dim(1)
b	dim(2)
c	dim(3)
d	dim(4)

Blended variables cannot be separated. The blend of the two variables corresponds to one dimension. Consider the following:

```
a*(b+c)*d
```

The variables in the algebra correspond to the following dimensions:

Table 3. Variables and dimensions with blended variables

Variable	**Dimension**
a	dim(1)
b+c	dim(2)
d	dim(3)

With nesting, you still count crossed variables, but nested groups are counted only once. To refer to each variable in the nested group, you count from the outside in, using a dot convention. The outermost variable in the nested group gets the primary dimension number (for example, `dim(1)`), and the next variable gets the primary dimension number followed by a dot and a 1 (for example, `dim(1.1)`). Consider the following:

```
a*b/c*d
```

The variables in the algebra correspond to the following dimensions:

Variable	**Dimension**
a	dim(1)
c	dim(2)
b	dim(2.1)
d	dim(3)

Examples

```
COORD: rect(dim(1,2))
```

Figure 164. Example: Specifying a two-dimensional, rectangular coordinate system

```
GUIDE: axis(dim(2), label("Mean Salary"))
```

Figure 165. Example: Specifying an axis label for the second dimension

Applies To

“base.coordinate Function ” on page 64
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.rect Function ” on page 72
“reflect Function ” on page 183
“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“axis Guide Type ” on page 39
“asn Scale ” on page 28
“atanh Scale ” on page 28
“cat Scale ” on page 29
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“logit Scale ” on page 33
“pow Scale ” on page 33
“prob Scale ” on page 34
“probit Scale ” on page 35
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

end Function

Syntax

```
end()
```

Description

Specifies the end of the GPL block that defines a particular graph or page.

Examples

```
GRAPH: begin()
ELEMENT: line(position(x*y))
GRAPH: end()
```

Figure 166. Example: Defining a particular graph

```
PAGE: begin(scale(400px,300px))
SOURCE: s=csvSource(file("mydata.csv"))
DATA: x=col(source(s), name("x"))
DATA: y=col(source(s), name("y"))
ELEMENT: line(position(x*y))
PAGE: end()
```

Figure 167. Example: Defining a page

Applies To

“GRAPH Statement ” on page 20
“PAGE Statement ” on page 20

eval Function

Syntax

```
eval(<expression>)
```

<expression>. A mathematical expression, such as `log(salary)`.

Description

Evaluates a mathematical expression for each case in the data. You can use many different mathematical functions in the expression. See the topic “eval Operators and Functions ” on page 118 for more information. If needed, you can wrap the result of a function in another function. Therefore, `datetostring(date())` is a valid expression.

The `eval` function is also useful for evaluating a Boolean expression whose result can be used in the `summary.countTrue`, `summary.percentTrue`, and `summary.proportionTrue` functions. This combination allows you to plot the number or percent of cases greater than or less than a specific value.

Examples

```
TRANS: saldiff = eval(salary-salbegin)
ELEMENT: point(position(summary.mean(jobcat*saldiff)))
```

Figure 168. Example: Plotting the difference between two variables

```
DATA: x = iter(-100,100,1)
TRANS: y = eval(x**2)
ELEMENT: line(position(x*y))
```

Figure 169. Example: Creating a graph from an equation

```
TRANS: salGreaterThan = eval(salary>50000)
ELEMENT: interval(position(summary.percentTrue(jobcat*salGreaterThan)))
```

Figure 170. Example: Plotting percent greater than a value

Applies To

“TRANS Statement ” on page 22
“collapse Function ” on page 80

eval Operators and Functions

Following are the operators and functions that you can use with the `eval` function. See “eval Function ” on page 118 for information about the `eval` function.

Table 4. Operators

Operator	Meaning	Notes
+	Addition or string concatenation	Using + with numbers adds the numbers. Using it with strings concatenates the strings.
-	Subtraction	
*	Multiplication	
/	Division	

Table 4. Operators (continued)		
Operator	**Meaning**	**Notes**
()	Grouping	Grouped expressions are calculated before other expressions.
**	Exponentiation	
==	Equal	
!=	Not equal	
<	Less than	
>	Greater than	
<=	Less than or equal to	
>=	Greater than or equal to	
&&	Logical AND	
||	Logical OR	
? :	Conditional	These operators are shorthand for *then-else* when evaluating a Boolean operand. For example, `x>15?"High":"Low"` returns "High" if x > 15. Otherwise, the expression returns "Low".

Table 5. Mathematical Functions		
Function	**Result**	**Notes**
abs(n)	The absolute value of *n*	
acos(n)	The inverse cosine (arcosine) of *n*	
asin(n)	The inverse sine (arcsine) of *n*	
atan(n)	The inverse tangent (arctangent) of *n*	
atanh(n)	The hyperbolic inverse tangent (hyperbolic arctangent) of *n*	
ceil(n)	The smallest integer that is greater than *n*	Round up
cos(n)	The cosine of *n*	
cosh(n)	The hyperbolic cosine of *n*	
exp(n)	*e* raised to the power *n*, where *e* is the base of the natural logarithms	
floor(n)	The largest integer that is less than *n*	Round down
gamma(n)	The complete Gamma function of *n*	

Table 5. Mathematical Functions (continued)		
Function	**Result**	**Notes**
int(n)	The value of *n* truncated to an integer (toward 0)	
lgamma(n)	The logarithm of the complete Gamma function of *n*	
log(n)	The natural (base-e) logarithm of *n*	
log2(n)	The base-2 logarithm of *n*	
log10(n)	The base-10 logarithm of *n*	
mod(n, modulus)	The remainder when *n* is divided by *modulus*	
pow(n, power)	The value of *n* raised to the power of *power*	
round(n)	The integer that results from rounding the absolute value of *n* and then reaffixing the sign. Numbers ending in 0.5 exactly are rounded away from 0. For example, `round(-4.5)` rounds to -5.	
sin(n)	The sine of *n*	
sinh(n)	The hyperbolic sine of *n*	
sqrt(n)	The positive square root of *n*	
tan(n)	The tangent of *n*	
tanh(n)	The hyperbolic tangent of *n*	

Table 6. String Functions		
Function	**Result**	**Notes**
concatenate(string1, string2)	A string that is the concatenation of *string1* and *string2*	
datetostring(date)	The string that results when *date* is converted to a string	

Table 6. String Functions (continued)

Function	Result	Notes
indexof(haystack,needle[,divisor])	A number that indicates the position of the first occurrence of *needle* in *haystack*. The optional third argument, *divisor*, is a number of characters used to divide *needle* into separate strings. Each substring is used for searching and the function returns the first occurrence of any of the substrings. For example, indexof(x, “abcd”) will return the value of the starting position of the complete string "abcd" in the string variable x; indexof(x, “abcd”, 1) will return the value of the position of the first occurrence of any of the values in the string; and indexof(x, “abcd”, 2) will return the value of the first occurrence of either "ab" or "cd". Divisor must be a positive integer and must divide evenly into the length of needle. Returns 0 if *needle* does not occur within *haystack*.	
length(string)	A number indicating the length of *string*	
lowercase(string)	*string* with uppercase letters changed to lowercase and other characters unchanged	
ltrim(string[, char])	*string* with any leading instances of *char* removed. If *char* is not specified, leading blanks are removed. Char must resolve to a single character.	
midstring(string , start, end)	The substring beginning at position *start* of *string* and ending at *end*	
numbertostring(n)	The string that results when *n* is converted to a string	
replace(target, old, new)	In *target*, instances of *old* are replaced with *new*. All arguments are strings.	
rtrim(string[, char])	*string* with any trailing instances of *char* removed. If *char* is not specified, trailing blanks are removed. Char must resolve to a single character.	

Table 6. String Functions (continued)		
Function	**Result**	**Notes**
stringtodate(string)	The value of the string expression *string* as a date	
stringtonumber(string)	The value of the string expression *string* as a number	
substring(string, start, length)	The substring beginning at position *start* of *string* and running for length *length*	
trim(string)	*string* with any leading and trailing blanks removed	
uppercase(string)	*string* with lowercase letters changed to uppercase and other characters unchanged	

Table 7. Date and Time Functions		
Function	**Result**	**Notes**
date()	The current date	
time()	The current time	

Table 8. Constants		
Constant	**Meaning**	**Notes**
true	True	
false	False	
pi	pi	
e	Euler's number or the base of the natural logarithm	

exclude Function

Syntax

```
exclude("category name" ...)
```

<category name>. The string representing the category on the axis. If specifying multiple categories, separate them with commas.

Description

Excludes the categories from the axis. These categories are not displayed on the axis. This function can be used only with categorical scales for dimensions, not scales for aesthetics.

Examples

```
SCALE: cat(dim(1), exclude("No Response"))
```

Figure 171. Example: Exclude a category

```
SCALE: cat(dim(1), exclude("No Response", "Didn't Ask"))
```

Figure 172. Example: Excluding multiple categories

Applies To
“cat Scale ” on page 29

exponent Function

Syntax

```
exponent(<numeric>)
```

<numeric>. A numeric value (including negative values) indicating the exponent for the power scale.

Description

Specifies an exponent for a power scale.

Examples

```
SCALE: power(dim(2), exponent(3))
```

Figure 173. Example: Specifying a different power exponent

Applies To
“pow Scale ” on page 33
“safePower Scale ” on page 36

exponential Function

Syntax

```
exponential(<rate>)
```

<rate>. Numeric value specifying the rate parameter for the distribution.

Description

Specifies an exponential distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), exponential(1.5))
```

Figure 174. Example: Specifying an exponential distribution for the probability scale

Applies To
“prob Scale ” on page 34

f Function

Syntax

```
f(<degrees of freedom>, <degrees of freedom>)
```

<degrees of freedom>. Numeric values specifying the degrees of freedom parameters for the distribution. Values must be greater than 0.

Description

Specifies an *F* distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), f(5, 2))
```

Applies To
“prob Scale ” on page 34

format Function

Syntax

```
format("date format")
```

<format>. The date format of the data.

Description

Indicates the input format for a date variable in the source. This function does not change the format for the date; it only specifies the format that GPL should expect. Use one or more of the following abbreviations and date separators to indicate the exact format.

Table 9. Abbreviations for date formats

Abbreviation	Meaning
M	Month
d	Day
y	Year
m	Minute
s	Second

Note: In IBM SPSS Statistics, dates from a `userSource` are passed to GPL as numeric values. Therefore, this function does not have any affect on a `userSource`.

Examples

```
DATA: date = col(source(mydata), name("date"), unit.time(), format("M/d/yyyy"))
```

Figure 175. Example: Indicating a date format of the form 1/31/2006

Applies To

“col Function ” on page 79

format.date Function

Syntax

```
format.date()
```

Description

Specifies that the data are formatted as dates when displayed in an axis' tick marks. The underlying data must be dates or times.

Examples

```
GUIDE: axis(dim(1), format.date())
```

Figure 176. Example: Displaying dates on an axis

Applies To

“axis Guide Type ” on page 39

format.dateTime Function

Syntax

```
format.dateTime()
```

Description

Specifies that the are formatted as dates and times when displayed in an axis' tick marks. The underlying data must be dates or times.

Examples

```
GUIDE: axis(dim(1), format.dateTime())
```

Figure 177. Example: Displaying dates and times on an axis

Applies To

“axis Guide Type ” on page 39

format.time Function

Syntax

```
format.time()
```

Description

Specifies that the data are formatted times when displayed in an axis' tick marks. The underlying data must be dates or times.

Examples

```
GUIDE: axis(dim(1), format.time())
```

Figure 178. Example: Displaying times on an axis

Applies To

“axis Guide Type ” on page 39

from Function

Syntax

```
from(<variable name>)
```

<variable name>. The name of a variable previously defined in the GPL by a DATA statement.

Description

Specifies one of the pair of nodes that defines an edge relation. The is the node that defines the starting point for the edge.

Examples

```
ELEMENT: edge(position(layout.dag(node(id), from(fromVar), to(toVar))))
```

Figure 179. Example: Creating a directed acyclic graph

Applies To

“layout.circle Function ” on page 131
“layout.dag Function ” on page 133
“layout.data Function ” on page 135
“layout.grid Function ” on page 137
“layout.network Function ” on page 139
“layout.random Function ” on page 141
“layout.tree Function ” on page 143

gamma Function

Syntax

```
gamma(<rate>)
```

<rate>. Numeric value specifying the shape parameter for the distribution. This values must be greater than 0.

Description

Specifies a gamma distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), gamma(2.5))
```

Figure 180. Example: Specifying a gamma distribution for the probability scale

Applies To

“prob Scale ” on page 34

gap Function

Syntax

```
gap(<value>)
```

<value>. A number with units (for example, 0px).

Description

Specifies the size of the gap between adjacent axes in a faceted graph. This function is used to close the space between adjacent axes in population pyramids and matrix scatterplots.

Examples

```
GUIDE: axis(dim(3), gap(0px))
```

Figure 181. Example: Forcing adjacent axes to touch

Applies To

“axis Guide Type ” on page 39

gridlines Function

Syntax

```
gridlines()
```

Description

Specifies that grid lines should be drawn for the axis. These are lines drawn from the major tick marks to the opposite side of the graph. They can assist in determining the exact location of a graphic element in the graph.

Examples

```
GUIDE: axis(dim(2), gridlines())
```

Figure 182. Example: Displaying grid lines

Applies To

“axis Guide Type ” on page 39

in Function

Syntax

```
in(min(<value>), max(<value>))
```

<value>. Numeric values for defining the range that determines which values to include.

Description

Includes only the continuous values that are in the range specified by the `min` and `max` parameters. This function is valid only for continuous variables.

Examples

```
DATA: gender = col(source(mydata), name("salary"),
                          in(min(0), max(50000)))
```

Figure 183. Example: Including only a subset of continuous values

Applies To

“col Function ” on page 79

include Function

Syntax

```
include(<value> ...)
```

<value>. The string representing the category on the axis or a numeric value on the axis. If specifying multiple values, separate them with commas.

Description

Includes the categories or values on the axis or legend, even if the data do not include the categories or values. These categories or values are always displayed on the axis or legend. For example, you may use `include(0)` in a bar chart to ensure bars begin at 0.

Examples

```
SCALE: cat(dim(1), include("No Response"))
```

Figure 184. Example: Include a category

```
SCALE: cat(dim(1), include("No Response", "Didn't Ask"))
```

Figure 185. Example: Including multiple categories

```
SCALE: linear(dim(2), include(0))
```

Figure 186. Example: Include a value

```
SCALE: linear(dim(2), include(0, 100))
```

Figure 187. Example: Including multiple values

Applies To

“cat Scale ” on page 29
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“pow Scale ” on page 33
“safeLog Scale ” on page 35

“safePower Scale ” on page 36
“time Scale ” on page 37

index Function

Syntax

```
index()
```

Description

Creates a new variable by indexing each case with an integer. The new variable is essentially a case number.

Examples

```
TRANS: casenum = index()
ELEMENT: point(position(x*y), label(casenum))
```

Figure 188. Example: Create an index variable and label by the variable

Applies To

“TRANS Statement ” on page 22
“collapse Function ” on page 80

iter Function

Syntax

```
iter(<from>, <to>, <step>)
```

<from>. The first value in the new column. Subsequent values are iterated from this one.

<to>. The maximum value that the new column can contain.

<step>. A value defining the amount by which values are iterated.

Description

Creates a new column of data with values in a specified range. Intermediate values are calculated by adding the step value to the previous value. For example, `iter(1,5,1)` generates the values 1, 2, 3, 4, 5. `iter(1,10,2)` generates the values 1, 3, 5, 7, 9. Note that 10 is not included in the second example because it cannot be iterated from the previous value.

Examples

```
DATA: x = iter(-100,100,1)
TRANS: y = eval(x**2)
ELEMENT: line(position(x*y))
```

Figure 189. Example: Creating a graph from an equation

Applies To

“DATA Statement ” on page 21

jump Function

Syntax

```
jump()
```

Description

Used with `smooth.step`, `smooth.step.left`, `smooth.step.center`, and `smooth.step.right` to indicate that the interpolation line or area jumps to the next value. There is no vertical line connecting the values.

Examples

```
ELEMENT: line(position(smooth.step(educ*salary)), jump())
```

Figure 190. Example: Specifying an interpolation line

Applies To

“area Element ” on page 44
“line Element ” on page 47

label Function (For GPL Graphic Elements)

Note: If you are modifying the label for a guide (like an axis), refer to “label Function (For GPL Guides)” on page 130 .

Syntax

```
label("label text", <function>)
```

or

```
label(<algebra>, <function>)
```

or

```
label(<statistic function>, <function>)
```

"label text". The text that appears in the label. Multiple strings are concatenated when each string is separated by a comma (for example, `label("This is a ", "long label")`).

<function>. One or more valid functions. These are optional.

<algebra>. Graph algebra, using one variable or a blend of variables.

<statistic function>. A valid statistic function.

Description

Specifies a label for a graphic element. The label appears on the graphic element. Multiple label functions can be specified. The result of each label function is displayed on a separate line in the graph.

Examples

```
ELEMENT: point(position(salbegin*salary), label(gender))
```

Figure 191. Example: Labeling by another variable

```
ELEMENT: point(position(summary.count(jobcat)), label(summary.count()))
```

Figure 192. Example: Labeling by the result of a statistic

```
ELEMENT: interval(position(summary.mean(jobcat*salary)), label(summary.mean(salary)))
```

Figure 193. Example: Labeling by the result of a statistic

```
ELEMENT: interval(position(summary.mean(jobcat*salary)), label("Count:"), label(summary.count()))
```

Figure 194. Example: Creating a multi-line label

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions

“showAll Function ” on page 207

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

label Function (For GPL Guides)

Note: If you are modifying the label for a graphic element (like a bar or point), refer to “label Function (For GPL Graphic Elements)” on page 129 .

Syntax

```
label("label text" ...)
```

"label text". The text that appears in the label. You can specify multiple strings by separating the strings with commas (for example, `label("This is a ", "long label")`). The strings are concatenated in the resulting graph.

Description

Specifies a label for a guide (for example, an axis or legend). This is text that is displayed on the resulting graph.

Examples

```
GUIDE: axis(dim(1), label("Job Category"))
```

Figure 195. Example: Specifying an axis title

```
GUIDE: legend(aesthetic(aesthetic.color), label("Gender"))
```

Figure 196. Example: Specifying a legend title

```
GUIDE: text.title(label("Sales By Region"))
```

Figure 197. Example: Specifying a graph title

Applies To

“axis Guide Type ” on page 39
“form.line Guide Type ” on page 40
“legend Guide Type ” on page 40
“text.footnote Guide Type ” on page 41
“text.subfootnote Guide Type ” on page 41
“text.subsubfootnote Guide Type ” on page 42
“text.subtitle Guide Type ” on page 42
“text.subsubtitle Guide Type ” on page 42
“text.title Guide Type ” on page 43

layout.circle Function

Syntax

```
layout.circle(<function>))
```

or

```
layout.circle(<algebra>, <function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra for network graphs is `1*1` because the position of elements is determined by the layout method and is not tied to a coordinate value (such as a value on a dimension). This algebra is implied and needs to be specified for network graphs only when faceting is needed.

<functions>. Valid functions. The `from` and `to` functions are required. The `node` function is optional for edges, allowing you to draw edges without a separate node data source.

Description

Lays out graphic elements in a circle. The function is used for network graphs, which are visual representations of data that consist of nodes and relations between nodes (edges). The circle l layout is a layout that can be applied to any graph. It lays out a graph assuming that links are undirected and treats all nodes identically. Nodes are placed only around the perimeter of a circle

Note: Network graphs that display nodes and edges require two data sources, one for the unique nodes and one for the edges. If the edge data source includes weights and the weight variable is indicated in the SOURCE statement, the weights influence the length of edges in the graph, with higher weights having shorter edges.

Examples

```
ELEMENT: edge(position(layout.circle(node(id), from(fromVar), to(toVar))))
ELEMENT: point(position(layout.circle(node(id), from(fromVar), to(toVar))), label(id))
```

Figure 198. Example: Creating a circular network diagram

Valid Functions

“from Function ” on page 125
“node Function ” on page 177
“to Function ” on page 300

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162

“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296

“transparency Function (For GPL Graphic Elements)” on page 300

layout.dag Function

Syntax

```
layout.dag(<function>))
```

or

```
layout.dag(<algebra>, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra for network graphs is 1*1 because the position of elements is determined by the layout method and is not tied to a coordinate value (such as a value on a dimension). This algebra is implied and needs to be specified for network graphs only when faceting is needed.

<functions>. Valid functions. The `from` and `to` functions are required. The `node` function is optional for edges, allowing you to draw edges without a separate node data source.

Description

Lays out graphic elements as a directed acyclic graph (DAG). The function is used for network graphs, which are visual representations of data that consist of nodes and relations between nodes (edges). The DAG layout should be used only for directed graphs *without* a primary root node (compare with `layout.tree`). This layout produces tree-like structures from parent nodes down to leaf nodes, so the layout works well with hierarchical data.

Note: Network graphs that display nodes and edges require two data sources, one for the unique nodes and one for the edges. If the edge data source includes weights and the weight variable is indicated in the SOURCE statement, the weights influence the length of edges in the graph, with higher weights having shorter edges.

Examples

```
ELEMENT: edge(position(layout.dag(node(id), from(fromVar), to(toVar))))
ELEMENT: point(position(layout.dag(node(id), from(fromVar), to(toVar))), label(id))
```

Figure 199. Example: Creating a directed acyclic graph

Valid Functions

“from Function ” on page 125
“node Function ” on page 177
“to Function ” on page 300

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155

“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290

“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

layout.data Function

Syntax

```
layout.data(<algebra>, <function>)
```

or

```
layout.data(<function>))
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. Unlike other network graphs, the algebra determines the position of elements and ties the position to a specific coordinate value (such as a value on a dimension). You can omit the algebra for edges if you are using algebra in another ELEMENT statement to specify the position of the points.

<functions>. Valid functions. The `from` and `to` functions are required. The node function is optional for edges, allowing you to draw edges without a separate node data source.

Description

Lays out graphic elements in the coordinate positions specified by the data. The function is used for network graphs, which are visual representations of data that consist of nodes and relations between nodes (edges).

Note: Network graphs that display nodes and edges require two data sources, one for the unique nodes and one for the edges. If the edge data source includes weights and the weight variable is indicated in the SOURCE statement, the weights influence the length of edges in the graph, with higher weights having shorter edges.

Examples

```
ELEMENT: point(position(layout.data(x*y, node(id), from(fromVar), to(toVar))), label(id))
ELEMENT: edge(position(layout.data(node(id), from(fromVar), to(toVar))))
```

Figure 200. Example: Creating a network diagram

Valid Functions

“from Function ” on page 125
“node Function ” on page 177
“to Function ” on page 300

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151

“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286

“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

layout.grid Function

Syntax

```
layout.grid(<function>))
```

or

```
layout.grid(<algebra>, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra for network graphs is 1*1 because the position of elements is determined by the layout method and is not tied to a coordinate value (such as a value on a dimension). This algebra is implied and needs to be specified for network graphs only when faceting is needed.

<functions>. Valid functions. The `from` and `to` functions are required. The node function is optional for edges, allowing you to draw edges without a separate node data source.

Description

Lays out graphic elements in a grid. The function is used for network graphs, which are visual representations of data that consist of nodes and relations between nodes (edges). The grid layout is a general layout that can be applied to any graph. It lays out a graph assuming that links are undirected and treats all nodes identically. Nodes are placed only at grid points within the space.

Note: Network graphs that display nodes and edges require two data sources, one for the unique nodes and one for the edges. If the edge data source includes weights and the weight variable is indicated in the SOURCE statement, the weights influence the length of edges in the graph, with higher weights having shorter edges.

Examples

```
ELEMENT: edge(position(layout.grid(node(id), from(fromVar), to(toVar))))
ELEMENT: point(position(layout.grid(node(id), from(fromVar), to(toVar))), label(id))
```

Figure 201. Example: Creating a grid network diagram

Valid Functions

“from Function ” on page 125
“node Function ” on page 177
“to Function ” on page 300

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147

“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282

“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

layout.network Function

Syntax

```
layout.network(<function>))
```

or

```
layout.network(<algebra>, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra for network graphs is 1*1 because the position of elements is determined by the layout method and is not tied to a coordinate value (such as a value on a dimension). This algebra is implied and needs to be specified for network graphs only when faceting is needed.

<functions>. Valid functions. The `from` and `to` functions are required. The node function is optional for edges, allowing you to draw edges without a separate node data source.

Description

Lays out graphic elements in a network. The function is used for network graphs, which are visual representations of data that consist of nodes and relations between nodes (edges). The network layout is a general layout that can be applied to any graph. It lays out a graph assuming that links are undirected and treats all nodes identically. Nodes are placed freely within the space.

Note: Network graphs that display nodes and edges require two data sources, one for the unique nodes and one for the edges. If the edge data source includes weights and the weight variable is indicated in the SOURCE statement, the weights influence the length of edges in the graph, with higher weights having shorter edges.

Examples

```
ELEMENT: edge(position(layout.network(node(id), from(fromVar), to(toVar))))
ELEMENT: point(position(layout.network(node(id), from(fromVar), to(toVar))), label(id))
```

Figure 202. Example: Creating a network diagram

Valid Functions

“from Function ” on page 125
“node Function ” on page 177
“to Function ” on page 300

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85

“layout.tree Function ” on page 143
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275

“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

layout.random Function

Syntax

```
layout.random(<function>))
```

or

```
layout.random(<algebra>, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra for network graphs is 1*1 because the position of elements is determined by the layout method and is not tied to a coordinate value (such as a value on a dimension). This algebra is implied and needs to be specified for network graphs only when faceting is needed.

<functions>. Valid functions. The `from` and `to` functions are required. The node function is optional for edges, allowing you to draw edges without a separate node data source.

Description

Lays out graphic elements randomly. The function is used for network graphs, which are visual representations of data that consist of nodes and relations between nodes (edges). The network layout is a general layout that can be applied to any graph. It lays out a graph assuming that links are undirected and treats all nodes identically. Nodes are placed randomly within the space.

Note: Network graphs that display nodes and edges require two data sources, one for the unique nodes and one for the edges. If the edge data source includes weights and the weight variable is indicated in the SOURCE statement, the weights influence the length of edges in the graph, with higher weights having shorter edges.

Examples

```
ELEMENT: edge(position(layout.random(node(id), from(fromVar), to(toVar))))
ELEMENT: point(position(layout.random(node(id), from(fromVar), to(toVar))), label(id))
```

Figure 203. Example: Creating a random network diagram

Valid Functions

“from Function ” on page 125
“node Function ” on page 177
“to Function ” on page 300

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81

“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272

“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

layout.tree Function

Syntax

```
layout.tree(<function>))
```

or

```
layout.tree(<algebra>, <function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra. The algebra for network graphs is `1*1` because the position of elements is determined by the layout method and is not tied to a coordinate value (such as a value on a dimension). This algebra is implied and needs to be specified for network graphs only when faceting is needed.

<functions>. Valid functions. The `from` and `to` functions are required. The `node` function is optional for edges, allowing you to draw edges without a separate node data source. Also, you should use the `root` function when you want to ensure the correct node is used as the root node.

Description

Lays out graphic elements as a directed tree. The function is used for network graphs, which are visual representations of data that consist of nodes and relations between nodes (edges). The tree layout should be used only for directed graphs *with* a primary root node (compare with `layout.dag`). This layout produces tree-like structures from parent nodes down to leaf nodes, so the layout works well with hierarchical data. If the root node is not specified by the `root` function, the function picks the most likely node as the root.

Note: Network graphs that display nodes and edges require two data sources, one for the unique nodes and one for the edges. If the edge data source includes weights and the weight variable is indicated in the SOURCE statement, the weights influence the length of edges in the graph, with higher weights having shorter edges.

Examples

```
ELEMENT: edge(position(layout.tree(node(id), from(fromVar), to(toVar), root("A"))))
ELEMENT: point(position(layout.tree(node(id), from(fromVar), to(toVar), root("A"))), label(id))
```

Figure 204. Example: Creating a tree

Valid Functions

“from Function ” on page 125
“root Function ” on page 201
“to Function ” on page 300

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260

“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.alpha Function

Syntax

```
link.alpha(<algebra>, radius(<numeric>))
```

or

```
link.alpha(<binning function>, radius(<numeric>))
```

or

```
link.alpha(<statistic function>, radius(<numeric>))
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<numeric>. A numeric value indicating the This is **required**.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.alpha`.

Description

Calculates the alpha shape for the values. This function is typically used with the edge graphic element. The alpha shape is a generalization of the convex hull. If the radius is sufficiently large, the result is a convex hull. For smaller radius values, the shape shrinks and becomes concave. It also may not connect or contain some data values.

Examples

```
ELEMENT: edge(position(link.alpha(x*y, radius(50))))
```

Figure 205. Example: Creating an alpha shape graph

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252

“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.complete Function

Syntax

```
link.complete(<algebra>)
```

or

```
link.complete(<binning function>)
```

or

```
link.complete(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.complete`.

Description

Calculates the complete graph for the values. This function is typically used with the edge graphic element. The complete graph connects every data value with every other data value.

Examples

```
ELEMENT: edge(position(link.complete(x*y)))
```

Figure 206. Example: Creating a complete graph

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250

“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.delaunay Function

Syntax

```
link.delaunay(<algebra>)
```

or

```
link.delaunay(<binning function>)
```

or

```
link.delaunay(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.delaunay`.

Description

Calculates the Delaunay triangulation for the values. This function is typically used with the edge graphic element. The triangulation connects all values so that the connecting segments form triangles.

Examples

```
ELEMENT: edge(position(link.delaunay(x*y)))
```

Figure 207. Example: Creating a Delaunay triangulation

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248

“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.distance Function

Syntax

```
link.distance(<algebra>, radius(<numeric>))
```

or

```
link.distance(<binning function>, radius(<numeric>))
```

or

```
link.distance(<statistic function>, radius(<numeric>))
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<numeric>. A numeric value indicating the distance to determine whether values are connected.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.distance`.

Description

Calculates the distance graph for the values. This function is typically used with the edge graphic element. The distance graph connects any two values whose distance is less than or equal to the specified radius.

Examples

```
ELEMENT: edge(position(link.distance(x*y), radius(5000)))
```

Figure 208. Example: Creating a distance graph

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242

“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.gabriel Function

Syntax

```
link.gabriel(<algebra>)
```

or

```
link.gabriel(<binning function>)
```

or

```
link.gabriel(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.gabriel`.

Description

Calculates the Gabriel graph for the values. This function is typically used with the edge graphic element. A Gabriel graph connects values if they are Gabriel neighbors. Gabriel neighbors are defined by imagining a circle whose diameter is the line connecting two values. The values are Gabriel neighbors if the circle doesn't contain any other values.

Examples

```
ELEMENT: edge(position(link.gabriel(x*y)))
```

Figure 209. Example: Creating a Gabriel graph

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238

“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.hull Function

Syntax

```
link.hull(<algebra>)
```

or

```
link.hull(<binning function>)
```

or

```
link.hull(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.hull`.

Description

Calculates the convex hull around the values. This function is typically used with the edge graphic element. A convex hull connects the least number of outermost values so that the hull contains all values. The hull contains all possible connections between any two values. Note that the convex hull is the boundary of the Delaunay triangulation.

Examples

```
ELEMENT: edge(position(link.hull(x*y)))
```

Figure 210. Example: Creating a convex hull

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229

“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.influence Function

Syntax

```
link.influence(<algebra>)
```

or

```
link.influence(<binning function>)
```

or

```
link.influence(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.influence`.

Description

Calculates a sphere of influence graph for the values. This function is typically used with the edge graphic element. The sphere of influence graph connects values if the distance between two values is less than or equal to the sum of the nearest neighbor distances for the two values. The nearest neighbor distance for a value is the distance between it and the value closest to it.

Examples

```
ELEMENT: edge(position(link.influence(x*y)))
```

Figure 211. Example: Creating a sphere of influence graph

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198

“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.join Function

Syntax

```
link.join(<algebra>)
```

or

```
link.join(<binning function>)
```

or

```
link.join(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.join`.

Description

Joins sets of points from a blend. `link.join` may be used for repeated measures or observations over time, among others. In all cases, there is a blend that defines the relation. This distinguishes `link.join` from the other link functions.

Examples

```
ELEMENT: edge(position(link.join(x1*y1 + x2*y2)), label(a))
ELEMENT: point(position(x1*y1 + x2*y2), label("Before"+"After"))
```

Figure 212. Example: Creating a bridge plot

This example assumes data that is in a format like the following:

Table 10. Example data

a	x1	y1	x2	y2
Bill	45	50	58	67
Alice	32	40	33	40
Bob	22	31	26	35
Audrey	55	59	52	64

```
TRANS: zero = eval(0)
ELEMENT: edge(position(link.join(zero*zero + x*y)), shape(shape.arrow))
```

Figure 213. Example: Drawing vectors from the origin

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151

“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288

“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.mst Function

Syntax

```
link.mst(<algebra>)
```

or

```
link.mst(<binning function>)
```

or

```
link.mst(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.mst`.

Description

Calculates the minimum spanning tree (MST) to connect the values specified by the algebra. This function is typically used with the edge graphic element. The MST connects all values by the shortest distance and never intersects a value twice.

Examples

```
ELEMENT: edge(position(link.mst(x*y)))
```

Figure 214. Example: Creating a minimal spanning tree

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147

“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284

“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.neighbor Function

Syntax

```
link.neighbor(<algebra>, neighborCount(<integer>))
```

or

```
link.neighbor(<binning function>, neighborCount(<integer>))
```

or

```
link.neighbor(<statistic function>, neighborCount(<integer>))
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<integer>. An integer defining the number of neighboring values to connect to a value.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.neighbor`.

Description

Calculates the nearest neighbor graph for the values. This function is typically used with the edge graphic element. The nearest neighbor graph connects a value *p* to the specified number of values with the shortest distance to *p*.

Examples

```
ELEMENT: edge(position(link.neighbor(x*y, neighborCount(3))))
```

Figure 215. Example: Creating a nearest neighbor graph

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84

“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277

“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.relativeNeighborhood Function

Syntax

```
link.relativeNeighborhood(<algebra>)
```

or

```
link.relativeNeighborhood(<binning function>)
```

or

```
link.relativeNeighborhood(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.relativeNeighborhood`.

Description

Calculates the relative neighborhood graph for the values. This function is typically used with the edge graphic element. A relative neighborhood graph connects values if they are relative neighbors. Relative neighbors are defined by imagining two circles whose centers are the two values, where the radius of the circles is the distance between the values. If the area created by the intersection of the two circles does not contain any other values, the values are relative neighbors.

Examples

```
ELEMENT: edge(position(link.relativeNeighborhood(x*y)))
```

Figure 216. Example: Creating a relative neighborhood graph

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
Applies To
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70

“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270

“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.sequence Function

Syntax

```
link.sequence(<algebra>)
```

or

```
link.sequence(<binning function>)
```

or

```
link.sequence(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.sequence`.

Description

Connects the values in the order in which their associated cases appear in the dataset. This function is typically used with the edge graphic element. In many cases, the result is the same as what you obtain by using the path graphic element without a statistic.

Examples

```
ELEMENT: edge(position(link.sequence(x*y)))
```

Figure 217. Example: Creating a sequence graph

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
Applies To
“bin.dot Function ” on page 66

“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267

“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

link.tsp Function

Syntax

```
link.tsp(<algebra>)
```

or

```
link.tsp(<binning function>)
```

or

```
link.tsp(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `link.tsp`.

Description

Calculates the solution to the travelling salesman problem (TSP) for the values. This function is typically used with the edge or path graphic element. The solution to the TSP is the shortest path that traverses all values once and starts and ends at the same value.

Examples

```
ELEMENT: edge(position(link.tsp(x*y)))
```

Figure 218. Example: Drawing the solution to the travelling salesman problem

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262

“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

logistic Function

Syntax

```
logistic(<location>, <scale>)
```

<location>. Numeric value specifying the location parameter for the distribution.

<scale>. Numeric value specifying the scale parameter for the distribution. This value must be greater than 0.

Description

Specifies a logistic distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), logistic(5, 2))
```

Figure 219. Example: Specifying a logistic distribution for the probability scale

Applies To
“prob Scale ” on page 34

map Function

Syntax

```
map((<value>, <aesthetic>) ...)
```

<value>. A categorical value that is being mapped to a specific aesthetic.

<aesthetic>. A valid aesthetic value or constant (for example, `color.red` or `size."5px"`) that will be used for the categorical value.

Note: A value and aesthetic pair is enclosed in parentheses. If you are specifying multiple pairs, use commas to separate the pairs.

Description

Maps a specific categorical value to a specific aesthetic value. For example, if you were creating a bar chart showing the median income in each U.S. state, you could use the map function to force the color of the bar corresponding to Illinois to be blue.

Examples

```
SCALE: cat(aesthetic(aesthetic.color), map(("IL", color.blue)))
```

Figure 220. Example: Mapping a category to a color

```
SCALE: cat(aesthetic(aesthetic.color), map(("IL", color.blue), ("CA", color.green)))
```

Figure 221. Example: Mapping multiple categories to colors

Applies To
“cat Scale ” on page 29

marron Function

Syntax

```
marron()
```

Description

Uses the Marron adjustment to normalize the default fixed window across different kernel functions. Different kernel functions have different optimal windows. Therefore, normalizing the fixed window is useful when you need to compare the results of multiple kernel functions.

Examples

```
ELEMENT: line(position(density.kernel.epanechnikov(x, marron())))
```

Figure 222. Example: Adding a kernel distribution

Applies To
“density.kernel Function ” on page 98

max Function

Syntax

```
max(<numeric>)
```

<numeric>. A numeric value indicating the maximum scale value.

Description

Specifies a maximum value for a scale.

Examples

```
SCALE: linear(dim(2), max(50000))
```

Figure 223. Example: Specifying a maximum on the 2nd dimension (y axis)

Applies To
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“pow Scale ” on page 33
“safeLog Scale ” on page 35
“safePower Scale ” on page 36

“time Scale ” on page 37

min Function

Syntax

```
min(<numeric>)
```

<numeric>. A numeric value indicating the minimum scale value.

Description

Specifies a minimum value for a scale.

Examples

```
SCALE: linear(dim(2), min(0))
```

Figure 224. Example: Specifying a minimum on the 2nd dimension (y axis)

Applies To

“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“pow Scale ” on page 33
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

mirror Function

Syntax

```
mirror(<coord>)
```

<coord>. A valid coordinate type or transformation function. This is optional.

Description

Mirrors facets in the *x* axis dimension. This is useful for creating population pyramids.

Examples

```
COORD: transpose(mirror(rect(dim(1, 2))))
```

Figure 225. Example: Mirroring dimensions in a population pyramid

Coordinate Types and Transformations

“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“project Function ” on page 182
“reflect Function ” on page 183
“transpose Function ” on page 301
“wrap Function ” on page 305

Applies To

“COORD Statement ” on page 22
“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24

“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“project Function ” on page 182

missing.gap Function

Syntax

```
missing.gap()
```

Description

Specifies that the graphic element ends at a valid value and does not continue until the next valid value. There is a gap between valid values.

Examples

```
ELEMENT: line(position(x*y), missing.gap())
```

Figure 226. Example: Specifying gaps for missing values

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“line Element ” on page 47
“path Element ” on page 48

missing.interpolate Function

Syntax

```
missing.interpolate()
```

Description

Specifies that the graphic element is interpolated through missing values. That is, the graphic element is continuous from one valid value to another, regardless of missing values between the valid values.

Examples

```
ELEMENT: line(position(x*y), missing.interpolate())
```

Figure 227. Example: Interpolating through missing values

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“line Element ” on page 47
“path Element ” on page 48

missing.listwise Function

Syntax

```
missing.listwise()
```

Description

Specifies that a case is excluded from the graph if the case is missing a value for any variable in the dataset. It does not matter if the variable is actually used in the graph.

Examples

```
SOURCE: s = csvSource(file("mydata.csv"), missing.listwise())
```

Figure 228. Example: Excluding missing values listwise

Applies To

“csvSource Function ” on page 86
“sqlSource Function ” on page 227
“userSource Function ” on page 303

missing.pairwise Function

Syntax

```
missing.pairwise()
```

Description

Specifies that a case is excluded from the graph if the case is missing a value for any of the variables actually used in the graph. For example, if variables *x*, *y*, and *z* are used in the graph, a case is excluded only if it is missing a value for one of these variables. The missingness of other variables is not considered. `missing.pairwise` is the default behavior for handling missing values.

Examples

```
SOURCE: s = csvSource(file("mydata.csv"), missing.pairwise())
```

Figure 229. Example: Excluding missing values pairwise

Applies To

“csvSource Function ” on page 86
“sqlSource Function ” on page 227
“userSource Function ” on page 303

missing.wings Function

Syntax

```
missing.wings()
```

Description

Specifies that the graphic element continues after a valid value in the direction of the next valid value but then breaks just before and after the missing value. This is like interpolating through the missing value and erasing the graphic element at the missing value. For line charts, the result looks similar to wings.

Examples

```
ELEMENT: line(position(x*y), missing.wings())
```

Figure 230. Example: Specifying wings for missing values

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“line Element ” on page 47
“path Element ” on page 48

multiple Function

Syntax

```
multiple(<numeric>)
```

<numeric>. A positive numeric value.

Description

Specifies a multiplier for statistic functions.

Examples

```
ELEMENT: interval(position(region.spread.sd(x*y, multiple(2))), shape(shape.ibeam))
```

Figure 231. Example: Specifying 2 standard deviations

Applies To

“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198

noConstant Function

Syntax

```
noConstant()
```

Description

Specifies that no constant value is used in the smoother equation. Therefore, the smoother is calculated through the origin.

Examples

```
ELEMENT: line(position(smooth.linear(salbegin*salary, noConstant())))
```

Figure 232. Example: Creating a linear fit line through the origin

Applies To

“region.confi.smooth Function ” on page 192
“smooth.cubic Function ” on page 208
“smooth.linear Function ” on page 211
“smooth.quadratic Function ” on page 220

node Function

Syntax

```
node(<variable name>)
```

<variable name>. The name of a variable previously defined in the GPL by a DATA statement.

Description

Specifies the variable containing the unique nodes in the dataset.

Examples

```
ELEMENT: edge(position(layout.dag(node(id), from(fromVar), to(toVar))))
```

Figure 233. Example: Creating a directed acyclic graph

Applies To

“layout.circle Function ” on page 131
“layout.dag Function ” on page 133
“layout.data Function ” on page 135
“layout.grid Function ” on page 137
“layout.network Function ” on page 139
“layout.random Function ” on page 141
“layout.tree Function ” on page 143

notIn Function

Syntax

```
notIn("category name" ...)
```

<category name>. The string representing the category to be excluded. If specifying multiple categories, separate them with commas.

Description

Excludes the categories from the variable. These categories are not displayed or used in statistical calculations. This function is valid only for variables defined as categorical.

Examples

```
DATA: gender = col(source(mydata), name("gender"), unit.category(), notIn("Missing"))
```

Figure 234. Example: Excluding a category from a variable

Applies To

“col Function ” on page 79

normal Function

Syntax

```
normal(<mean>, <standard deviation>)
```

<mean>. Numeric value indicating the mean parameter for the distribution.

<standard deviation>. Numeric value indicating the standard deviation parameter for the distribution.

Description

Specifies a normal distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), normal(50000, 15000))
```

Figure 235. Example: Specifying a normal distribution for the probability scale

Applies To

“prob Scale ” on page 34

opposite Function

Syntax

```
opposite()
```

Description

Positions an axis on the side opposite from the one on which it normally appears. For example, using opposite with the *y* axis would position it on the right side. `opposite` can also be used to create two axes, in which case the opposite one is often an alternate scale or a transformed version of the original.

Examples

```
GUIDE: axis(dim(2), label("Count"), opposite())
```

Figure 236. Example: Moving the y-axis to the opposite side

```
GUIDE: axis(dim(2), label("Cumulative Count"))
GUIDE: axis(dim(2), label("Cumulative Percent"), opposite(), unit.percent())
```

Figure 237. Example: Adding a derived axis

Applies To

"axis Guide Type " on page 39

origin Function (For GPL Graphs)

Note: If you are modifying the origin for a scale, refer to "origin Function (For GPL Scales)" on page 179 .

Syntax

```
origin(<value>, <value>)
```

<value>. Indicates an absolute value or a percentage for the origin of the graph. The value is relative to the top left corner of the page and does not include axis labels. The first value indicates the *x* value relative to this position, and the second value indicates the *y* value relative to this position. Units or a percent sign can be included with the value (e.g., `30px`, `5cm`, or `25%`). If units are omitted, they are assumed to be pixels. Percentages are proportional to the whole page.

Description

Specifies the position of the graph relative to the top left corner of the page.

Examples

```
GRAPH: start(origin(2in, 4in))
```

Figure 238. Example: Positioning a graph with absolute units

```
GRAPH: start(origin(10%, 100%))
```

Figure 239. Example: Positioning a graph with percentages

Applies To

"begin Function (For GPL Graphs)" on page 65

origin Function (For GPL Scales)

Note: If you are modifying the origin for a graph, refer to "origin Function (For GPL Graphs)" on page 179 .

Syntax

```
origin(<numeric>)
```

<numeric>. A numeric value indicating the value of the scale's origin.

Description

Specifies the origin for a scale. The origin is typically used to specify a value from which area or interval graphic elements extend. The graphic elements originate at the origin and extend toward their value. For example, if your bar chart includes values of 367 and 48 and the origin is 100, one bar extends *up* from 100 to 367 (in default coordinates), while the other bar extends *down* to 48.

Examples

```
SCALE: linear(dim(2), origin(100))
```

Figure 240. Example: Specifying the origin

Applies To
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“pow Scale ” on page 33
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

poisson Function

Syntax

```
poisson(<rate>)
```

<rate>. Numeric value specifying the rate parameter for the distribution. This value must be greater than 0.

Description

Specifies a poisson distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), poisson(5.5))
```

Figure 241. Example: Specifying a poisson distribution for the probability scale

Applies To
“prob Scale ” on page 34

position Function (For GPL Graphic Elements)

Note: If you are specifying a position for a reference line (`form.line`), refer to “position Function (For GPL Guides)” on page 181 .

Syntax

```
position(<algebra>)
```

or

```
position(<binning function>)
```

or

```
position(<statistic function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. A statistic function.

Description

Specifies the position of the graphic element in the graph. When a statistic function is used in position, the statistic function is calculated on the second crossed variable in a 2-D coordinate system and the third crossed variable in a 3-D coordinate system.

Examples

```
ELEMENT: point(position(x*y))
```

Figure 242. Example: Scatterplot

```
ELEMENT: interval(position(summary.mean(x*y)))
```

Figure 243. Example: Bar chart of means

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

position Function (For GPL Guides)

Note: If you are specifying a position for a graphic element, refer to “position Function (For GPL Graphic Elements)” on page 180 .

Syntax

```
position(<x coordinate>, <y coordinate>)
```

<coordinate>. A numeric value or an asterisk (*) indicating the position of the line in relation to a particular axis. The asterisk is a wildcard character that represents all values on the associated axis. For example, `*,10` indicates a line at all *x*-axis values and at 10 on the *y* axis. In other words, these coordinates specify a horizontal reference line at 10.

Description

Specifies the position of a reference line (`form.line` guide).

Examples

```
GUIDE: form.line(position(*, 5000))
```

Figure 244. Example: Horizontal reference line

```
GUIDE: form.line(position(5000, *))
```

Figure 245. Example: Vertical reference line

Applies To
“form.line Guide Type ” on page 40

preserveStraightLines Function

Syntax

```
preserveStraightLines()
```

Description

Specifies that the graphic element is not curved in the space between points. Rather, the graphic element is drawn straight from point to point. This function is relevant only for graphic elements drawn in polar coordinates.

Examples

```
ELEMENT: line(position(x*y), preserveStraightLines())
```

Figure 246. Example: Drawing straight lines

Applies To
“area Element ” on page 44
“edge Element ” on page 45
“line Element ” on page 47
“path Element ” on page 48
“polygon Element ” on page 49

project Function

Syntax

```
project.<projection>()
```

<projection>. A valid projection name.

Description

Transforms the coordinate system using a map projection.

Examples

```
SOURCE: mapsrc = mapSource(file("World.smz"), layer("World"))
DATA: lon*lat = mapVariables(source(mapsrc))
COORD: project.mercator()
ELEMENT: polygon(position(lon*lat))
```

Figure 247. Example: Creating a map

Valid Projection Names

Valid projection names are: lambert, mercator, transverseMercator, winkelTripel

Applies To
“COORD Statement ” on page 22
“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26

proportion Function

Syntax

```
proportion(<numeric>)
```

<numeric>. A numeric value between 0 and 1.

Description

Specifies the proportion of data points to include when calculating the smooth function. This specifies the size of the window used for the smoother.

Examples

```
ELEMENT: line(position(smooth.loess(salbegin*salary, proportion(0.9))))
```

Figure 248. Example: Creating a loess fit line with specific smoother window

Applies To

“smooth.cubic Function ” on page 208
“smooth.linear Function ” on page 211
“smooth.loess Function ” on page 213
“smooth.mean Function ” on page 215
“smooth.median Function ” on page 217
“smooth.quadratic Function ” on page 220

reflect Function

Syntax

```
reflect(dim(<numeric>), <coord>)
```

<numeric>. A numeric value indicating the dimension across which the graph is reflected. See the topic “dim Function ” on page 115 for more information.

<coord>. A valid coordinate type or transformation function. This is optional.

Description

Reflects the coordinate system across the specified dimension.

Examples

```
COORD: rect(dim(1,2), reflect(dim(2)))
ELEMENT: interval(position(x*y))
```

Figure 249. Example: Creating an icicle plot

Coordinate Types and Transformations

“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“mirror Function ” on page 174
“project Function ” on page 182
“transpose Function ” on page 301
“wrap Function ” on page 305

Applies To

“COORD Statement ” on page 22
“parallel Coordinate Type ” on page 23

“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“project Function ” on page 182

region.confi.count Function

Syntax

```
region.confi.count(<algebra>, <function>)
```

or

```
region.confi.count(<statistic function>, <function>)
```

<algebra>. Graph algebra, such as x or x*y. In the second case, the confidence interval for the count is calculated for cases with non-missing y-variable values. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<function>. One or more valid functions. These are optional. If no `alpha` function is specified, 0.95 is used for the alpha.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `region.confi.count`.

Description

Calculates the confidence interval around the count. The function creates two values. When using the `interval`, `area`, or `edge` graphic elements, this function results in one graphic element showing the range of the confidence interval. All other graphic elements result in two separate elements, one showing the confidence interval below the count and one showing the confidence interval above the count.

Examples

```
ELEMENT: interval(position(region.confi.count(jobcat)), shape(shape.ibeam))
```

Figure 250. Example: Creating error bars

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions
“alpha Function ” on page 62
Applies To
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155

“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292

“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

region.confi.mean Function

Syntax

```
region.confi.mean(<algebra>, <function>)
```

or

```
region.confi.mean(<statistic function>, <function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<function>. One or more valid functions. These are optional. If no `alpha` function is specified, 0.95 is used for the alpha.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `region.confi.mean`.

Description

Calculates the confidence interval around the mean. The function creates two values. When using the `interval`, `area`, or `edge` graphic elements, this function results in one graphic element showing the range of the confidence interval. All other graphic elements result in two separate elements, one showing the confidence interval below the mean and one showing the confidence interval above the mean.

Examples

```
ELEMENT: interval(position(region.confi.mean(jobcat*salary)), shape(shape.ibeam))
```

Figure 251. Example: Creating error bars

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions

“alpha Function ” on page 62

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159

“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296

“transparency Function (For GPL Graphic Elements)” on page 300

region.confi.percent.count Function

Syntax

```
region.confi.percent.count(<algebra>, <function>, <base function>)
```

or

```
region.confi.percent.count(<statistic function>, <function>, <base function>)
```

<algebra>. Graph algebra, such as x or x*y. In the second case, the confidence interval for the percentage is calculated for cases with non-missing y-variable values. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<function>. One or more valid functions. These are optional. If no `alpha` function is specified, 0.95 is used for the alpha.

<base function>. A function that specifies the percentage base for `region.confi.percent.count`. This is optional. The default is `base.all()`.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `region.confi.percent.count`.

Description

Calculates the confidence interval around the percentage within each group compared to the total number of cases. The function creates two values. When using the `interval`, `area`, or `edge` graphic elements, this function results in one graphic element showing the range of the confidence interval. All other graphic elements result in two separate elements, one showing the confidence interval below the percentage value and one showing the confidence interval above the percentage value.

Examples

```
ELEMENT: interval(position(region.confi.percent.count(jobcat)), shape(shape.ibeam))
```

Figure 252. Example: Creating error bars

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions

“alpha Function ” on page 62

Base Functions

“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147

“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284

“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

region.confi.proportion.count Function

Syntax

```
region.confi.proportion.count(<algebra>, <function>, <base function>)
```

or

```
region.confi.proportion.count(<statistic function>, <function>, <base function>)
```

<algebra>. Graph algebra, such as x or x*y. In the second case, the confidence interval for the proportion is calculated for cases with non-missing y-variable values. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<function>. One or more valid functions. These are optional. If no `alpha` function is specified, 0.95 is used for the alpha.

<base function>. A function that specifies the percentage base for `region.confi.proportion.count`. This is optional. The default is `base.all()`.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `region.confi.proportion.count`.

Description

Calculates the confidence interval around the proportion within each group compared to the total number of cases. The function creates two values. When using the `interval`, `area`, or `edge` graphic elements, this function results in one graphic element showing the range of the confidence interval. All other graphic elements result in two separate elements, one showing the confidence interval below the proportion value and one showing the confidence interval above the proportion value.

Examples

```
ELEMENT: interval(position(region.confi.proportion.count(jobcat)), shape(shape.ibeam))
```

Figure 253. Example: Creating error bars

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions
“alpha Function ” on page 62
Base Functions
“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64
Applies To
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272

“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

region.confi.smooth Function

Syntax

```
region.confi.smooth.<smooth function>(<algebra>, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<smooth function>. The smooth function for which to calculate the confidence interval. Valid values are `cubic`, `linear`, and `quadratic`.

<function>. One or more valid functions. These are optional. If no `alpha` function is specified, 0.95 is used for the alpha.

Description

Calculates the confidence interval around a smoother function. The function creates two values. When using the `interval`, `area`, or `edge` graphic elements, this function results in one graphic element showing the range of the confidence interval. All other graphic elements result in two separate elements, one showing the confidence interval below the smoother function and one showing the confidence interval above the smoother function.

Examples

```
ELEMENT: line(position(region.confi.smooth.linear(salbegin*salary)))
```

Figure 254. Example: Showing the confidence interval around a fit line

Valid Functions

“alpha Function ” on page 62
“noConstant Function ” on page 177

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147

“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284

“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

region.spread.range Function

Syntax

```
region.spread.range(<algebra>)
```

or

```
region.spread.range(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `region.spread.range`.

Description

Calculates the minimum and maximum for a variable or variables identified by the algebra. That is, `region.spread.range` calculates the range of the variables. When using the `interval`, `area`, or `edge` graphic elements, this function results in one graphic element showing the range. All other graphic elements result in two separate elements, one showing the start of the range and one showing the end of the range.

Examples

```
ELEMENT: interval(position(region.spread.range(jobcat*salary)))
```

Figure 255. Example: Range bar showing minimum and maximum of one variable

```
ELEMENT: interval(position(region.spread.range(jobcat*(salbegin+salary))))
```

Figure 256. Example: Range bar showing minimum of one variable to maximum of another

Statistic Functions

See “GPL Functions” on page 53 .

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“density.beta Function ” on page 88
“density.chiSquare Function ” on page 90
“density.exponential Function ” on page 92
“density.f Function ” on page 94

“density.gamma Function ” on page 96
“density.kernel Function ” on page 98
“density.logistic Function ” on page 101
“density.normal Function ” on page 103
“density.poisson Function ” on page 105
“density.studentizedRange Function ” on page 107
“density.t Function ” on page 109
“density.uniform Function ” on page 111
“density.weibull Function ” on page 113
“layout.circle Function ” on page 131
“layout.dag Function ” on page 133
“layout.data Function ” on page 135
“layout.grid Function ” on page 137
“layout.network Function ” on page 139
“layout.random Function ” on page 141
“layout.tree Function ” on page 143
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248

“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

region.spread.sd Function

Syntax

```
region.spread.sd(<algebra>, <function>)
```

or

```
region.spread.sd(<binning function>, <function>)
```

or

```
region.spread.sd(<statistic function>, <function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `region.spread.sd`.

<function>. One or more valid functions. These are optional. If no `multiple` function is specified, 1 is used for the multiplier.

Description

Calculates the mean minus the standard deviation and the mean plus the standard deviation for the variables identified by the algebra. The function creates two values. When using the `interval`, `area`, or `edge` graphic elements, this function results in one graphic element showing the range of the standard deviation around the mean. All other graphic elements result in two separate elements, one showing the standard deviation below the mean and one showing the standard deviation above the mean.

Examples

```
ELEMENT: interval(position(region.spread.sd(jobcat*salary)))
```

Figure 257. Example: Creating an error bar

```
ELEMENT: interval(position(region.spread.sd(jobcat*salary, multiple(2))))
```

Figure 258. Example: Creating an error bar for 2 standard deviations

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions

“multiple Function ” on page 177

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227

“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

region.spread.se Function

Syntax

```
region.spread.se(<algebra>, <function>)
```

or

```
region.spread.se(<binning function>, <function>)
```

or

```
region.spread.se(<statistic function>, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `region.spread.se`.

<function>. One or more valid functions. These are optional. If no `multiple` function is specified, 1 is used for the multiplier.

Description

Calculates the mean minus the standard error and the mean plus the standard error for the variable identified by the algebra. The function creates two values. When using the `interval`, `area`, or `edge` graphic elements, this function results in one graphic element showing the range of the standard error around the mean. All other graphic elements result in two separate elements, one showing the standard error below the mean and one showing the standard error above the mean.

Examples

```
ELEMENT: interval(position(region.spread.se(jobcat*salary)))
```

Figure 259. Example: Creating an error bar

```
ELEMENT: interval(position(region.spread.se(jobcat*salary, multiple(2))))
```

Figure 260. Example: Creating an error bar for 2 standard errors

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions

“multiple Function ” on page 177

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170

“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

reverse Function

Syntax

```
reverse()
```

Description

When used in conjunction with a scale, this function reverses the scale. For categorical scales, this function can be used in conjunction with an explicit sorting function.

When used in conjunction with a polar coordinate system, this function reverses the direction of the coordinate system. Thus, it would draw pie slices in a counterclockwise direction.

Examples

```
SCALE: cat(dim(1), sort.natural(), reverse())
```

Figure 261. Example: Using reverse alpha-numeric sorting

```
SCALE: linear(dim(2), reverse())
```

Figure 262. Example: Reversing a linear scale

```
SCALE: linear(aesthetic(aesthetic.color.brightness), reverse())
```

Figure 263. Example: Reversing the direction of an aesthetic scale

```
COORD: polar.theta(reverse())
```

Figure 264. Example: Reversing the direction of a coordinate system

Applies To

“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“cat Scale ” on page 29
“cLoglog Scale ” on page 30
“linear Scale ” on page 31
“log Scale ” on page 32
“pow Scale ” on page 33
“safeLog Scale ” on page 35
“safePower Scale ” on page 36
“time Scale ” on page 37

root Function

Syntax

```
root("variable value")
```

"variable value". A variable value.

Description

Specifies which node is the root node. This corresponds to a value for the variable identified by the node function.

Examples

```
ELEMENT: edge(position(layout.tree(node(id), from(fromVar), to(toVar), root("A"))))
```

Figure 265. Example: Specifying a root node

Applies To

“layout.tree Function ” on page 143

sameRatio Function

Syntax

```
sameRatio()
```

Description

Specifies that the same distance on each scale in a rectangular coordinate system represents the same difference in data values. For example, 2cm on both scales represent a difference of 1000.

Examples

```
COORD: rect(dim(1,2), sameRatio())
ELEMENT: point(position(salbegin*salary))
```

Figure 266. Example: Creating a scatterplot with equal scales

Applies To

“rect Coordinate Type ” on page 26

savSource Function

Syntax

```
savSource(file("file path"))
```

Note: This function works with the IBM SPSS Statistics data file driver. You must have the driver installed and configured before using the function. The driver is available with the installation media for your product. You can also use the `userSource` function to read IBM SPSS Statistics data directly. See the topic “userSource Function ” on page 303 for more information.

"file path". The absolute path to the SAV file. To ensure that the file is located correctly, this should be an absolute path. Backslashes must be escaped with another backslash (for example, `C:\\data\\myfile.csv`). You can also use forward slashes (for example, `C:/data/myfile.csv`).

Description

Reads the contents of a SAV data file for IBM SPSS Statistics. This function is used to assign the contents of the file to a data source.

Examples

```
SOURCE: mydata = savSource(file("/Data/Employee data.sav"))
```

Figure 267. Example: Reading a SAV file

Applies To

“SOURCE Statement ” on page 21

scale Function (For GPL Axes)

Note: If you are specifying a size for a graph, refer to “scale Function (For GPL Graphs)” on page 203 . If you are specifying a scale associated with a graphic element (like a bar or line) or a reference line (`form.line`), refer to “scale Function (For GPL Graphic Elements and form.line)” on page 203 . If you are specifying a size for a page, refer to “scale Function (For GPL Pages)” on page 204 .

Syntax

```
scale(<scale name>)
```

<scale name>. A scale previously defined by a SCALE statement. This is used when there are multiple scales in a single dimension (as in a "dual axis" graph).

Description

Specifies the scale to which an axis applies.

Examples

```
SCALE: y2= linear(dim(2))
GUIDE: axis(scale(y2), label("Count"))
```

Figure 268. Example: Associating an axis with a named scale

Applies To

"axis Guide Type " on page 39

scale Function (For GPL Graphs)

Note: If you are specifying a scale associated with an axis, refer to "scale Function (For GPL Axes)" on page 202 . If you are specifying a scale associated with a graphic element (like a bar or line) or a reference line (`form.line`), refer to "scale Function (For GPL Graphic Elements and form.line)" on page 203 . If you are specifying a size for a page, refer to "scale Function (For GPL Pages)" on page 204 .

Syntax

```
scale(<value>, <value>)
```

<value>. Indicates an absolute value or a percentage for the graph size. The first value indicates the *x* component of the size (width), and the second value indicates the *y* component of the size (height). Units or a percent sign can be included with either value (e.g., `30px`, `5cm`, or `25%`). If units are omitted, they are assumed to be pixels. Percentages are proportional to the whole page.

Description

Specifies the size of the data area of a graph, not including axes and legends.

Examples

```
GRAPH: begin(scale(2in, 4in))
```

Figure 269. Example: Sizing a graph with absolute units

```
GRAPH: begin(scale(80%, 100%))
```

Figure 270. Example: Sizing a graph with percentages

Applies To

"begin Function (For GPL Graphs)" on page 65

scale Function (For GPL Graphic Elements and form.line)

Note: If you are specifying a scale associated with an axis, refer to "scale Function (For GPL Axes)" on page 202 . If you are specifying a size for a graph, refer to "scale Function (For GPL Graphs)" on page 203 . If you are specifying a size for a page, refer to "scale Function (For GPL Pages)" on page 204 .

Syntax

```
scale(<scale name> ...)
```

<scale name>. A scale previously defined by a SCALE statement. This is used when there are multiple scales in a single dimension (as in a "dual-axis" graph). You can specify multiple scales if the scales are associated with different dimensions. Use commas to separate the multiple scales.

Description

Specifies the scale to which a graphic element or reference line (`form.line`) applies.

Examples

```
SCALE: y2= linear(dim(2))
ELEMENT: line(scale(y2), position(summary.count(x)))
```

Figure 271. Example: Associating a graphic element with a named scale

Applies To

“form.line Guide Type ” on page 40
“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

scale Function (For GPL Pages)

Note: If you are specifying a scale associated with an axis, refer to “scale Function (For GPL Axes)” on page 202 . If you are specifying a scale associated with a graphic element (like a bar or line) or a reference line (`form.line`), refer to “scale Function (For GPL Graphic Elements and form.line)” on page 203 . If you are specifying a size for a graph, refer to “scale Function (For GPL Graphs)” on page 203 .

Syntax

```
scale(<value>, <value>)
```

<value>. Indicates an absolute value for the page size. The first value indicates the *x* component of the size (width), and the second value indicates the *y* component of the size (height). Units can be included with either value (e.g., `600px`, `15cm`, or `5in`). If units are omitted, they are assumed to be pixels.

Description

Specifies the size of the graph.

Examples

```
PAGE: begin(scale(500px, 400px))
```

Figure 272. Example: Sizing a visualization

Applies To

“begin Function (For GPL Pages)” on page 65

scaledToData Function

Syntax

```
scaledToData("true")
```

or

```
scaledToData("false")
```

Description

Indicates whether a density function is scaled to the number of cases in the data being displayed. By default, density functions are scaled to the data. This is appropriate when a graph contains both a

histogram and a density function, because you want to be able to compare the histogram and the density function on the same scale. However, if you want to compare density functions across multiple graphs displaying very different sample sizes, you may want to use `scaledToData("false")`.

Examples

```
ELEMENT: line(position(density.normal(salary, scaledToData("false"))), color(color.black))
ELEMENT: line(position(density.normal(salary_subset, scaledToData("false"))), color(color.blue))
```

Figure 273. Example: Comparing different densities on the same scale

Applies To

“density.beta Function ” on page 88
“density.chiSquare Function ” on page 90
“density.exponential Function ” on page 92
“density.f Function ” on page 94
“density.gamma Function ” on page 96
“density.kernel Function ” on page 98
“density.logistic Function ” on page 101
“density.normal Function ” on page 103
“density.poisson Function ” on page 105
“density.studentizedRange Function ” on page 107
“density.t Function ” on page 109
“density.uniform Function ” on page 111
“density.weibull Function ” on page 113

segments Function

Syntax

```
segments(<integer>)
```

<integer>. A positive integer.

Description

Specifies the number of segments that are calculated and drawn for the density function. Excluding this function will result in a default number of segments, which should be sufficient for most cases.

Examples

```
ELEMENT: line(position(density.kernel.epanechnikov(x, segments(150))))
```

Figure 274. Example: Adding a kernel distribution

Applies To

“density.kernel Function ” on page 98

shape Function (For GPL Graphic Elements)

Note: If you are modifying the shape for a guide, refer to “shape Function (For GPL Guides)” on page 206 .

Syntax

```
shape(<algebra>)
```

or

```
shape(shape.<constant>)
```

<algebra>. Graph algebra using one categorical variable or a blend of categorical variables.

<constant>. A constant indicating a specific shape, such as `shape.square`. See the topic “Shape Constants ” on page 385 for more information.

Description

Controls the shape of a graphic element. What `shape` controls depends on the graphic element type. The shape of a line specifies whether the line is solid or dashed. The border around a bar has a similar shape. The shape of a point or interval specifies whether the point or interval is shaped like a square or a line. All of these shapes are controlled by the `shape` function, but you can append `.interior` or `.exterior` to the function to ensure that you are specifying the desired one. `shape.interior` specifies the overall shape of the graphic element, including the dashing of edge, line, and path elements. `shape.exterior` specifies the shape of the exterior of the graphic element, which is often the border on graphic elements with fills. `shape.exterior` does not apply to edge, line, and path elements. Using `shape` without a qualifier implies `shape.interior`.

Examples

```
ELEMENT: point(position(salbegin*salary), shape.interior(shape.square))
```

Figure 275. Example: Specifying a shape value

```
ELEMENT: point(position(salbegin*salary), shape.interior(jobcat))
```

Figure 276. Example: Using the values of a variable to control shape

```
ELEMENT: line(position(salbegin*salary), shape.interior(jobcat))
```

Figure 277. Example: Using the values of a variable to control line dashing

```
ELEMENT: interval(position(gender*salary*jobcat), shape.exterior(gender))
```

Figure 278. Example: Using the values of a variable to control border dashing

Applies To

“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49

shape Function (For GPL Guides)

Note: If you are modifying the shape for a graphic element (like a bar or point), refer to “shape Function (For GPL Graphic Elements)” on page 205 .

Syntax

```
shape(shape.<shape constant>)
```

<shape constant>. A constant indicating a specific shape, such as `shape.dash`. See the topic “Shape Constants ” on page 385 for more information.

Description

Controls the dashing of reference lines.

Examples

```
GUIDE: form.line(position(*,2000), shape(shape.dash))
```

Figure 279. Example: Specifying a dashed reference line

Applies To

“form.line Guide Type ” on page 40

showAll Function

Syntax

```
showAll()
```

Description

Display all labels, even if they overlap. Without this function, some overlapping labels may not be displayed, depending on the available space.

Examples

```
ELEMENT: point(position(x*y), label(z, showAll()))
```

Figure 280. Example: Displaying all labels

Applies To

“label Function (For GPL Graphic Elements)” on page 129

size Function (For GPL Graphic Elements)

Note: If you are modifying the size for a guide, refer to “size Function (For GPL Guides)” on page 208 .

Syntax

```
size(<algebra>)
```

or

```
size(size."size value")
```

or

```
size(size.<constant>)
```

or

```
size(<statistic function>)
```

<algebra>. Graph algebra using one variable or a blend of variables. This is not available for line elements.

"size value". A specific value that indicates a size. This can be a percentage of the available space (for example, 40%) or a number with units (for example, 2in).

<constant>. A size constant. See the topic “Size Constants ” on page 385 for more information.

<statistic function>. A statistic function.

Description

Specifies the size of the individual graphic elements.

Examples

```
ELEMENT: point(position(x*y), size(z))
```

Figure 281. Example: Using a variable to control size

```
ELEMENT: interval(position(x*y), size(size."60%"))
```

Figure 282. Example: Specifying a percentage for size

```
ELEMENT: interval(position(x*y), size(size."6px"))
```

Figure 283. Example: Specifying a value for size

```
ELEMENT: interval(position(x*y), size(size.large))
```

Figure 284. Example: Specifying a constant for size

Statistic Functions

See “GPL Functions” on page 53 .

Applies To

“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

size Function (For GPL Guides)

Note: If you are modifying the size for a graphic element (like a bar or point), refer to “size Function (For GPL Graphic Elements)” on page 207 .

Syntax

```
size(size."size value")
```

or

```
size(size.<constant>)
```

"size value". A specific value that indicates a size. This can be a percentage of the available space (for example, 40%) or a number with units (for example, 2in).

<constant>. A size constant. See the topic “Size Constants ” on page 385 for more information.

Description

Controls the thickness of reference lines.

Examples

```
GUIDE: form.line(position(*,1000), size(size."15px"))
```

Figure 285. Example: Specifying a reference line with a thickness of 15 pixels

Applies To

“form.line Guide Type ” on page 40

smooth.cubic Function

Syntax

```
smooth.cubic(<algebra>, <function>)
```

or

```
smooth.cubic.<kernel>(<algebra, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<kernel>. A kernel function for the smoother. This specifies how data are weighted by the smoother, depending on how close the data are to the current point. If no kernel is specified, Epanechnikov is used.

<function>. One or more valid functions. These are optional. If no `proportion` function is specified, 1 is used for the proportion.

Description

Uses regression to determine values that best fit the data to a cubic polynomial.

Examples

```
ELEMENT: line(position(smooth.cubic(salbegin*salary)))
```

Figure 286. Example: Creating a cubic fit line

Kernel Functions

uniform. All data receive equal weights.

epanechnikov. Data near the current point receive higher weights than extreme data receive. This function weights extreme points more than the triweight, biweight, and tricube kernels but less than the Gaussian and Cauchy kernels.

biweight. Data far from the current point receive more weight than the triweight kernel allows but less weight than the Epanechnikov kernel permits.

tricube. Data close to the current point receive higher weights than both the Epanechnikov and biweight kernels allow.

triweight. Data close to the current point receive higher weights than any other kernel allows. Extreme cases get very little weight.

gaussian. Weights follow a normal distribution, resulting in higher weighting of extreme cases than the Epanechnikov, biweight, tricube, and triweight kernels.

cauchy. Extreme values receive more weight than the other kernels, with the exception of the uniform kernel, allow.

Valid Functions

“noConstant Function ” on page 177
“proportion Function ” on page 183

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159

“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294

“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

smooth.linear Function

Syntax

```
smooth.linear(<algebra>, <function>)
```

or

```
smooth.linear.<kernel>(<algebra, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<kernel>. A kernel function for the smoother. This specifies how data are weighted by the smoother, depending on how close the data are to the current point. If no kernel is specified, Epanechnikov is used.

<function>. One or more valid functions. These are optional. If no `proportion` function is specified, 1 is used for the proportion.

Description

Uses regression to determine values that best fit the data to a linear slope.

Examples

```
ELEMENT: line(position(smooth.linear(salbegin*salary)))
```

Figure 287. Example: Creating a linear fit line

Kernel Functions

uniform. All data receive equal weights.

epanechnikov. Data near the current point receive higher weights than extreme data receive. This function weights extreme points more than the triweight, biweight, and tricube kernels but less than the Gaussian and Cauchy kernels.

biweight. Data far from the current point receive more weight than the triweight kernel allows but less weight than the Epanechnikov kernel permits.

tricube. Data close to the current point receive higher weights than both the Epanechnikov and biweight kernels allow.

triweight. Data close to the current point receive higher weights than any other kernel allows. Extreme cases get very little weight.

gaussian. Weights follow a normal distribution, resulting in higher weighting of extreme cases than the Epanechnikov, biweight, tricube, and triweight kernels.

cauchy. Extreme values receive more weight than the other kernels, with the exception of the uniform kernel, allow.

Valid Functions

“noConstant Function ” on page 177
“proportion Function ” on page 183

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81

“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272

“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

smooth.loess Function

Syntax

```
smooth.loess(<algebra>, <function>)
```

or

```
smooth.loess.<kernel>(<algebra>, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<kernel>. A kernel function for the smoother. This specifies how data are weighted by the smoother, depending on how close the data are to the current point. If no kernel is specified, tricube is used.

<function>. One or more valid functions. These are optional. If no `proportion` function is specified, 1 is used for the proportion.

Description

Uses iterative weighted least squares to determine values that best fit the data.

Examples

```
ELEMENT: line(position(smooth.loess(salbegin*salary)))
```

Figure 288. Example: Creating a loess fit line

```
ELEMENT: line(position(smooth.loess.uniform(salbegin*salary)))
```

Figure 289. Example: Creating a loess fit line with specific kernel

Kernel Functions

uniform. All data receive equal weights.

epanechnikov. Data near the current point receive higher weights than extreme data receive. This function weights extreme points more than the triweight, biweight, and tricube kernels but less than the Gaussian and Cauchy kernels.

biweight. Data far from the current point receive more weight than the triweight kernel allows but less weight than the Epanechnikov kernel permits.

tricube. Data close to the current point receive higher weights than both the Epanechnikov and biweight kernels allow.

triweight. Data close to the current point receive higher weights than any other kernel allows. Extreme cases get very little weight.

gaussian. Weights follow a normal distribution, resulting in higher weighting of extreme cases than the Epanechnikov, biweight, tricube, and triweight kernels.

cauchy. Extreme values receive more weight than the other kernels, with the exception of the uniform kernel, allow.

Valid Functions

“proportion Function ” on page 183

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244

“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

smooth.mean Function

Syntax

```
smooth.mean(<algebra>, <function>)
```

or

```
smooth.mean.<kernel>(<algebra, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<kernel>. A kernel function for the smoother. This specifies how data are weighted by the smoother, depending on how close the data are to the current point. If no kernel is specified, Epanechnikov is used.

<function>. One or more valid functions. These are optional. If no `proportion` function is specified, 1 is used for the proportion.

Description

Calculates the smoothed mean of *y* in a 2-D (*x*y*) frame and *z* in a 3-D (*x*y*z*) frame. To force a straight line (a constant value), use the uniform kernel function.

Examples

```
ELEMENT: line(position(smooth.mean.uniform(salbegin*salary)))
```

Figure 290. Example: Creating a line at the mean of the y axis

Kernel Functions

uniform. All data receive equal weights.

epanechnikov. Data near the current point receive higher weights than extreme data receive. This function weights extreme points more than the triweight, biweight, and tricube kernels but less than the Gaussian and Cauchy kernels.

biweight. Data far from the current point receive more weight than the triweight kernel allows but less weight than the Epanechnikov kernel permits.

tricube. Data close to the current point receive higher weights than both the Epanechnikov and biweight kernels allow.

triweight. Data close to the current point receive higher weights than any other kernel allows. Extreme cases get very little weight.

gaussian. Weights follow a normal distribution, resulting in higher weighting of extreme cases than the Epanechnikov, biweight, tricube, and triweight kernels.

cauchy. Extreme values receive more weight than the other kernels, with the exception of the uniform kernel, allow.

Valid Functions

“proportion Function ” on page 183

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198

“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

smooth.median Function

Syntax

```
smooth.median(<algebra>, <function>)
```

or

```
smooth.median.<kernel>(<algebra, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<kernel>. A kernel function for the smoother. This specifies how data are weighted by the smoother, depending on how close the data are to the current point. If no kernel is specified, Epanechnikov is used.

<function>. One or more valid functions. These are optional. If no `proportion` function is specified, 1 is used for the proportion.

Description

Calculates the smoothed median of *y* in a 2-D (*x*y*) frame and *z* in a 3-D (*x*y*z*) frame. To force a straight line (a constant value), use the uniform kernel function.

Examples

```
ELEMENT: line(position(smooth.median.uniform(salbegin*salary)))
```

Figure 291. Example: Creating a line at the median of the y axis

Kernel Functions

uniform. All data receive equal weights.

epanechnikov. Data near the current point receive higher weights than extreme data receive. This function weights extreme points more than the triweight, biweight, and tricube kernels but less than the Gaussian and Cauchy kernels.

biweight. Data far from the current point receive more weight than the triweight kernel allows but less weight than the Epanechnikov kernel permits.

tricube. Data close to the current point receive higher weights than both the Epanechnikov and biweight kernels allow.

triweight. Data close to the current point receive higher weights than any other kernel allows. Extreme cases get very little weight.

gaussian. Weights follow a normal distribution, resulting in higher weighting of extreme cases than the Epanechnikov, biweight, tricube, and triweight kernels.

cauchy. Extreme values receive more weight than the other kernels, with the exception of the uniform kernel, allow.

Valid Functions

“proportion Function ” on page 183
“smooth.cubic Function ” on page 208
“smooth.linear Function ” on page 211
“smooth.loess Function ” on page 213
“smooth.mean Function ” on page 215
“smooth.quadratic Function ” on page 220

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157

“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292

“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

smooth.quadratic Function

Syntax

```
smooth.quadratic(<algebra>, <function>)
```

or

```
smooth.quadratic.<kernel>(<algebra, <function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<kernel>. A kernel function for the smoother. This specifies how data are weighted by the smoother, depending on how close the data are to the current point. If no kernel is specified, Epanechnikov is used.

<function>. One or more valid functions. These are optional. If no `proportion` function is specified, 1 is used for the proportion.

Description

Uses regression to determine values that best fit the data to a quadratic polynomial.

Examples

```
ELEMENT: line(position(smooth.quadratic(salbegin*salary)))
```

Figure 292. Example: Creating a quadratic fit line

Kernel Functions

uniform. All data receive equal weights.

epanechnikov. Data near the current point receive higher weights than extreme data receive. This function weights extreme points more than the triweight, biweight, and tricube kernels but less than the Gaussian and Cauchy kernels.

biweight. Data far from the current point receive more weight than the triweight kernel allows but less weight than the Epanechnikov kernel permits.

tricube. Data close to the current point receive higher weights than both the Epanechnikov and biweight kernels allow.

triweight. Data close to the current point receive higher weights than any other kernel allows. Extreme cases get very little weight.

gaussian. Weights follow a normal distribution, resulting in higher weighting of extreme cases than the Epanechnikov, biweight, tricube, and triweight kernels.

cauchy. Extreme values receive more weight than the other kernels, with the exception of the uniform kernel, allow.

Valid Functions

“noConstant Function ” on page 177
“proportion Function ” on page 183

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270

“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

smooth.spline Function

Syntax

```
smooth.spline(<algebra>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

Description

Calculates the cubic spline for the data. You should use a cubic spline curve only when you believe there is no error in your data.

Examples

```
ELEMENT: line(position(smooth.spline(salbegin*salary)))
```

Figure 293. Example: Creating a cubic spline curve

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166

“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

smooth.step Function

Syntax

```
smooth.step(<algebra>)
```

or

```
smooth.step.<position>(<algebra>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to "Brief Overview of GPL Algebra" on page 3 for an introduction to graph algebra.

<position>. The position of the data value in relation to the drawn line. Valid values are `right`, `left`, and `center`, with `right` being the default.

Description

Uses step interpolation to draw the graphic element through the data values. Use the `jump` function to specify that no connecting lines are drawn between the lines at each data value.

Examples

```
ELEMENT: line(position(smooth.step.center(salbegin*salary)))
```

Figure 294. Example: Creating a step interpolation line

Applies To

"bin.dot Function " on page 66
"bin.hex Function " on page 68
"bin.quantile.letter Function " on page 70
"bin.rect Function " on page 72
"color Function (For GPL Graphic Elements)" on page 81
"color.brightness Function (For GPL Graphic Elements)" on page 82
"color.hue Function (For GPL Graphic Elements)" on page 84
"color.saturation Function (For GPL Graphic Elements)" on page 85
"link.alpha Function " on page 145
"link.complete Function " on page 147
"link.delaunay Function " on page 149
"link.distance Function " on page 151
"link.gabriel Function " on page 153
"link.hull Function " on page 155
"link.influence Function " on page 157
"link.join Function " on page 159
"link.mst Function " on page 162
"link.neighbor Function " on page 164
"link.relativeNeighborhood Function " on page 166
"link.sequence Function " on page 168
"link.tsp Function " on page 170
"position Function (For GPL Graphic Elements)" on page 180
"region.confi.count Function " on page 184
"region.confi.mean Function " on page 186
"region.confi.percent.count Function " on page 188
"region.confi.proportion.count Function " on page 190
"region.confi.smooth Function " on page 192
"region.spread.range Function " on page 194

“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

sort.data Function

Syntax

```
sort.data()
```

Description

Sorts the categorical values in the order they appear in the data.

Note: If the data are pre-aggregated (for example, by using a statistic function in the IBM SPSS Statistics GGRAPH command), this function will not work as expected because the categorical values may be sorted

during aggregation. Therefore, the GPL no longer knows the order in which the categorical values appeared in the original data.

Examples

```
SCALE: cat(dim(1), sort.data())
```

Figure 295. Example: Sorting the categories

Applies To
“cat Scale ” on page 29

sort.natural Function

Syntax

```
sort.natural()
```

Description

Sorts the categorical values in alphanumeric order.

Examples

```
SCALE: cat(dim(1), values("Male", "Female"), sort.natural())
```

Figure 296. Example: Sorting the categories

Applies To
“cat Scale ” on page 29

sort.statistic Function

Syntax

```
sort.statistic(<statistic function>)
```

<statistic function>. A statistic function.

Description

Sorts the categorical values based on the result of the statistic function for each category.

Examples

```
SCALE: cat(dim(1), sort.statistic(summary.mean(salary)))
```

Figure 297. Example: Sorting the categories

Statistic Functions

See “GPL Functions” on page 53 .

Applies To
“cat Scale ” on page 29

sort.values Function

Syntax

```
sort.values("category name" ...)
```

"category name". The name of a category in the data. Delineate each name with a comma.

Description

Sorts the categorical values based on the order in which they appear in this function. You do not need to specify every category. The categories will be ordered as they appear, and any other categories that are not specified will appear in the order they appear in the data. If you are using this function with a data file from IBM SPSS Statistics, you need to specify the data values for the categories, not the data labels that appear in the resulting visualization.

Examples

```
SCALE: cat(dim(1), sort.values("Male"))
```

Figure 298. Example: Sorting the categories explicitly

In this example, *Male* will appear first. It is not necessary to specify that *Female* will appear next.

Applies To
“cat Scale ” on page 29

split Function

Syntax

```
split(<algebra>)
```

or

```
color(<statistic function>)
```

<algebra>. The name of a categorical variable.

<statistic function>. A statistic function.

Description

Splits the graphic element into multiple graphic elements or groups of graphic elements for each category in a categorical variable. This result is similar to that obtained by the aesthetic functions, but there is no specific aesthetic associated with each group of graphic elements.

Examples

```
ELEMENT: line(position(salbegin*salary), split(gender))
```

Figure 299. Example: Creating groups of lines

Statistic Functions

See “GPL Functions” on page 53 .

Applies To
“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

sqlSource Function

Syntax

```
sqlSource(url("url"), user("user name"), password("user password"), query("sql query"))
```

"url". The URL string for connecting to the database. The URL is typically of the form *jdbc:<jdbc vendor>:<vendor information>://<host name>:<port number>*, where items between angled brackets (<>) are variables specific to each database driver vendor. Consult your database vendor's documentation for more information.

"user name". A user name for accessing the database.

"user password". The user's password.

"sql query". A SQL query string for extracting data from the database.

Description

Reads data from a database using a SQL query string.

Examples

```
SOURCE: mydata = sqlSource(url("jdbc:microsoft:sqlserver://localhost:1433"), user("fred"),
   password("secret"), query("select * from employeeData"))
```

Figure 300. Example: Reading from a database

Valid Functions

“missing.listwise Function ” on page 175
“missing.pairwise Function ” on page 176
“weight Function ” on page 305

Applies To

“SOURCE Statement ” on page 21

start Function

Syntax

```
start(<value>)
```

<value>. A numeric value indicating the location of the first major tick.

Description

Specifies the value at which the first major tick appears.

Examples

```
GUIDE: axis(dim(1), start(1000))
```

Figure 301. Example: Specifying the first major tick

Applies To

“axis Guide Type ” on page 39

startAngle Function

Syntax

```
startAngle(<integer>)
```

<integer>. An integer indicating the number of degrees relative to 12:00.

Description

Indicates the angle at which the coordinate system begins. Often used to indicate the position of the first slice in a pie chart. The specified degrees are relative to the 12:00 position, and rotation is counterclockwise.

Examples

```
COORD: polar.theta(startAngle(90))
```

Figure 302. Example: Specifying the first slice at 9:00

```
COORD: polar.theta(startAngle(-90))
```

Figure 303. Example: Specifying the first slice at 3:00 (90 degrees)

Applies To

“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25

studentizedRange Function

Syntax

```
studentizedRange(<degrees of freedom>, <k>)
```

<degrees of freedom>. Numeric value indicating the degrees of freedom parameter for the distribution. The value must be greater than 0.

<k>. Numeric value indicating the k (number of groups) parameter for the distribution. The value must be greater than 0.

Description

Specifies a Studentized range distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), studentizedRange(5, 2.5))
```

Figure 304. Example: Specifying a Studentized range distribution for the probability scale

Applies To

“prob Scale ” on page 34

summary.count Function

Syntax

```
summary.count(<algebra>)
```

or

```
summary.count(<binning function>)
```

or

```
summary.count(<statistic function>)
```

<algebra>. Graph algebra, such as x or x*y. In the second case, the count is calculated for cases with non-missing y-variable values. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.count`.

Description

Calculates the number of cases identified by the algebra or function. If using a function, a typical one would be a binning function. `summary.count` subsequently calculates the number of cases in each bin.

Examples

```
ELEMENT: interval(position(summary.count(jobcat)))
```

Figure 305. Example: Specifying a bar chart of counts

```
ELEMENT: interval(position(summary.count(jobcat*salary)))
```

Figure 306. Example: Counting non-missing cases for a continuous variable

```
ELEMENT: interval(position(summary.count(bin.rect(salary))))
```

Figure 307. Example: Specifying a histogram

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198

“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.count.cumulative Function

Syntax

```
summary.count.cumulative(<algebra>)
```

or

```
summary.count.cumulative(<binning function>)
```

or

```
summary.count.cumulative(<statistic function>)
```

<algebra>. Graph algebra, such as x or x*y. In the second case, the count is calculated for cases with non-missing y-variable values. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.count.cumulative`.

Description

Calculates the cumulative number of cases identified by the algebra or function. If using a function, a typical one would be a binning function. `summary.count` subsequently calculates the number of cases in each bin.

If the graph is paneled (faceted), the cumulation begins again with each panel.

Note: If there are multiple ELEMENT statements, you cannot use cumulative statistics for some graphic elements but not for others. This behavior is prohibited because the results of each statistic function would be blended on the same scale. The units for cumulative statistics do not match the units for non-cumulative statistics, so blending these results is impossible.

Examples

```
ELEMENT: interval(position(summary.count.cumulative(jobcat)))
```

Figure 308. Example: Specifying a bar chart of cumulative counts

```
ELEMENT: interval(position(summary.count.cumulative(bin.rect(salary))))
```

Figure 309. Example: Specifying a cumulative histogram

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162

“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.countTrue Function

Syntax

```
summary.countTrue(<algebra>)
```

or

```
summary.countTrue(<binning function>)
```

or

```
summary.countTrue(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.countTrue`.

Description

Calculates the number of cases that evaluate to a *true* value. If the function is evaluating graph algebra, the analysis variable is typically the Boolean result of expression evaluated by the `eval` function. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
TRANS: salGreaterThan = eval(salary>50000)
ELEMENT: interval(position(summary.countTrue(jobcat*salGreaterThan)))
```

Figure 310. Example: Plotting count greater than a value

```
TRANS: salLessThan = eval(salary<50000)
TRANS: salEqualTo = eval(salary==50000)
TRANS: salGreaterThan = eval(salary>50000)
ELEMENT: interval(position(summary.countTrue("Less than 50000"*salLessThan)))
ELEMENT: interval(position(summary.countTrue("Equal to 50000"*salEqualTo)))
ELEMENT: interval(position(summary.countTrue("Greater than 50000"*salGreaterThan)))
```

Figure 311. Example: Plotting count less than, equal to, and greater than a value

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145

“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282

“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.first Function

Syntax

```
summary.first(<algebra>)
```

or

```
summary.first(<binning function>)
```

or

```
summary.first(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.max`.

Description

Gets the first value that appears in the data. If the function is evaluating graph algebra, the first value of the analysis variable is returned for each subgroup. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.first(jobcat*salary)))
```

Figure 312. Example: Calculating the first salary value for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84

“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277

“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.kurtosis Function

Syntax

```
summary.kurtosis(<algebra>)
```

or

```
summary.kurtosis(<binning function>)
```

or

```
summary.kurtosis(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.kurtosis`.

Description

Calculates the kurtosis, which measures whether the data peak more compared to the normal distribution. If the function is evaluating graph algebra, the kurtosis of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.kurtosis(jobcat*salary)))
```

Figure 313. Example: Calculating the kurtosis of salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272

“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.last Function

Syntax

```
summary.last(<algebra>)
```

or

```
summary.last(<binning function>)
```

or

```
summary.last(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.max`.

Description

Gets the last value that appears in the data. If the function is evaluating graph algebra, the last value of the analysis variable is returned for each subgroup. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.last(jobcat*salary)))
```

Figure 314. Example: Calculating the last salary value for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
Applies To
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68

“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269

“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.max Function

Syntax

```
summary.max(<algebra>)
```

or

```
summary.max(<binning function>)
```

or

```
summary.max(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.max`.

Description

Calculates the maximum value. If the function is evaluating graph algebra, the maximum value of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.max(jobcat*salary)))
```

Figure 315. Example: Calculating the maximum salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262

“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.mean Function

Syntax

```
summary.mean(<algebra>)
```

or

```
summary.mean(<binning function>)
```

or

```
summary.mean(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.mean`.

Description

Calculates the arithmetic mean. If the function is evaluating graph algebra, the mean of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.mean(jobcat*salary)))
```

Figure 316. Example: Calculating the mean salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68

“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258

“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.median Function

Syntax

```
summary.median(<algebra>)
```

or

```
summary.median(<binning function>)
```

or

```
summary.median(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.median`.

Description

Calculates the median, which is the value above and below which half of the cases fall. The result is equivalent to `summary.percentile` with an alpha of 0.5. If the function is evaluating graph algebra, the median of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.median(jobcat*salary)))
```

Figure 317. Example: Calculating the median salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252

“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.min Function

Syntax

```
summary.min(<algebra>)
```

or

```
summary.min(<binning function>)
```

or

```
summary.min(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.min`.

Description

Calculates the minimum value. If the function is evaluating graph algebra, the minimum value of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.min(jobcat*salary)))
```

Figure 318. Example: Calculating the minimum salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.mode Function ” on page 250

“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.mode Function

Syntax

```
summary.mode(<algebra>)
```

or

```
summary.mode(<binning function>)
```

or

```
summary.mode(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.mode`.

Description

Calculates the mode, which is the most frequent value. If there is a tie, the smallest value is returned. If the function is evaluating graph algebra, the mode of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.mode(jobcat*educationalLevel)))
```

Figure 319. Example: Calculating the mode of educationalLevel for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246

“summary.min Function ” on page 248
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.percent Function

Description

`summary.percent` is an alias for `summary.percent.sum`. See the topic “summary.percent.sum Function ” on page 260 for more information.

Examples

```
ELEMENT: interval(position(summary.percent(jobcat*salary)))
```

Figure 320. Example: Calculating percentages of a summed variable

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153

“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290

“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.percent.count Function

Syntax

```
summary.percent.count(<algebra>, <base function>)
```

or

```
summary.percent.count(<binning function>, <base function>)
```

or

```
summary.percent.count(<statistic function>, <base function>)
```

<algebra>. Graph algebra, such as x or x*y. In the second case, the percentage is calculated for cases with non-missing y-variable values. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<base function>. A function that specifies the percentage base for `summary.percent.count`. This is optional. The default is `base.all()`.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.percent.count`.

Description

Calculates the percentage of cases within each subgroup compared to the total number of cases.

Examples

```
ELEMENT: interval(position(summary.percent.count(jobcat)))
```

Figure 321. Example: Calculating percentages of counts

```
ELEMENT: interval(position(summary.percent.count(bin.rect(salary))))
```

Figure 322. Example: Graphing a histogram of percentages

Statistic Functions

See “GPL Functions” on page 53 .

Base Functions
“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64
Binning Functions
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
Applies To
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68

“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269

“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.percent.count.cumulative Function

```
summary.percent.count.cumulative(<algebra>, <base function>)
```

or

```
summary.percent.count.cumulative(<binning function>, <base function>)
```

or

```
summary.percent.count.cumulative(<statistic function>, <base function>)
```

<algebra>. Graph algebra, such as x or x*y. In the second case, the percentage is calculated for cases with non-missing y-variable values. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<base function>. A function that specifies the percentage base for `summary.percent.count.cumulative`. This is optional. The default is `base.all()`.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.percent.count.cumulative`.

Description

Calculates the cumulative percentage of cases within each group compared to the total number of cases.

Note: If there are multiple ELEMENT statements, you cannot use cumulative statistics for some graphic elements but not for others. This behavior is prohibited because the results of each statistic function would be blended on the same scale. The units for cumulative statistics do not match the units for non-cumulative statistics, so blending these results is impossible.

Examples

```
ELEMENT: interval(position(summary.percent.count.cumulative(jobcat)))
```

Figure 323. Example: Calculating cumulative percentages of counts

```
ELEMENT: interval(position(summary.percent.count.cumulative(bin.rect(salary))))
```

Figure 324. Example: Calculating a cumulative histogram of percentages

Statistic Functions

See “GPL Functions” on page 53 .

Base Functions

“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240

“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.percent.cumulative Function

Description

`summary.percent.cumulative` is an alias for `summary.percent.sum.cumulative`. See the topic “summary.percent.sum.cumulative Function ” on page 262 for more information.

Note: If there are multiple ELEMENT statements, you cannot use cumulative statistics for some graphic elements but not for others. This behavior is prohibited because the results of each statistic function would be blended on the same scale. The units for cumulative statistics do not match the units for non-cumulative statistics, so blending these results is impossible.

Examples

```
ELEMENT: interval(position(summary.percent.cumulative(jobcat*salary)))
```

Figure 325. Example: Calculating cumulative percentages of a summed variable

Base Functions

“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68

“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.cumulative Function ” on page 258

“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.percent.sum Function

Syntax

```
summary.percent.sum(<algebra>, <base function>)
```

or

```
summary.percent.sum(<binning function>, <base function>)
```

or

```
summary.percent.sum(<statistic function>, <base function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<base function>. A function that specifies the percentage base for `summary.percent.sum`. This is optional. The default is `base.all()`.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.percent.sum`.

Description

Calculates the percentage within each subgroup based on a summed variable compared to the sum across all groups. `summary.percent` is an alias for this function. To obtain percentages of counts, use the `summary.percent.count` function. See the topic “summary.percent.count Function ” on page 254 for more information.

Examples

```
ELEMENT: interval(position(summary.percent.sum(jobcat*salary)))
```

Figure 326. Example: Calculating percentages of a summed variable

Statistic Functions

See “GPL Functions” on page 53 .

Base Functions

“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240

“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.percent.sum.cumulative Function

```
summary.percent.sum.cumulative(<algebra>, <base function>)
```

or

```
summary.percent.sum.cumulative(<binning function>, <base function>)
```

or

```
summary.percent.sum.cumulative(<statistic function>, <base function>)
```

<algebra>. Graph algebra, such as `x*y`. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<base function>. A function that specifies the percentage base for `summary.percent.sum.cumulative`. This is optional. The default is `base.all()`.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.percent.sum.cumulative`.

Description

Calculates the cumulative percentage within each subgroup based on a summed variable compared to the sum across all groups. `summary.percent.cumulative` is an alias for this function. To obtain

cumulative percentages of counts, use the `summary.percent.count.cumulative` function. See the topic “summary.percent.count.cumulative Function ” on page 256 for more information.

Note: If there are multiple ELEMENT statements, you cannot use cumulative statistics for some graphic elements but not for others. This behavior is prohibited because the results of each statistic function would be blended on the same scale. The units for cumulative statistics do not match the units for non-cumulative statistics, so blending these results is impossible.

Examples

```
ELEMENT: interval(position(summary.percent.sum.cumulative(jobcat*salary)))
```

Figure 327. Example: Calculating cumulative percentages of a summed variable

Statistic Functions

See “GPL Functions” on page 53 .

Base Functions

“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190

“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.percentile Function

Syntax

```
summary.percentile(<algebra>, <function>)
```

or

```
summary.percentile(<binning function>, <function>)
```

or

```
summary.percentile(<statistic function>, <function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<function>. One or more valid functions. These are optional. If no `alpha` function is specified, 0.95 is used for the alpha. If the alpha is 0.5, the result is equivalent to `summary.median`.

Description

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.percentile`.

Description

Calculates the percentile value at the specified alpha. If the function is evaluating graph algebra, the percentile value of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.percentile(jobcat*salary, alpha(0.25))))
```

Figure 328. Example: Calculating the 25th percentile of salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Valid Functions

“alpha Function ” on page 62

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164

“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.percentTrue Function

Syntax

```
summary.percentTrue(<algebra>)
```

or

```
summary.percentTrue(<binning function>)
```

or

```
summary.percentTrue(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.percentTrue`.

Description

Calculate the percentage of cases within each subgroup that evaluate to a *true* value compared to the total number of cases. If the function is evaluating graph algebra, the analysis variable is typically the Boolean result of expression evaluated by the `eval` function. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
TRANS: salGreaterThan = eval(salary>50000)
ELEMENT: interval(position(summary.percentTrue(jobcat*salGreaterThan)))
```

Figure 329. Example: Plotting percentage greater than a value

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155

“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292

“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.proportion Function

Description

`summary.proportion` is an alias for `summary.proportion.sum`. See the topic “summary.proportion.sum Function ” on page 275 for more information.

Examples

```
ELEMENT: interval(position(summary.proportion(jobcat*salary)))
```

Figure 330. Example: Calculating proportions of a summed variable

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234

“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.proportion.count Function

Syntax

```
summary.proportion.count(<algebra>, <base function>)
```

or

```
summary.proportion.count(<binning function>, <base function>)
```

or

```
summary.proportion.count(<statistic function>, <base function>)
```

<algebra>. Graph algebra, such as x or x*y. In the second case, the proportion is calculated for cases with non-missing y-variable values. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<base function>. A function that specifies the percentage base for `summary.proportion.count`. This is optional. The default is `base.all()`.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.proportion.count`.

Description

Calculates the proportion of cases within each subgroup compared to the total number of cases. This function is similar to `summary.percent.count` except it reports a value between 0 and 1 instead of 0 and 100.

Examples

```
ELEMENT: interval(position(summary.proportion.count(jobcat)))
```

Figure 331. Example: Calculating proportions of counts

Statistic Functions

See “GPL Functions” on page 53 .

Base Functions

“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190

“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.proportion.count.cumulative Function

```
summary.proportion.count.cumulative(<algebra>, <base function>)
```

or

```
summary.proportion.count.cumulative(<binning function>, <base function>)
```

or

```
summary.proportion.count.cumulative(<statistic function>, <base function>)
```

<algebra>. Graph algebra, such as x or x*y. In the second case, the proportion is calculated for cases with non-missing y-variable values. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<base function>. A function that specifies the percentage base for `summary.proportion.count.cumulative`. This is optional. The default is `base.all()`.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.proportion.count.cumulative`.

Description

Calculates the cumulative proportion of cases within each group compared to the total number of cases. This function is similar to `summary.percent.count.cumulative` except it reports a value between 0 and 1 instead of 0 and 100.

Note: If there are multiple ELEMENT statements, you cannot use cumulative statistics for some graphic elements but not for others. This behavior is prohibited because the results of each statistic function would be blended on the same scale. The units for cumulative statistics do not match the units for non-cumulative statistics, so blending these results is impossible.

Examples

```
ELEMENT: interval(position(summary.proportion.count.cumulative(jobcat)))
```

Figure 332. Example: Calculating cumulative proportions of counts

Statistic Functions

See “GPL Functions” on page 53 .

Base Functions

“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159

“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296

“transparency Function (For GPL Graphic Elements)” on page 300

summary.proportion.cumulative Function

Description

summary.proportion.cumulative is an alias for summary.proportion.sum.cumulative. See the topic “summary.proportion.sum.cumulative Function ” on page 277 for more information.

Note: If there are multiple ELEMENT statements, you cannot use cumulative statistics for some graphic elements but not for others. This behavior is prohibited because the results of each statistic function would be blended on the same scale. The units for cumulative statistics do not match the units for non-cumulative statistics, so blending these results is impossible.

Examples

```
ELEMENT: interval(position(summary.proportion.cumulative(jobcat*salary)))
```

Figure 333. Example: Calculating cumulative proportions of a summed variable

summary.proportion.sum Function

Syntax

```
summary.proportion.sum(<algebra>, <base function>)
```

or

```
summary.proportion.sum(<binning function>, <base function>)
```

or

```
summary.proportion.sum(<statistic function>, <base function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<base function>. A function that specifies the percentage base for summary.proportion.sum. This is optional. The default is base.all().

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate summary.proportion.sum.

Description

Calculates the proportion within each subgroup based on a summed variable compared to the sum across all groups. summary.proportion is an alias for this function. summary.proportion.sum is similar to summary.percent.sum except it reports a value between 0 and 1 instead of 0 and 100.

To obtain proportions of counts, use the summary.proportion.count function. See the topic “summary.proportion.count Function ” on page 270 for more information.

Examples

```
ELEMENT: interval(position(summary.proportion.sum(jobcat*salary)))
```

Figure 334. Example: Calculating proportions of a summed variable

Statistic Functions

See “GPL Functions” on page 53 .

Base Functions

“base.aesthetic Function ” on page 63

“base.all Function ” on page 63
“base.coordinate Function ” on page 64
Binning Functions
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
Applies To
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246

“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.proportion.sum.cumulative Function

```
summary.proportion.sum.cumulative(<algebra>, <base function>)
```

or

```
summary.proportion.sum.cumulative(<binning function>, <base function>)
```

or

```
summary.proportion.sum.cumulative(<statistic function>, <base function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<base function>. A function that specifies the percentage base for `summary.proportion.sum.cumulative`. This is optional. The default is `base.all()`.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.proportion.sum.cumulative`.

Description

Calculates the cumulative proportion within each subgroup based on a summed variable compared to the sum across all groups. `summary.proportion.cumulative` is an alias for this function. `summary.proportion.sum.cumulative` is similar to `summary.percent.sum.cumulative` except it reports a value between 0 and 1 instead of 0 and 100.

To obtain cumulative proportions of counts, use the `summary.proportion.count.cumulative` function. See the topic “summary.proportion.count.cumulative Function ” on page 272 for more information.

Note: If there are multiple ELEMENT statements, you cannot use cumulative statistics for some graphic elements but not for others. This behavior is prohibited because the results of each statistic function would be blended on the same scale. The units for cumulative statistics do not match the units for non-cumulative statistics, so blending these results is impossible.

Examples

```
ELEMENT: interval(position(summary.proportion.sum.cumulative(jobcat*salary)))
```

Figure 335. Example: Calculating cumulative proportions of a summed variable

Statistic Functions

See “GPL Functions” on page 53 .

Base Functions

“base.aesthetic Function ” on page 63
“base.all Function ” on page 63
“base.coordinate Function ” on page 64

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188

“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.proportionTrue Function

Syntax

```
summary.proportionTrue(<algebra>)
```

or

```
summary.proportionTrue(<binning function>)
```

or

```
summary.proportionTrue(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.proportionTrue`.

Description

Calculate the proportion of cases within each subgroup that evaluate to a *true* value compared to the total number of cases. This function is similar to `summary.percentTrue` except it reports a value between 0 and 1 instead of 0 and 100. If `summary.proportionTrue` is evaluating graph algebra, the analysis variable is typically the Boolean result of expression evaluated by the `eval` function. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
TRANS: salGreaterThan = eval(salary>50000)
ELEMENT: interval(position(summary.proportionTrue(jobcat*salGreaterThan)))
```

Figure 336. Example: Plotting proportion greater than a value

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170

“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.range Function

Syntax

```
summary.range(<algebra>)
```

or

```
summary.range(<binning function>)
```

or

```
summary.range(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.range`.

Description

Calculates the range, which is the difference between the minimum and maximum values. If the function is evaluating graph algebra, the range of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

This function returns only one value. If you want to display the interval between the minimum and maximum values, use `region.spread.range`. See the topic “region.spread.range Function ” on page 194 for more information.

Examples

```
ELEMENT: interval(position(summary.range(jobcat*salary)))
```

Figure 337. Example: Calculating the range of salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153

“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290

“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.sd Function

Syntax

```
summary.sd(<algebra>)
```

or

```
summary.sd(<binning function>)
```

or

```
summary.sd(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.sd`.

Description

Calculates the standard deviation, which is the square root of the variance. If the function is evaluating graph algebra, the standard deviation of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.sd(jobcat*salary)))
```

Figure 338. Example: Calculating the standard deviation of salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
Applies To
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149

“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“smooth.spline Function ” on page 222
“smooth.step Function ” on page 224
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279

“summary.range Function ” on page 282
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.se Function

Syntax

```
summary.se(<algebra>)
```

or

```
summary.se(<binning function>)
```

or

```
summary.se(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.se`.

Description

Calculates the standard error of the mean, which is the standard deviation of the sample means. If the function is evaluating graph algebra, the standard error of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.se(jobcat*salary)))
```

Figure 339. Example: Calculating the standard error of salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84

“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275

“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.se.kurtosis Function

Syntax

```
summary.se.kurtosis(<algebra>)
```

or

```
summary.se.kurtosis(<binning function>)
```

or

```
summary.se.kurtosis(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.se.kurtosis`.

Description

Calculates the standard error of the kurtosis, which is the standard deviation of the sample kurtosis values. If the function is evaluating graph algebra, the standard error of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.se.kurtosis(jobcat*salary)))
```

Figure 340. Example: Calculating the standard error of the kurtosis of salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270

“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.se.skewness Function

Syntax

```
summary.se.skewness(<algebra>)
```

or

```
summary.se.skewness(<binning function>)
```

or

```
summary.se.skewness(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.se.skewness`.

Description

Calculates the standard error of the skewness, which is the standard deviation of the sample skewness values. If the function is evaluating graph algebra, the standard error of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.se.skewness(jobcat*salary)))
```

Figure 341. Example: Calculating the standard error of the skewness of salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions
“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
Applies To
“bin.dot Function ” on page 66

“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264

“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.sum Function

Syntax

```
summary.sum(<algebra>)
```

or

```
summary.sum(<binning function>)
```

or

```
summary.sum(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.sum`.

Description

Calculates the sum. If the function is evaluating graph algebra, the sum of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.sum(jobcat*salary)))
```

Figure 342. Example: Calculating the sum of salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260

“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum.cumulative Function ” on page 294
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.sum.cumulative Function

Syntax

```
summary.sum.cumulative(<algebra>)
```

or

```
summary.sum.cumulative(<binning function>)
```

or

```
summary.sum.cumulative(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.sum.cumulative`.

Description

Calculates the cumulative sum. If the function is evaluating graph algebra, the cumulative sum of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Note: If there are multiple ELEMENT statements, you cannot use cumulative statistics for some graphic elements but not for others. This behavior is prohibited because the results of each statistic function would be blended on the same scale. The units for cumulative statistics do not match the units for non-cumulative statistics, so blending these results is impossible.

Examples

```
ELEMENT: interval(position(summary.sum(jobcat*salary)))
```

Figure 343. Example: Calculating the cumulative sum

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240
“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248

“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.variance Function ” on page 296
“transparency Function (For GPL Graphic Elements)” on page 300

summary.variance Function

Syntax

```
summary.variance(<algebra>)
```

or

```
summary.variance(<binning function>)
```

or

```
summary.variance(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to “Brief Overview of GPL Algebra” on page 3 for an introduction to graph algebra.

<binning function>. A binning function.

<statistic function>. Another statistic function. The result of the embedded statistic is used to calculate `summary.variance`.

Description

Calculates the variance, which is the sum of squared deviations from the mean divided by one less than the number of cases. If the function is evaluating graph algebra, the variance of the analysis variable is returned. For more information about analysis variables, see the discussion in “Brief Overview of GPL Algebra” on page 3 .

Examples

```
ELEMENT: interval(position(summary.variance(jobcat*salary)))
```

Figure 344. Example: Calculating the variance of salary for each jobcat category

Statistic Functions

See “GPL Functions” on page 53 .

Binning Functions

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72

Applies To

“bin.dot Function ” on page 66
“bin.hex Function ” on page 68
“bin.quantile.letter Function ” on page 70
“bin.rect Function ” on page 72
“color Function (For GPL Graphic Elements)” on page 81
“color.brightness Function (For GPL Graphic Elements)” on page 82
“color.hue Function (For GPL Graphic Elements)” on page 84
“color.saturation Function (For GPL Graphic Elements)” on page 85
“link.alpha Function ” on page 145
“link.complete Function ” on page 147
“link.delaunay Function ” on page 149
“link.distance Function ” on page 151
“link.gabriel Function ” on page 153
“link.hull Function ” on page 155
“link.influence Function ” on page 157
“link.join Function ” on page 159
“link.mst Function ” on page 162
“link.neighbor Function ” on page 164
“link.relativeNeighborhood Function ” on page 166
“link.sequence Function ” on page 168
“link.tsp Function ” on page 170
“position Function (For GPL Graphic Elements)” on page 180
“region.confi.count Function ” on page 184
“region.confi.mean Function ” on page 186
“region.confi.percent.count Function ” on page 188
“region.confi.proportion.count Function ” on page 190
“region.confi.smooth Function ” on page 192
“region.spread.range Function ” on page 194
“region.spread.sd Function ” on page 196
“region.spread.se Function ” on page 198
“size Function (For GPL Graphic Elements)” on page 207
“split Function ” on page 227
“summary.count Function ” on page 229
“summary.count.cumulative Function ” on page 231
“summary.countTrue Function ” on page 234
“summary.first Function ” on page 236
“summary.kurtosis Function ” on page 238
“summary.last Function ” on page 240

“summary.max Function ” on page 242
“summary.mean Function ” on page 244
“summary.median Function ” on page 246
“summary.min Function ” on page 248
“summary.mode Function ” on page 250
“summary.percent Function ” on page 252
“summary.percent.count Function ” on page 254
“summary.percent.count.cumulative Function ” on page 256
“summary.percent.cumulative Function ” on page 258
“summary.percent.sum Function ” on page 260
“summary.percent.sum.cumulative Function ” on page 262
“summary.percentile Function ” on page 264
“summary.percentTrue Function ” on page 267
“summary.proportion Function ” on page 269
“summary.proportion.count Function ” on page 270
“summary.proportion.count.cumulative Function ” on page 272
“summary.proportion.cumulative Function ” on page 275
“summary.proportion.sum Function ” on page 275
“summary.proportion.sum.cumulative Function ” on page 277
“summary.proportionTrue Function ” on page 279
“summary.range Function ” on page 282
“summary.sd Function ” on page 284
“summary.se Function ” on page 286
“summary.se.kurtosis Function ” on page 288
“summary.se.skewness Function ” on page 290
“summary.sum Function ” on page 292
“summary.sum.cumulative Function ” on page 294
“transparency Function (For GPL Graphic Elements)” on page 300

t Function

Syntax

```
t(<degree of freedom>)
```

<degrees of freedom>. Numeric value indicating the degrees of freedom. This values must be greater than 0.

Description

Specifies a Student's *t* distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), t(5))
```

Figure 345. Example: Specifying a Student's t distribution for the probability scale

Applies To
“prob Scale ” on page 34

texture.pattern Function

Syntax

```
texture.pattern(<algebra>)
```

or

```
texture.pattern(texture.pattern.<pattern constant>)
```

or

```
texture.pattern(<statistic function>)
```

<algebra>. Graph algebra using one categorical variable or a blend of categorical variables. Each unique variable value results in a different pattern. For example, if you were creating a stacked bar chart, the argument of the `texture.pattern` function would be the variable that controls the stacking. Each stack segment would have a different pattern.

<pattern constant>. A constant indicating a specific pattern, such as `stripes`. See the topic “Pattern Constants ” on page 385 for more information.

<statistic function>. A statistic function.

Description

Controls the fill pattern of the associated graphic element. The color of the lines in the pattern is specified by `color.exterior`. The color of the pattern's background is specified by `color.interior`. `texture.pattern.solid` contains no lines or foreground. Therefore, using `texture.pattern.solid` results in a solid element whose color is specified by `color.interior`.

Examples

```
ELEMENT: line(position(x*y), texture.pattern(texture.pattern.checkered))
```

Figure 346. Example: Specifying a pattern

```
ELEMENT: point(position(x*y), texture.pattern(z))
```

Figure 347. Example: Using the values of a variable to control pattern

Statistic Functions

See “GPL Functions” on page 53 .

Applies To

“area Element ” on page 44
“interval Element ” on page 46
“point Element ” on page 48
“schema Element ” on page 50

ticks Function

Syntax

```
ticks()
```

or

```
ticks(null())
```

Description

Specifies that major ticks should be drawn for the axis. Ticks are drawn by default, so this function is typically used only with `null()` to hide the tick marks.

Examples

```
GUIDE: axis(dim(2), ticks(null()))
```

Figure 348. Example: Hiding tick marks

Applies To
“axis Guide Type ” on page 39

to Function

Syntax

```
to(<variable name>)
```

<variable name>. The name of a variable previously defined in the GPL by a DATA statement.

Description

Specifies one of the pair of nodes that defines an edge relation. The is the node that defines the end point for the edge.

Examples

```
ELEMENT: edge(position(layout.dag(node(id), from(fromVar), to(toVar))))
```

Figure 349. Example: Creating a directed acyclic graph

Applies To
“layout.circle Function ” on page 131
“layout.dag Function ” on page 133
“layout.data Function ” on page 135
“layout.grid Function ” on page 137
“layout.network Function ” on page 139
“layout.random Function ” on page 141
“layout.tree Function ” on page 143

transparency Function (For GPL Graphic Elements)

Note: If you are modifying the transparency for a guide, refer to “transparency Function (For GPL Guides)” on page 301 .

Syntax

```
transparency(<algebra>)
```

or

```
transparency(transparency."transparency value")
```

or

```
transparency(<statistic function>)
```

<algebra>. Graph algebra using one variable or a blend of variables. The variable value results in a different transparency value. For example, if you were creating a stacked bar chart, the argument of the `transparency` function would be the variable that controls the stacking. Each stack segment would have a different degree of transparency.

"transparency value". A value between 0 and 1 that indicates the level of transparency. A value of 1 indicates full transparency, while a value of 0 indicates no transparency (completely opaque).

<statistic function>. A statistic function.

Description

Specifies the transparency of the associated graphic element. You can use another variable or variables to control the transparency or set a fixed value. To specify the transparency explicitly for the fill or border of

the graphic element, you can append `.interior` or `.exterior` to the function. Using `transparency` without a qualifier implies `transparency.interior`.

Examples

```
ELEMENT: point(position(x*y), transparency(z))
```

Figure 350. Example: Using a variable to control transparency

```
ELEMENT: interval(position(x*y), transparency(transparency."0.6"))
```

Figure 351. Example: Specifying a value for transparency

```
ELEMENT: interval(position(x*y),
        transparency.interior(transparency."0.8"))
```

Figure 352. Example: Specifying a transparency for the fill

Statistic Functions

See “GPL Functions” on page 53 .

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

transparency Function (For GPL Guides)

Note: If you are modifying the transparency for a graphic element (like a bar or point), refer to “transparency Function (For GPL Graphic Elements)” on page 300 .

Syntax

```
transparency(transparency."transparency value")
```

"transparency value". A value between 0 and 1 that indicates the level of transparency. A value of 1 indicates full transparency, while a value of 0 indicates no transparency (completely opaque).

Description

Controls the transparency of reference lines.

Examples

```
GUIDE: form.line(position(*,2000), transparency(transparency."0.5"))
```

Figure 353. Example: Specifying a transparency for a reference line

Applies To

“form.line Guide Type ” on page 40

transpose Function

Syntax

```
transpose(<coord>)
```

<coord>. A valid coordinate type or transformation function. This is optional.

Description

Transposes the coordinate system.

Examples

```
COORD: transpose()
```

Figure 354. Example: Transposing a 2-D rectangular coordinate system

```
COORD: transpose(rect(dim(1,2), cluster(3)))
```

Figure 355. Example: Transposing a clustered coordinate system

Coordinate Types and Transformations
“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“mirror Function ” on page 174
“project Function ” on page 182
“reflect Function ” on page 183
“wrap Function ” on page 305
Applies To
“COORD Statement ” on page 22
“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“project Function ” on page 182

uniform Function

Syntax

```
uniform(<minimum>, <maximum>)
```

<minimum>. Numeric value indicating the minimum.

<maximum>. Numeric value indicating the maximum.

Description

Specifies a uniform distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), uniform(5000, 20000))
```

Figure 356. Example: Specifying a uniform distribution for the probability scale

Applies To
“prob Scale ” on page 34

unit.percent Function

Syntax

```
unit.percent()
```

Description

Transforms the values on an axis into percents. The percent value is in relation to the largest value of the variable displayed on the axis. This transformation makes most sense when a cumulative value is displayed on the main axis.

Examples

```
GUIDE: axis(dim(2), label("Percent"), unit.percent())
```

Figure 357. Example: Adding a percent axis

Applies To

“axis Guide Type ” on page 39

userSource Function

Syntax

```
userSource(id("source name"), <function>)
```

"source name". The name of the data source as defined by the application that is calling GPL. For example, if you were using GPL with IBM SPSS Statistics GGRAPH syntax, the source name is the name as defined in the DATASET subcommand.

<function>. One or more valid functions. These are optional.

Description

Reads the contents of a data source that an IBM Corp. application passes to GPL.

Examples

```
SOURCE: mydata = userSource(id("graphdataset"))
```

Figure 358. Example: Reading a userSource

Valid Functions

“missing.listwise Function ” on page 175
“missing.pairwise Function ” on page 176
“weight Function ” on page 305

Applies To

“SOURCE Statement ” on page 21
“csvSource Function ” on page 86
“savSource Function ” on page 202
“sqlSource Function ” on page 227

values Function

Syntax

```
values("category name", "category name" ...)
```

"category name". The string representing the category on the axis.

Description

Specifies the categorical values on an axis. Only these specified values are included on the axis, even if these values do not occur in the data or other values do occur in the data.

Examples

```
SCALE: cat(dim(1), values("Male", "Female"))
```

Figure 359. Example: Specifying the categories

Applies To

“cat Scale ” on page 29

visible Function

Syntax

```
visible(<algebra>)
```

<algebra>. The name of a categorical variable.

Description

Controls the visibility of the graphic element, based on categories in a categorical variable. You can use this function in conjunction with the `map` function to hide specific categories of data. The specific constants for the visible aesthetic are `visible.true` and `visible.false`.

Examples

```
SCALE: cat(aesthetic(aesthetic.visible), map(("m", visible.false)))
ELEMENT: line(position(salbegin*salary), visible(gender))
```

Figure 360. Example: Hiding categories

Applies To

“area Element ” on page 44
“edge Element ” on page 45
“interval Element ” on page 46
“line Element ” on page 47
“path Element ” on page 48
“point Element ” on page 48
“polygon Element ” on page 49
“schema Element ” on page 50

weibull Function

Syntax

```
weibull(<rate>, <scale>)
```

<rate>. Numeric value specifying the rate parameter for the distribution.

<scale>. Numeric value specifying the scale parameter for the distribution. This value must be greater than 0.

Description

Specifies a Weibull distribution for the probability scale.

Examples

```
SCALE: prob(dim(2), weibull(5, 2))
```

Figure 361. Example: Specifying a Weibull distribution for the probability scale

Applies To

“prob Scale ” on page 34

weight Function

Syntax

```
weight(<variable name>)
```

<variable name>. The name of a variable defined in the GPL by a DATA statement.

Description

Specifies that a variable in the dataset contains weights. The weights affect the statistic functions that GPL calculates. Weights can be also used with network graphs to affect the distance between nodes.

In general, the weights act as frequency weights (that is, as if there were multiple occurrences of the records). This function is not suitable for sample weights (in which one case represents many).

Examples

```
SOURCE: mydata = csvSource(file("/Data/Edge data.csv"), weight(weightedVar))
DATA: weightedVar = col(source(mydata), name("weights"))
```

Figure 362. Example: Specifying a weighted variable

Applies To

“csvSource Function ” on page 86
“sqlSource Function ” on page 227
“userSource Function ” on page 303

wrap Function

Syntax

```
wrap(<coord>)
```

<coord>. A valid coordinate type or transformation function. This is optional.

Description

Combines faceted dimensions and wraps the facets depending on the available space for the graph. This function is useful when there are many facets because it forces the graph to utilize the available space. Without this function, faceted graphs can only shrink or grow to fit the space.

Examples

```
COORD: rect(dim(1,2), wrap())
```

Figure 363. Example: Wrapping facets

Coordinate Types and Transformations

“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24
“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“mirror Function ” on page 174
“project Function ” on page 182
“reflect Function ” on page 183
“transpose Function ” on page 301

Applies To

“COORD Statement ” on page 22
“parallel Coordinate Type ” on page 23
“polar Coordinate Type ” on page 24

“polar.theta Coordinate Type ” on page 25
“rect Coordinate Type ” on page 26
“project Function ” on page 182

Chapter 3. GPL Examples

This section provides examples organized by broad categories of graph types. You can run the examples by incorporating them into the syntax specific to your application. See the topic “Using the Examples in Your Application ” on page 307 for more information.

Using the Examples in Your Application

If you want to run the examples in your application, you need to incorporate them into the syntax specific to your application.

Using the Examples in IBM SPSS Statistics

The sample files installed with the product can be found in the *Samples* subdirectory of the installation directory. There is a separate folder within the Samples subdirectory for each of the following languages: English, French, German, Italian, Japanese, Korean, Polish, Russian, Simplified Chinese, Spanish, and Traditional Chinese.

Not all sample files are available in all languages. If a sample file is not available in a language, that language folder contains an English version of the sample file.

1. First, you need the right data source. The examples use three different userSources (*Employeedata*, *stocks*, and *customer_subset*), which correspond to IBM SPSS Statistics SAV files located in the directory identified above.
2. With the data source open, create a GGRAPH syntax command.
   - Modify the GRAPHDATASET subcommand by setting the NAME keyword to the `id` of the `userSource` in the GPL example. The VARIABLES keyword also needs to include all the variables identified in the GPL DATA statements.
   - Modify the GRAPHSPEC subcommand so that the SOURCE keyword equals `INLINE`.
3. Follow the GGRAPH command with `BEGIN GPL`, the GPL shown in the example, `END GPL`, and a period.

So if you want to run the simple bar chart example, your syntax would look like the following:

```
GGRAPH
 /GRAPHDATASET NAME="Employeedata" VARIABLES=jobcat salary
 /GRAPHSPEC SOURCE=INLINE.
BEGIN GPL
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary)))
END GPL.
```

Using the Examples in IBM SPSS Visualization Designer

1. Create or open a visualization in IBM SPSS Visualization Designer.
2. If the ViZml/GPL palette is not displayed, from the menus choose:

   View > **Palettes** > **ViZml/GPL**
3. Click the GPL tab.
4. Enter the GPL into the palette. You can also copy and paste examples from the online help.
5. Modify the `file` function for `csvSource` to reference the full path to the CSV file. All of the sample data files are located in the *data* subfolder of the product installation folder. For example:

```
SOURCE: s = csvSource(file("C:/Program Files/IBM/SPSS/Visualization Designer/data/Employee data.csv"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
```

```
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary)))
```

6. After entering and modifying the GPL, click the execute button.

Figure 364. Execute button

Summary Bar Chart Examples

This section provides examples of different types of summary bar charts.

Simple Bar Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary)))
```

Figure 365. GPL for simple bar chart

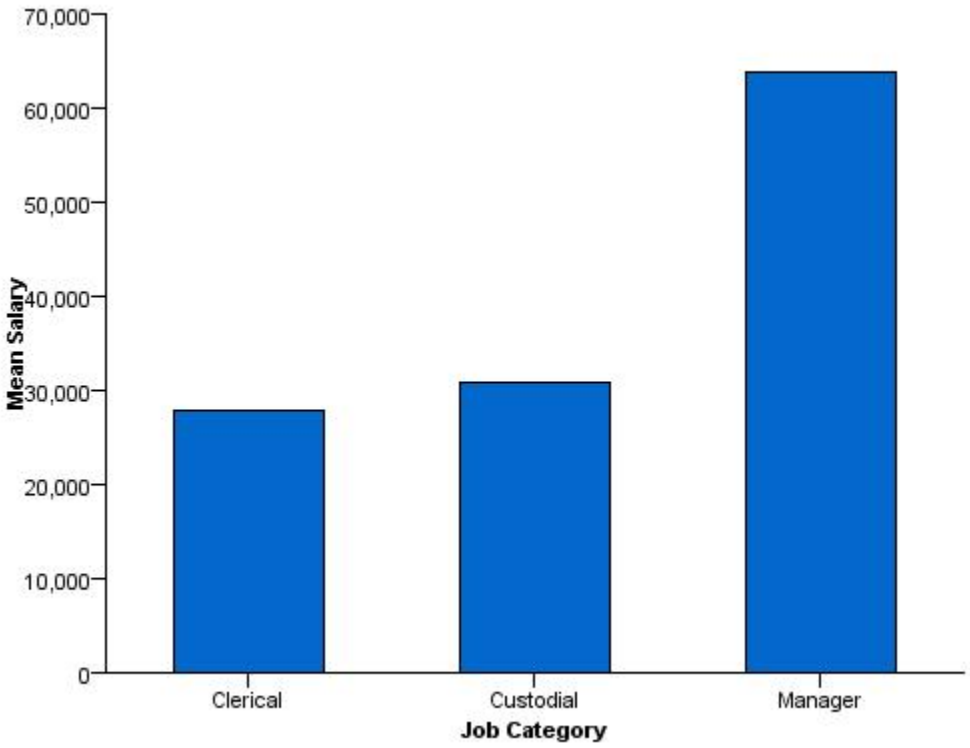


Figure 366. Simple bar chart

Simple Bar Chart of Counts

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Count"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.count(jobcat)))
```

Figure 367. GPL for simple bar chart of counts

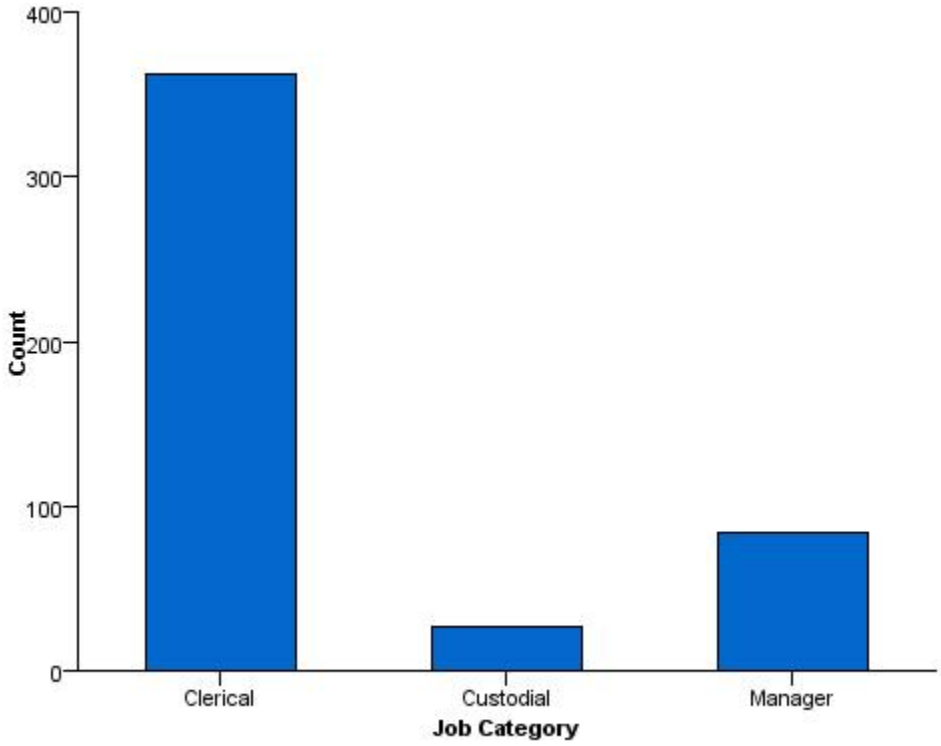


Figure 368. Simple bar chart of counts

Simple Horizontal Bar Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), min(0.0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
COORD: transpose()
ELEMENT: interval(position(summary.mean(jobcat*salary)))
```

Figure 369. GPL for simple horizontal bar chart

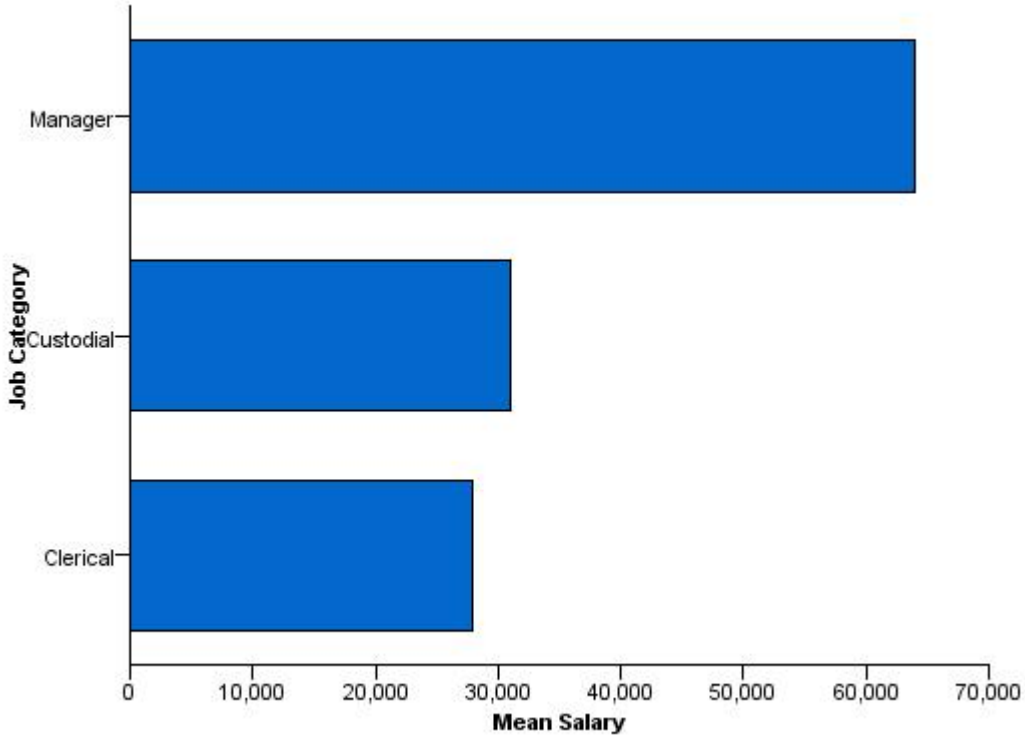


Figure 370. Simple horizontal bar chart

Simple Bar Chart With Error Bars

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary)))
ELEMENT: interval(position(region.confi.mean(jobcat*salary)),
                  shape(shape.ibeam), size(size.".4in"), color(color.black))
```

Figure 371. GPL for simple bar chart with error bars

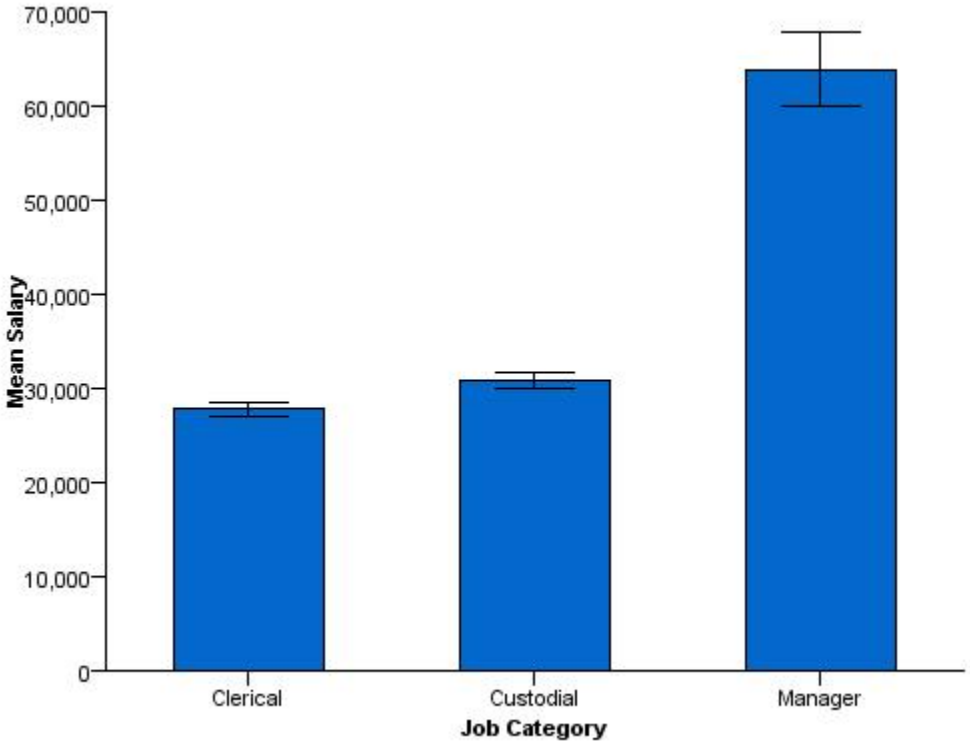


Figure 372. Simple bar chart with error bars

Simple Bar Chart with Bar for All Categories

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean((jobcat+"All")*salary)))
```

Figure 373. GPL for simple bar chart with bar for all categories

Note: Using "All" as the string is arbitrary. Any string would work (e.g., "Total" or "All Categories"). Because it is blended with the *jobcat* categorical variable, the string acts like a new categorical value. This value is the same for all cases in the dataset. Therefore, the bar associated with that string shows the result for all cases in the dataset.

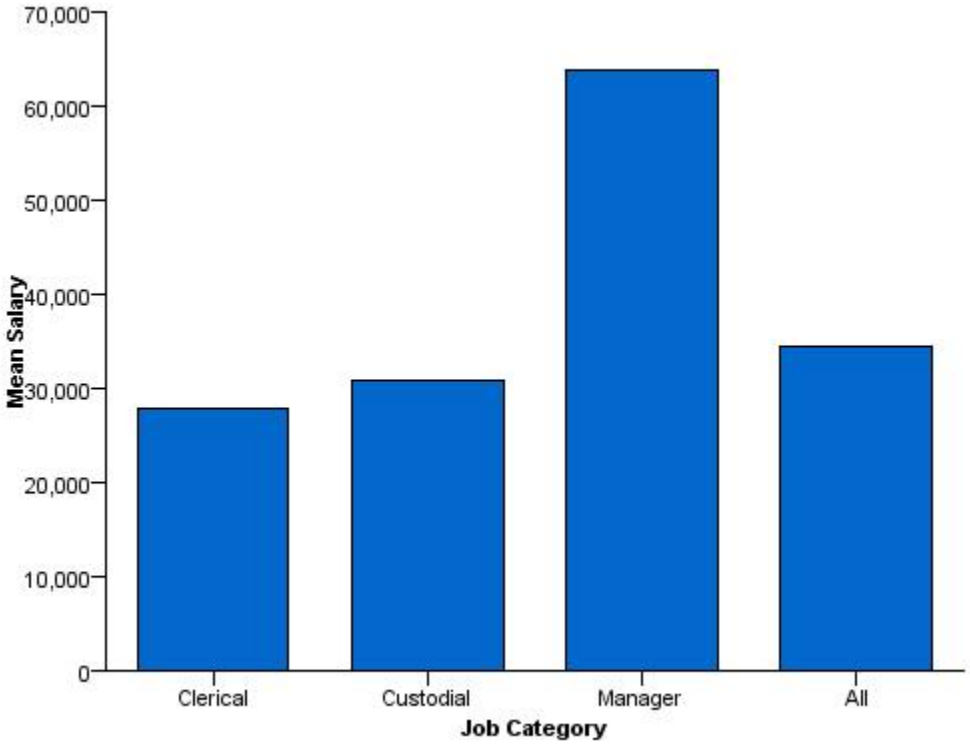


Figure 374. Simple bar chart with a bar for all categories

Stacked Bar Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Count"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval.stack(position(summary.count(jobcat)),
         color(gender))
```

Figure 375. GPL for stacked bar chart

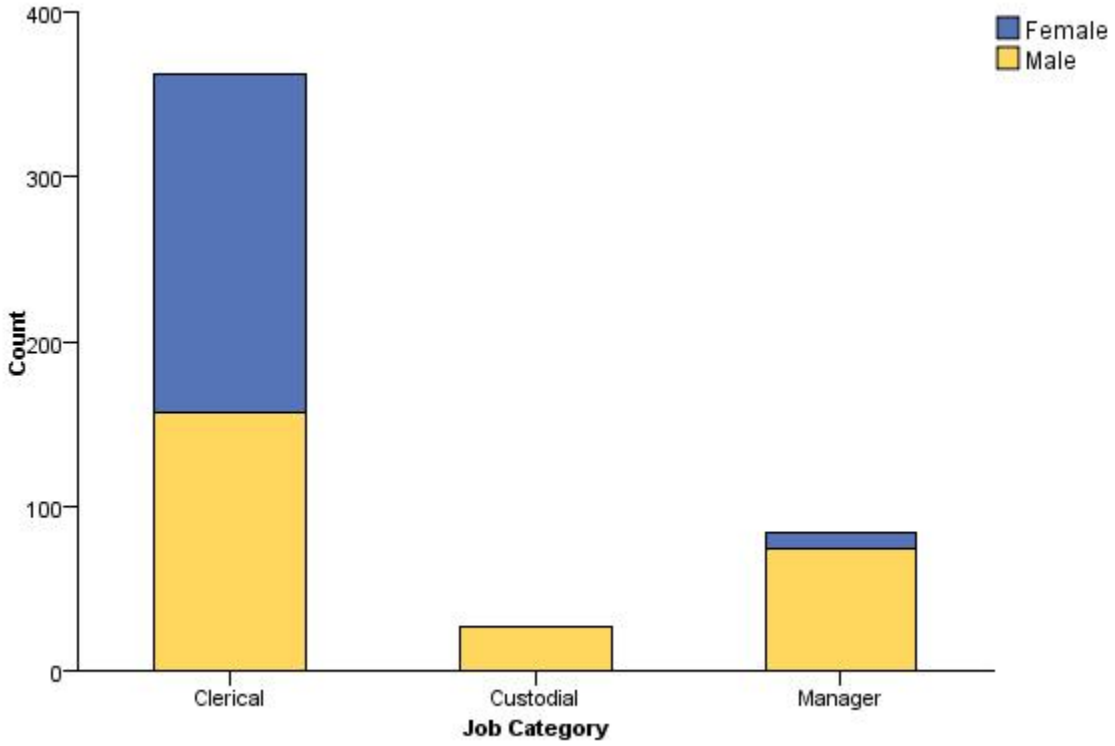


Figure 376. Stacked bar chart

Clustered Bar Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
DATA: salary=col(source(s), name("salary"))
COORD: rect(dim(1,2), cluster(3))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(3), label("Gender"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)), color(jobcat))
```

Figure 377. GPL for clustered bar chart

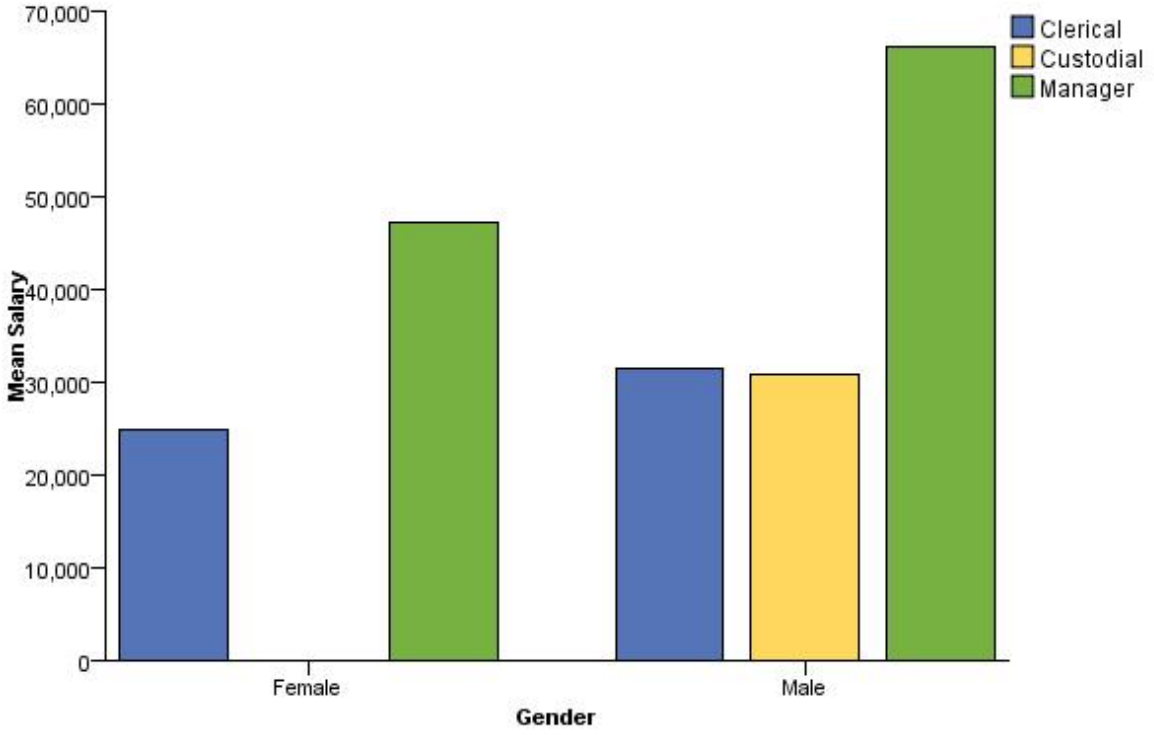


Figure 378. Clustered bar chart

Following is another option for creating a graph that appears clustered. It uses the dodge collision modifier. (See “dodge Collision Modifier ” on page 51 .) Note that the difference between this and the previous example is that empty space is not allocated for missing categories (in this case, the combination of "Female" and "Custodial").

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Gender"))
ELEMENT: interval.dodge(position(summary.mean(gender*salary)),
                        size(size."25%"), color(jobcat))
```

Figure 379. GPL for dodged bar chart

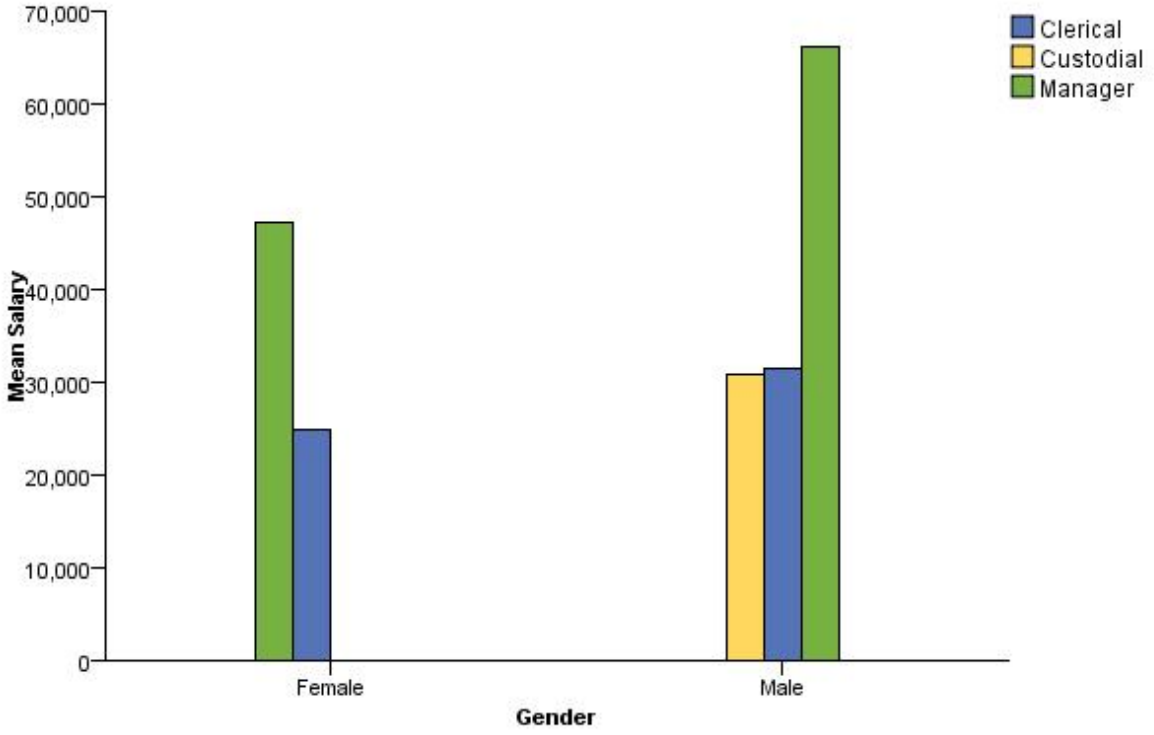


Figure 380. Dodged bar chart

Clustered and Stacked Bar Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: minority = col(source(s), name("minority"), unit.category())
DATA: salary = col(source(s), name("salary"))
COORD: rect(dim(1, 2), cluster(3, 0))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Sum Salary"))
GUIDE: axis(dim(3), label("Job Category"))
GUIDE: legend(aesthetic(aesthetic.color.interior), label("Gender"))
GUIDE: legend(aesthetic(aesthetic.texture.pattern.interior),
           label("Minority Classification"))
ELEMENT: interval.stack(position(summary.sum(gender*salary*jobcat)),
                        color.exterior(color.black),
                        color.interior(gender), texture.pattern.interior(minority))
```

Figure 381. GPL for clustered and stacked bar chart

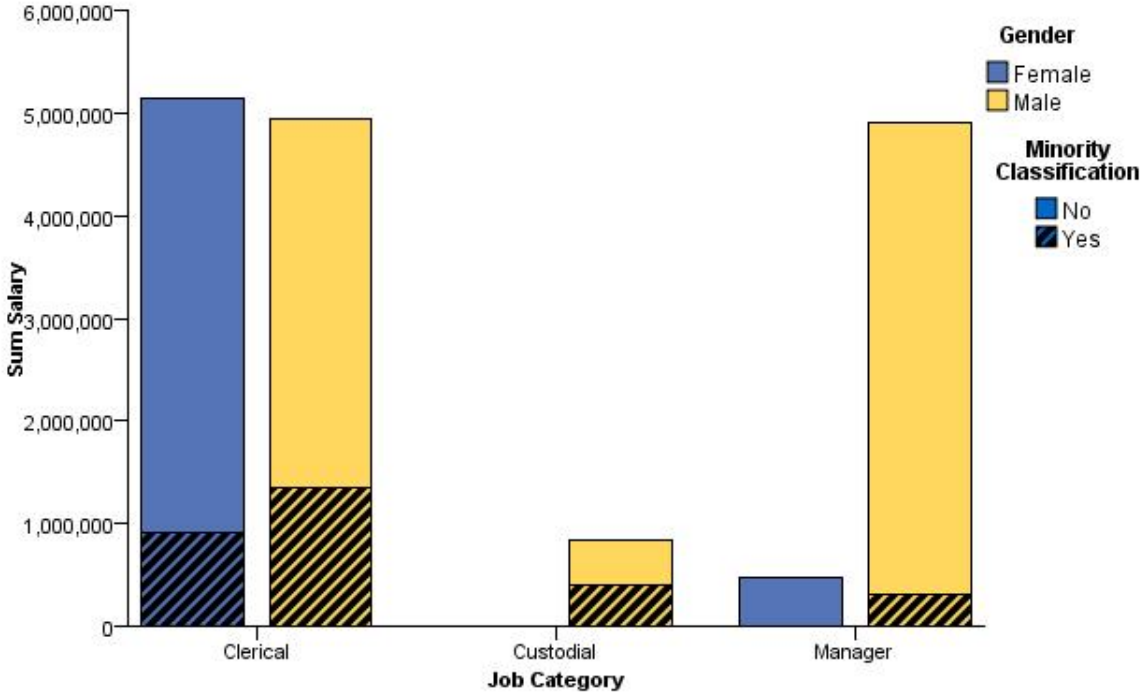


Figure 382. Clustered and stacked bar chart

Bar Chart Using an Evaluation Function

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin = col(source(s), name("salbegin"))
DATA: salary = col(source(s), name("salary"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
TRANS: saldiff = eval(((salary-salbegin)/salary)*100)
COORD: rect(dim(1, 2), cluster(3))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(2), label("Mean Percentage Salary Increase"))
GUIDE: axis(dim(3), label("Employment Category"))
GUIDE: legend(aesthetic(aesthetic.color.interior), label("Gender"))
ELEMENT: interval(position(summary.mean(gender*saldiff*jobcat)),
                  color.interior(gender))
```

Figure 383. GPL using an evaluation function to calculate mean percentage increase

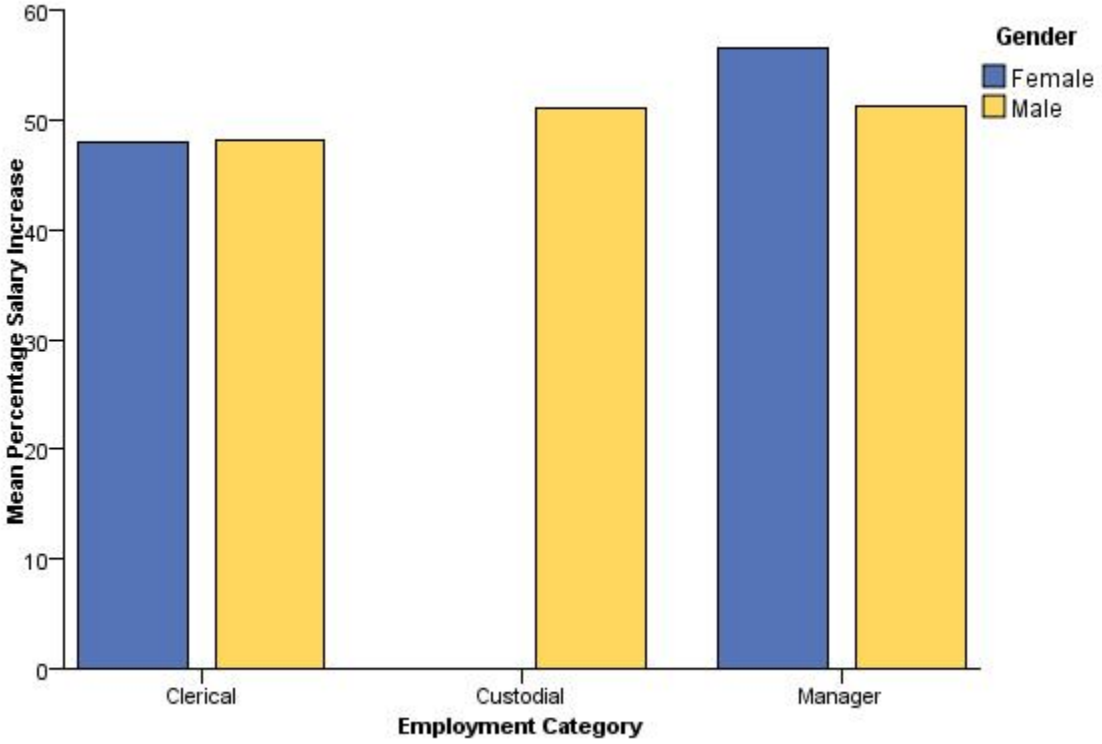


Figure 384. GPL using an evaluation function to calculate mean percentage increase

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
TRANS: greaterThan = eval(salary < 40000)
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(1), label("Employment Category"))
GUIDE: axis(dim(2), label("% < 40000 Salary"))
ELEMENT: interval(position(summary.percentTrue(jobcat*greaterThan)))
```

Figure 385. GPL using an evaluation function to calculate percent less than a value

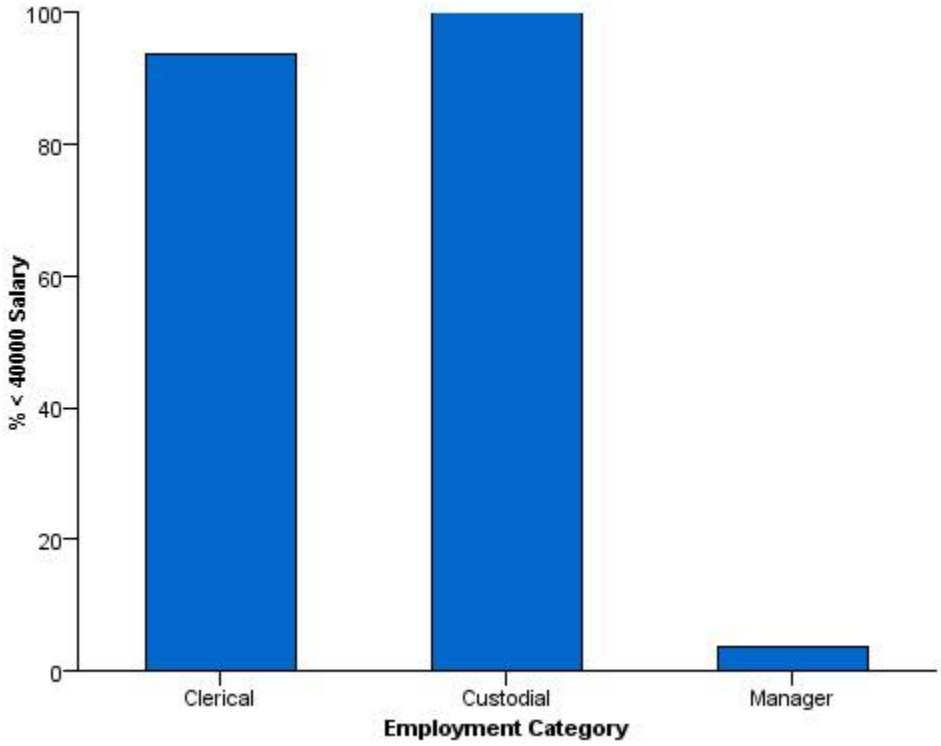


Figure 386. GPL using an evaluation function to calculate percent less than a value

Bar Chart with Mapped Aesthetics

This example demonstrates how you can map a specific categorical value in the graph to a specific aesthetic value. In this case, "Female" bars are colored green in the resulting graph.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
COORD: rect(dim(1, 2), cluster(3))
SCALE: cat(aesthetic(aesthetic.color), map(("f", color.green)))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(3), label("Job Category"))
ELEMENT: interval(position(summary.mean(gender*salary*jobcat)),
                  color(gender))
```

Figure 387. GPL for bar chart with mapped aesthetics

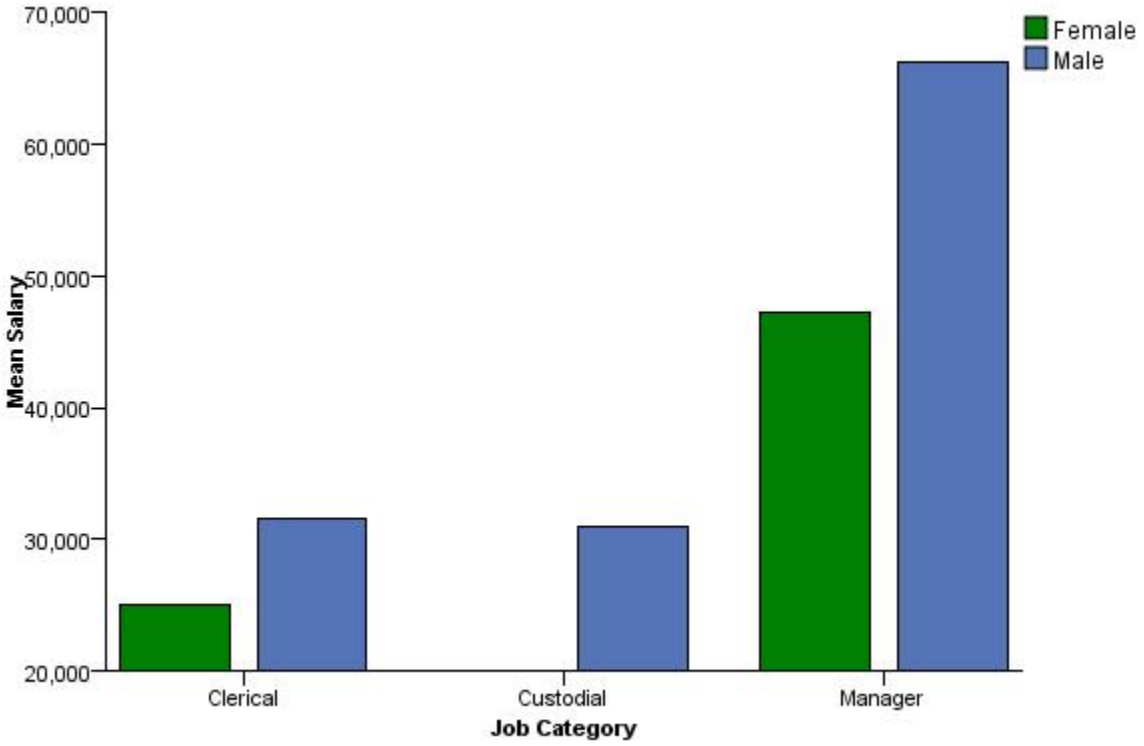


Figure 388. Bar chart with mapped aesthetics

Faceted (Paneled) Bar Chart

Although the following examples create bar charts, faceting is common to all graphs.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: salary = col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(3), label("Gender"))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)))
```

Figure 389. GPL for faceted bar chart

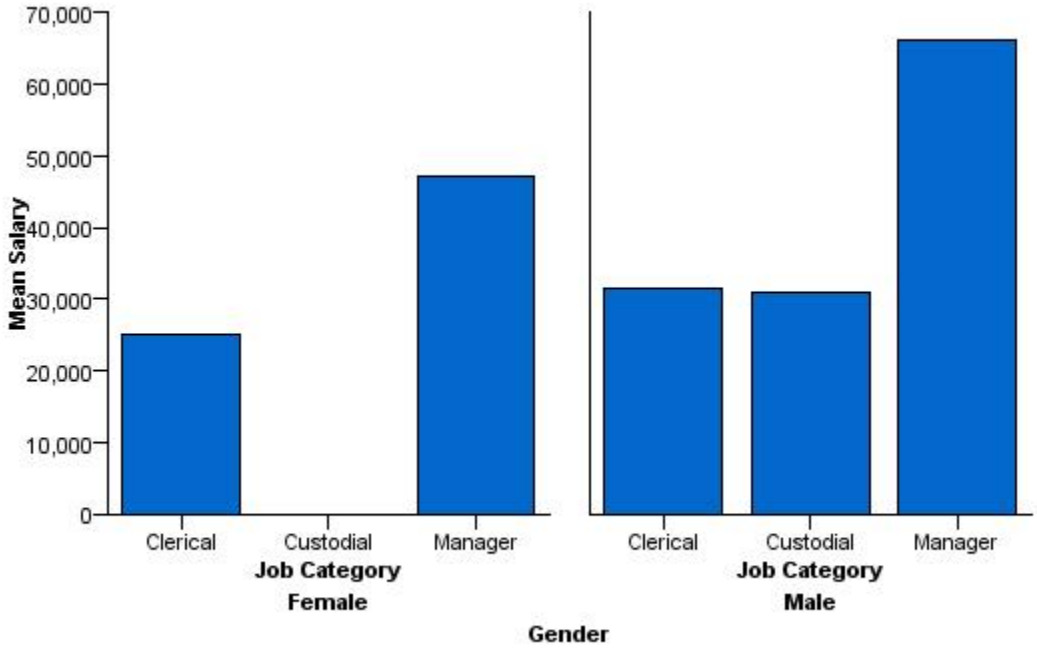


Figure 390. Faceted bar chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: salary = col(source(s), name("salary"))
SCALE: linear(dim(2), include(0.0))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1.1), label("Job Category"))
GUIDE: axis(dim(1), label("Gender"))
ELEMENT: interval(position(summary.mean(jobcat/gender*salary)))
```

Figure 391. GPL for faceted bar chart with nested categories

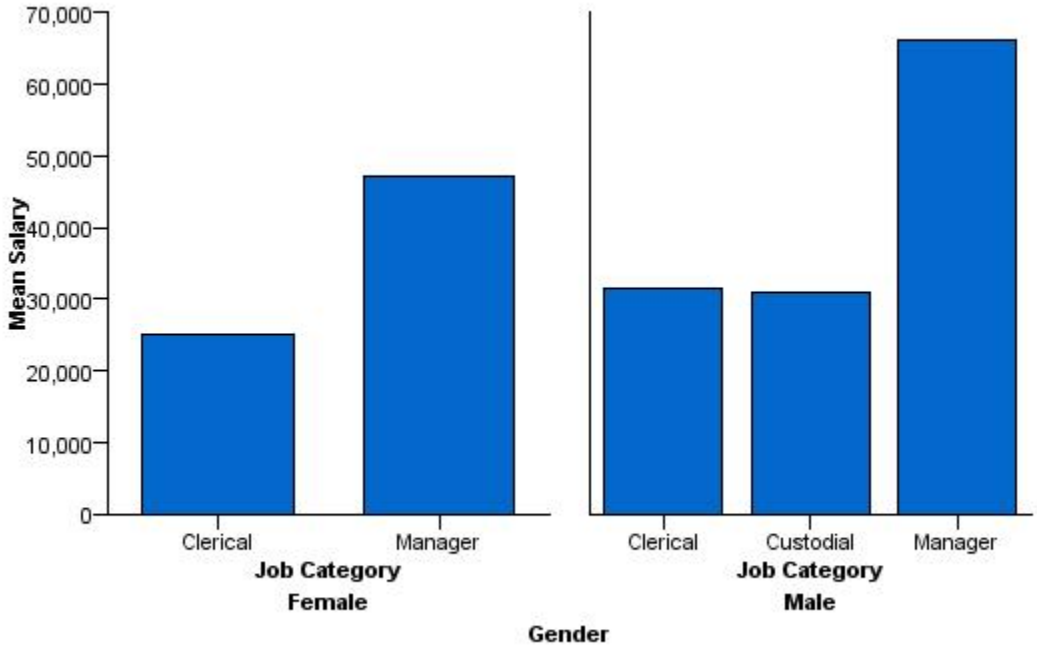


Figure 392. Faceted bar chart with nested categories

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
DATA: minority=col(source(s), name("minority"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), include(0))
GUIDE: axis(dim(4), label("Minority"))
GUIDE: axis(dim(3), label("Gender"))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender*minority)))
```

Figure 393. GPL for multi-faceted bar chart

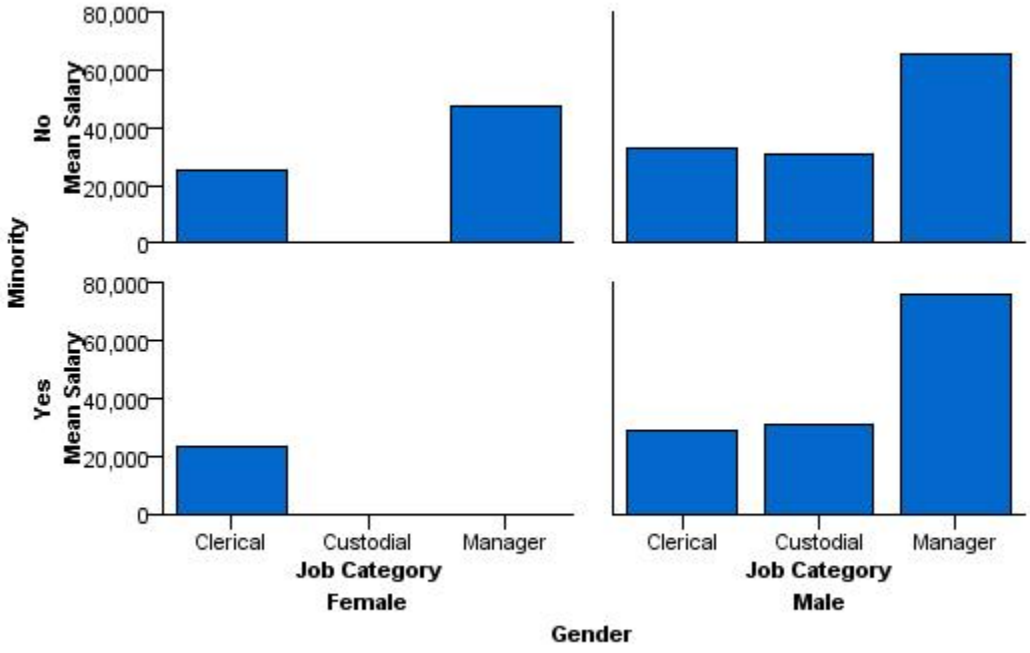


Figure 394. Multi-faceted bar chart

3-D Bar Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
DATA: salary=col(source(s), name("salary"))
COORD: rect(dim(1,2,3))
SCALE: linear(dim(3), include(0))
GUIDE: axis(dim(3), label("Mean Salary"))
GUIDE: axis(dim(2), label("Gender"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(summary.mean(jobcat*gender*salary)))
```

Figure 395. GPL for 3-D bar chart

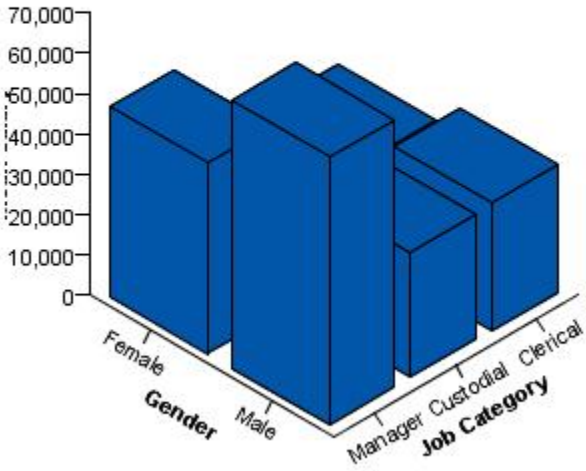


Figure 396. 3-D bar chart

Error Bar Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
GUIDE: axis(dim(2), label("Mean +- 1 SD Current Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(region.confi.mean(jobcat*salary)),
                  shape(shape.ibeam), size(size.".4in"))
ELEMENT: point(position(summary.mean(jobcat*salary)),
               shape(shape.circle), color(color.red), size(size."6px"))
```

Figure 397. GPL for error bar chart

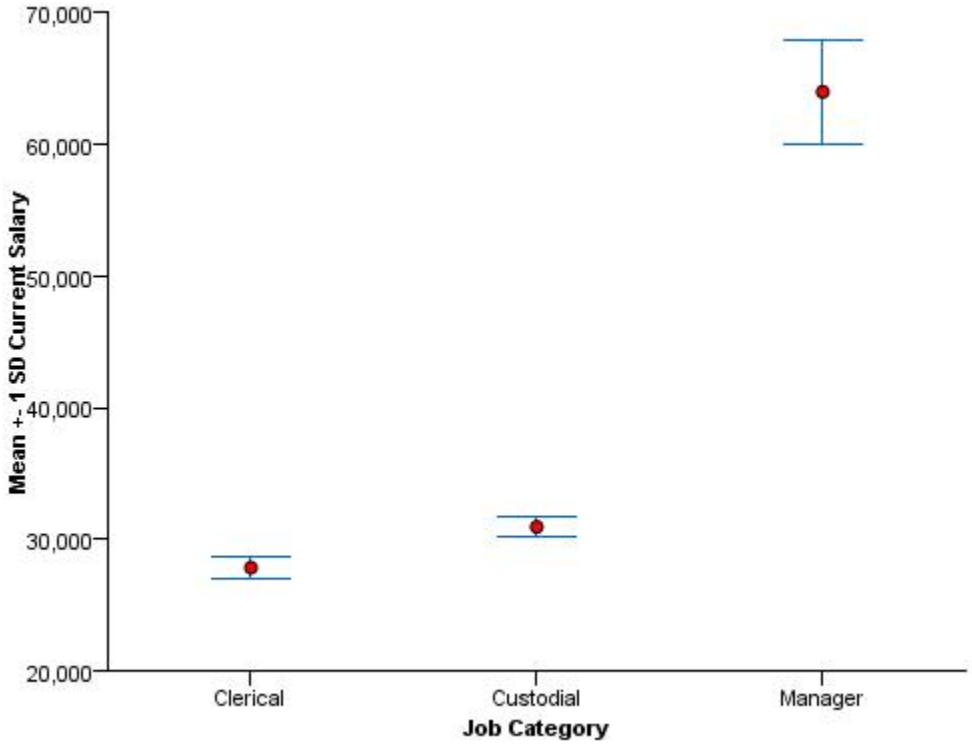


Figure 398. Error bar chart

Histogram Examples

This section provides examples of different types of histograms.

Histogram

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Count"))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: interval(position(summary.count(bin.rect(salary))))
```

Figure 399. GPL for histogram

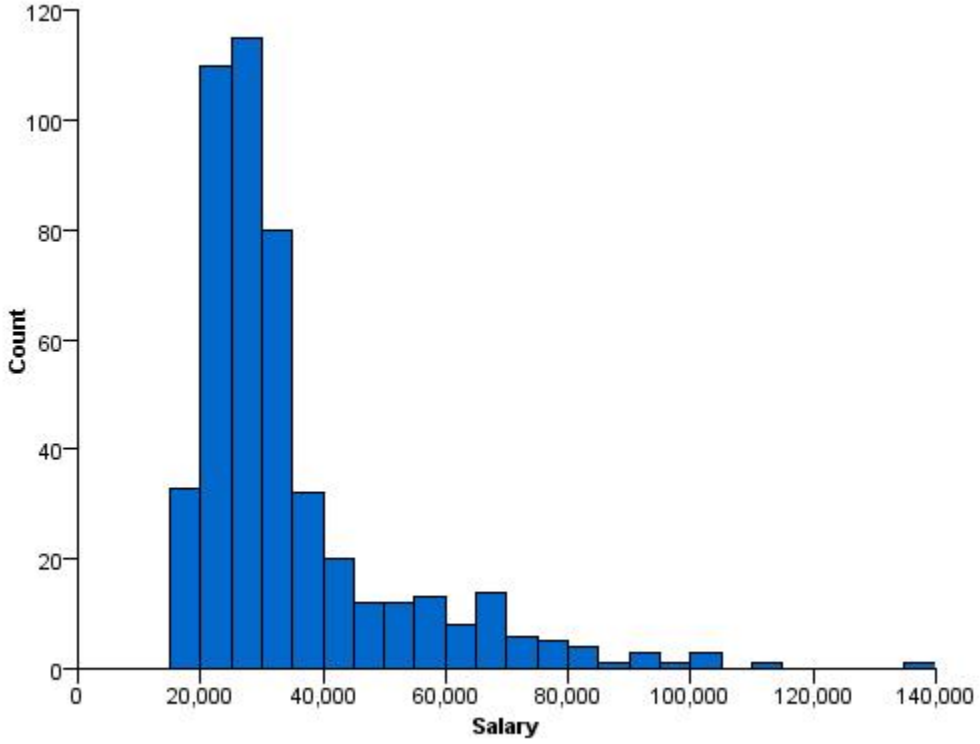


Figure 400. Histogram

Histogram with Distribution Curve

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Count"))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: interval(position(summary.count(bin.rect(salary))))
ELEMENT: line(position(density.normal(salary)))
```

Figure 401. GPL for histogram with normal curve

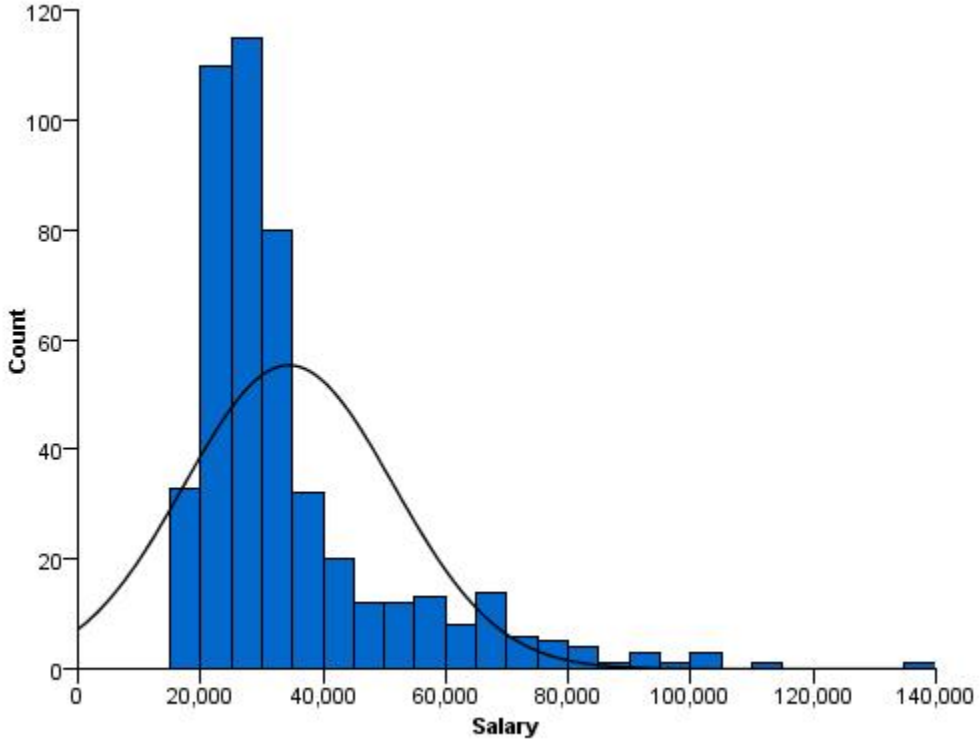


Figure 402. Histogram with normal curve

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Count"))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: interval(position(summary.count(bin.rect(salary))))
ELEMENT: line(position(density.kernel.epanechnikov(salary)))
```

Figure 403. GPL for histogram with kernel density curve

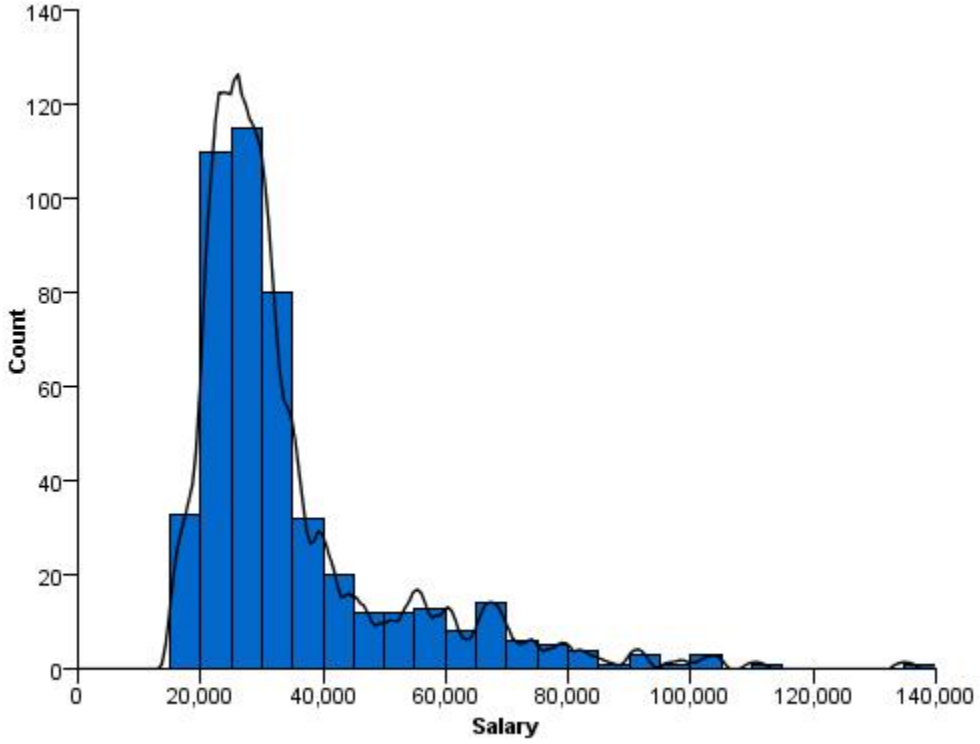


Figure 404. Histogram with kernel density curve

Percentage Histogram

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Percent"))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: interval(position(summary.percent.count(bin.rect(salary))))
```

Figure 405. GPL for percentage histogram

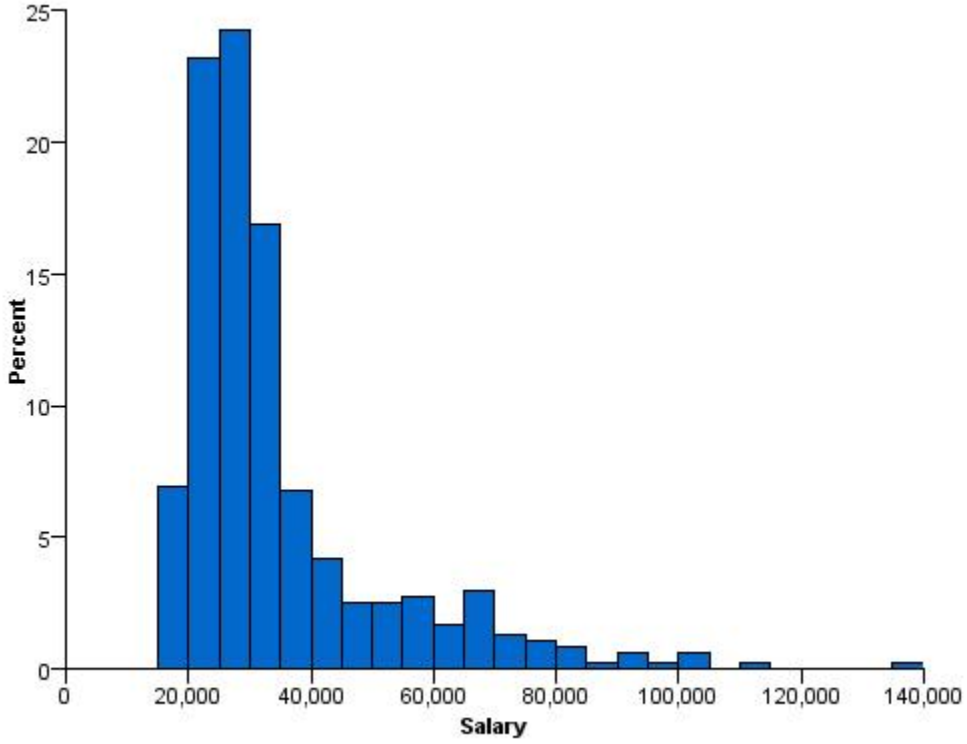


Figure 406. Percentage histogram

Frequency Polygon

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Count"))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: area(position(summary.count(bin.rect(salary))))
```

Figure 407. GPL for frequency polygon

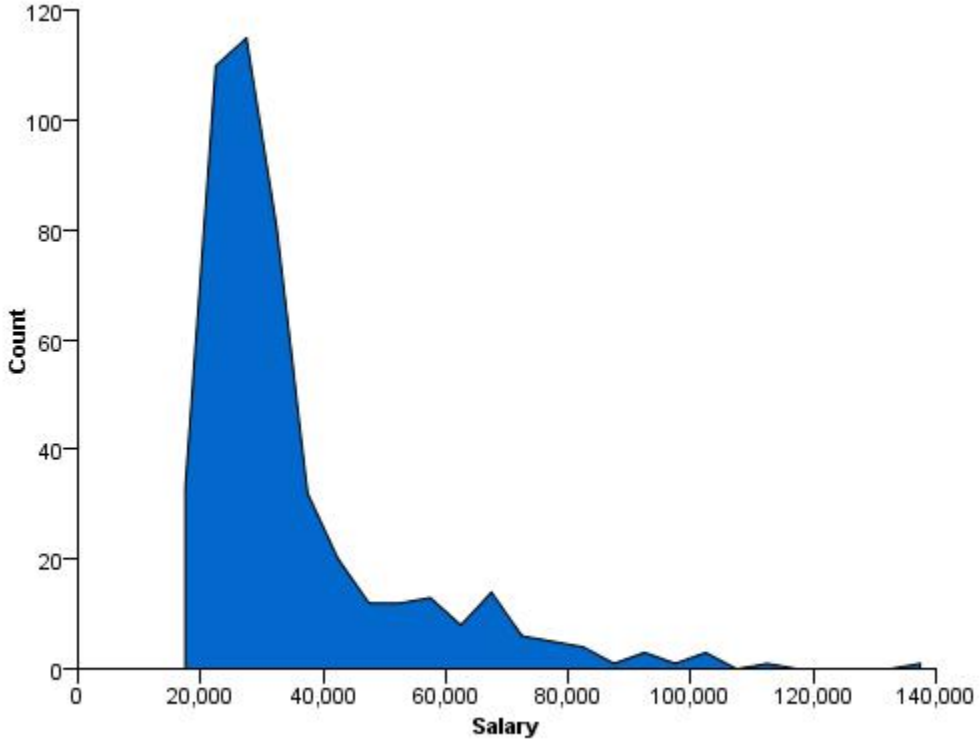


Figure 408. Frequency polygon

Stacked Histogram

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary=col(source(s), name("salary"))
DATA: gender=col(source(s), name("gender"), unit.category())
GUIDE: axis(dim(2), label("Count"))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: interval.stack(position(summary.count(bin.rect(salary))),
                        color(gender))
```

Figure 409. GPL for stacked histogram

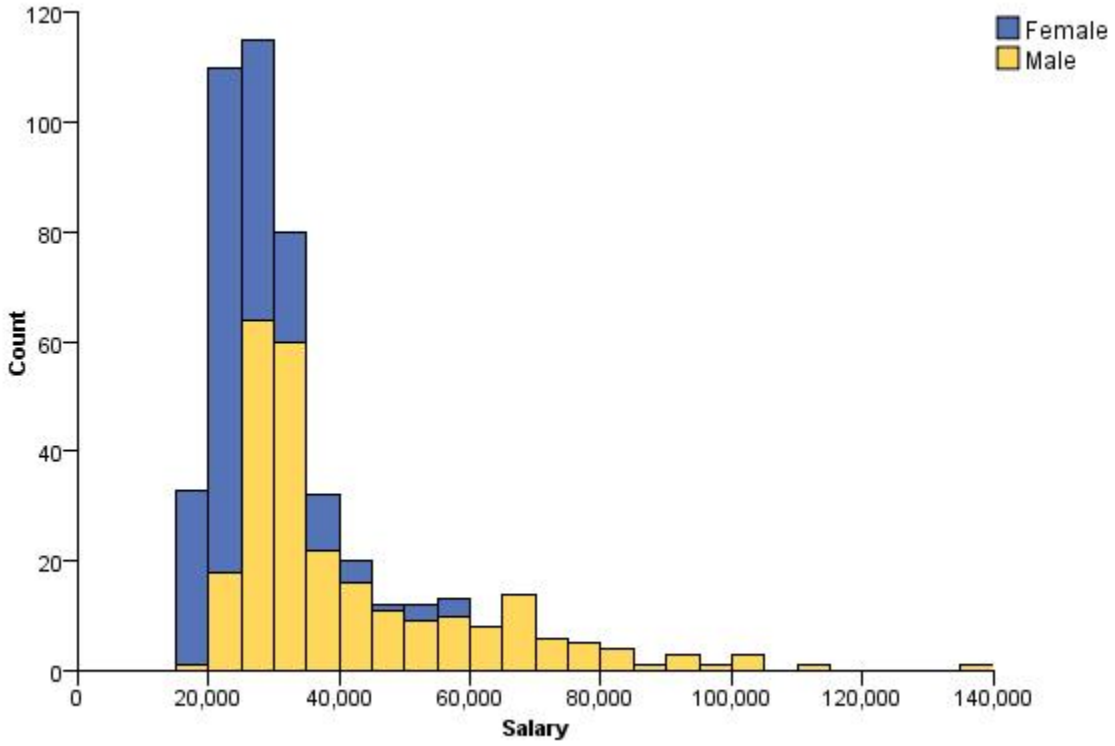


Figure 410. Stacked histogram

Faceted (Paneled) Histogram

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
DATA: gender = col(source(s), name("gender"), unit.category())
GUIDE: axis(dim(1), label("Salary"))
GUIDE: axis(dim(2), label("Count"))
GUIDE: axis(dim(4), label("Gender"))
ELEMENT: interval(position(summary.count(bin.rect(salary*1*1*gender))))
```

Figure 411. GPL for faceted histogram

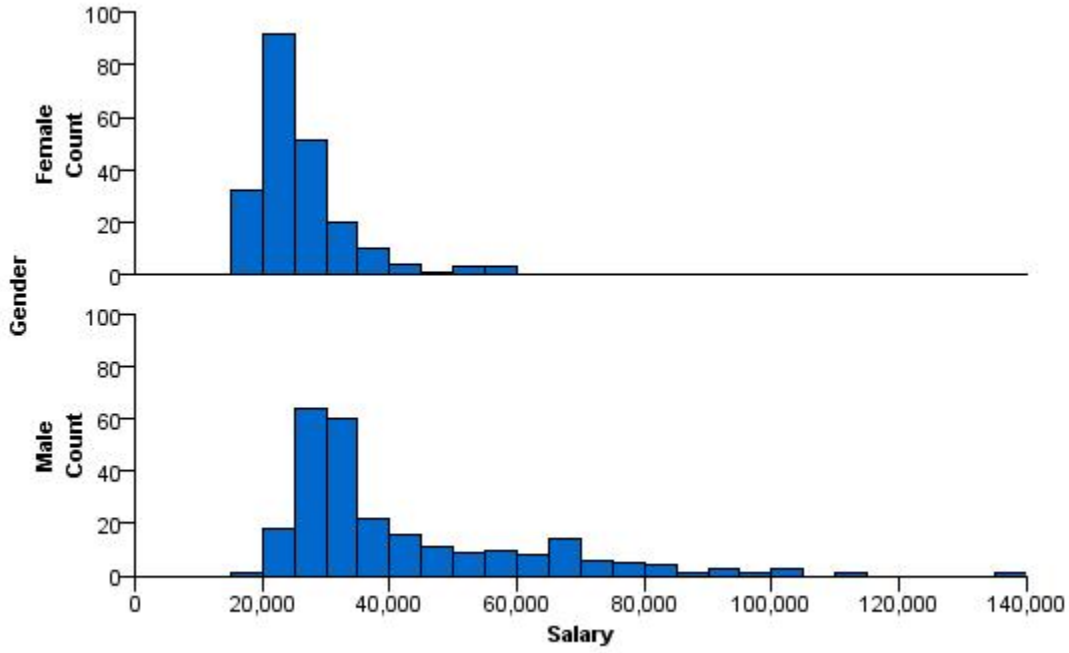


Figure 412. Faceted histogram

Population Pyramid

```
SOURCE: s = userSource(id("Employeedata"))
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: salary = col(source(s), name("salary"))
COORD: transpose(mirror())
GUIDE: axis(dim(1), label("Current Salary"))
GUIDE: axis(dim(1), label(""), opposite())
GUIDE: axis(dim(2), label("Frequency"))
GUIDE: axis(dim(3), label("Gender"), opposite(), gap(0px))
GUIDE: legend(aesthetic(aesthetic.color.interior), null())
ELEMENT: interval(position(summary.count(bin.rect(salary*1*gender))), color(gender))
```

Figure 413. GPL for population pyramid

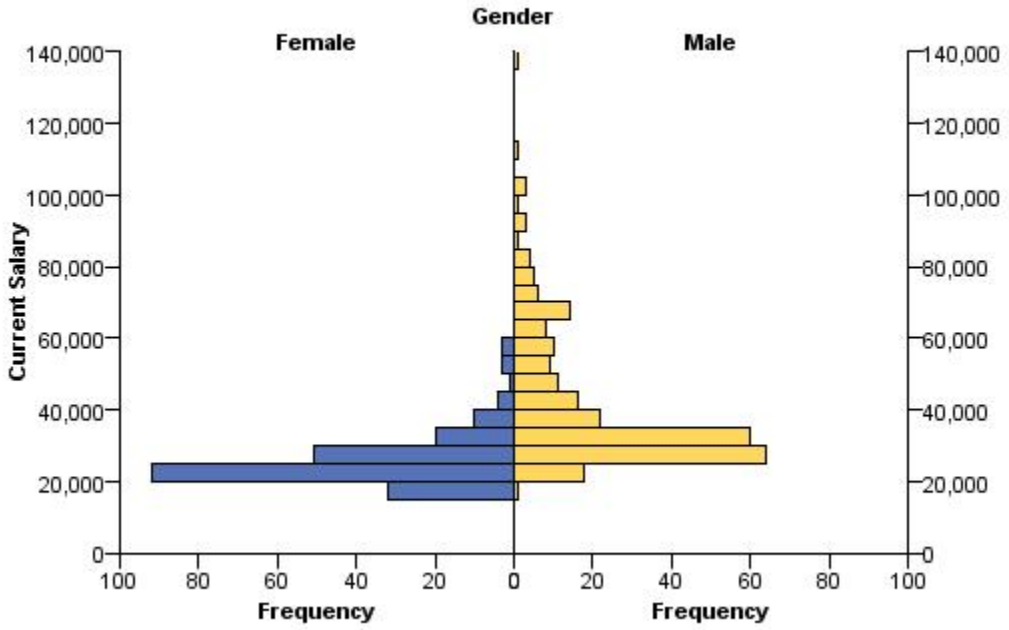


Figure 414. Population pyramid

Cumulative Histogram

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Cumulative Percent"))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: interval(position(summary.percent.count.cumulative(bin.rect(salary))))
```

Figure 415. GPL for cumulative histogram

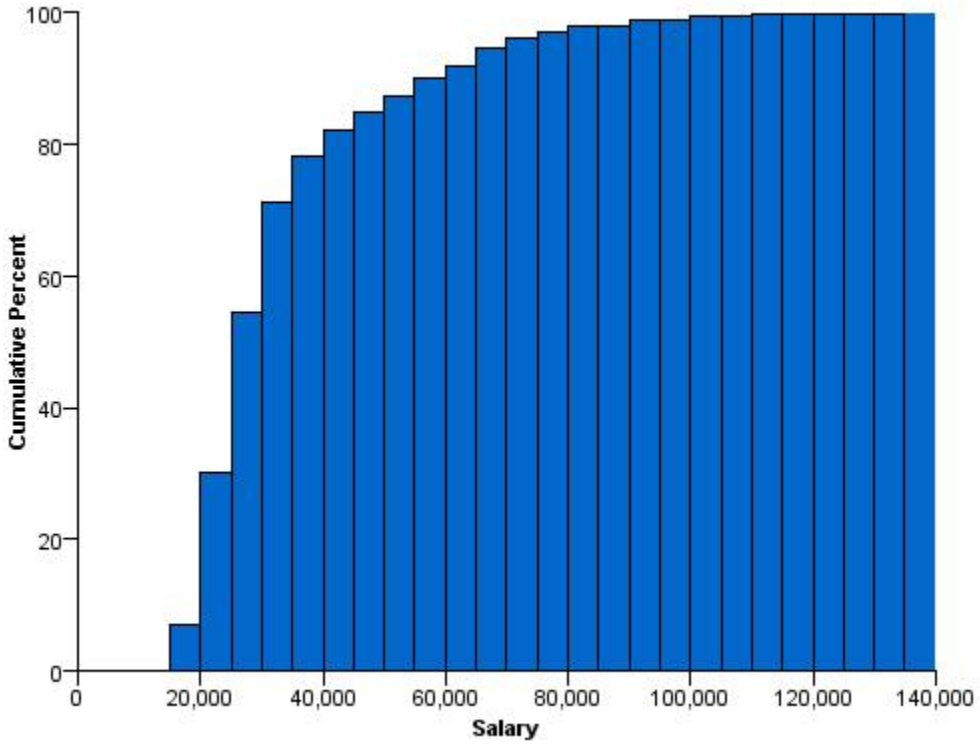


Figure 416. Cumulative histogram

3-D Histogram

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
COORD: rect(dim(1, 2, 3))
GUIDE: axis(dim(1), label("Employment Category"))
GUIDE: axis(dim(2), label("Salary"))
GUIDE: axis(dim(3), label("Count"))
ELEMENT: interval(position(summary.count(bin.rect(jobcat*salary, dim(2)))))
```

Figure 417. GPL for 3-D histogram

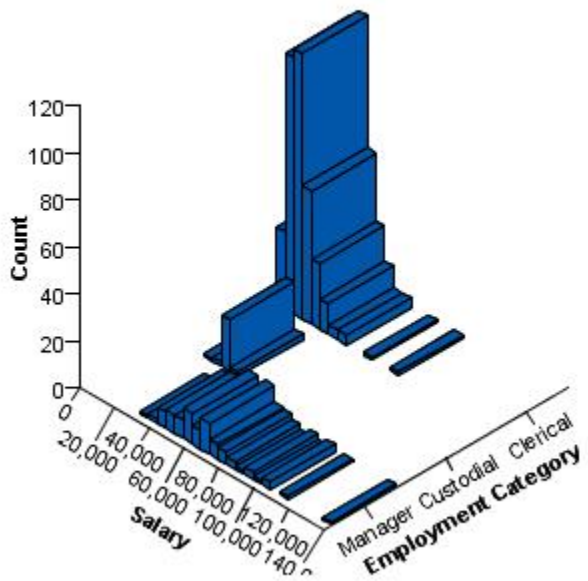


Figure 418. 3-D histogram

High-Low Chart Examples

This section provides examples of different types of high-low charts.

Simple Range Bar for One Variable

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), min(0.0))
GUIDE: axis(dim(2), label("Min Current Salary - Max Current Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(region.spread.range(jobcat*salary)))
```

Figure 419. GPL for simple range bar for one variable

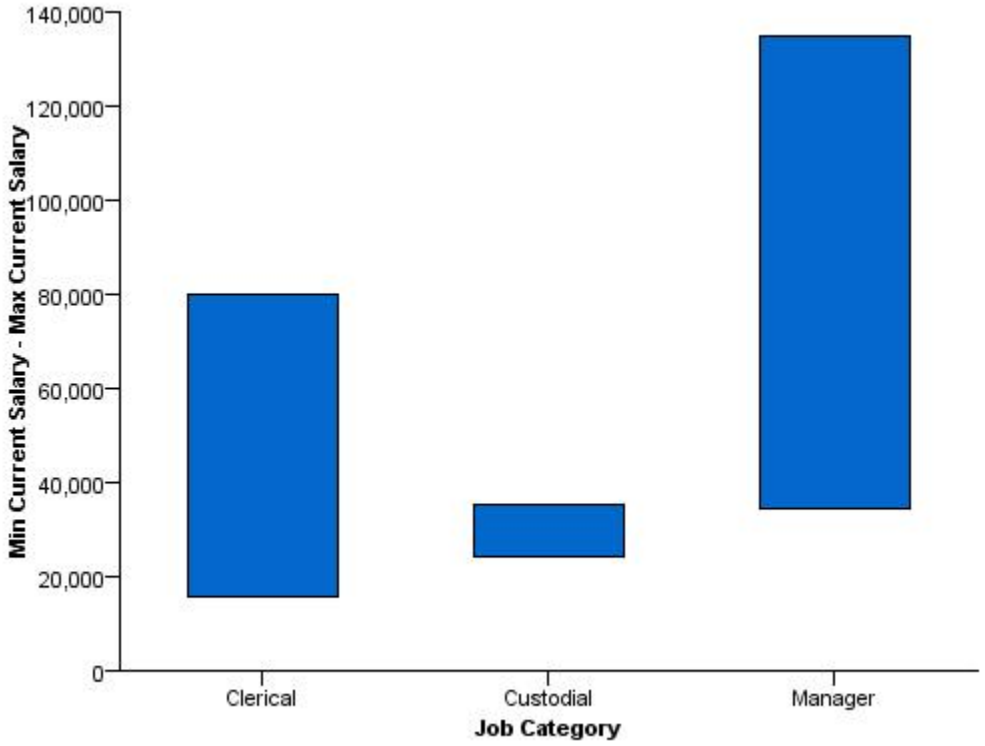


Figure 420. Simple range bar for one variable

Simple Range Bar for Two Variables

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
SCALE: linear(dim(2), min(0.0))
GUIDE: axis(dim(2), label("Min Beginning Salary - Max Current Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: interval(position(region.spread.range(jobcat*(salbegin+salary))))
```

Figure 421. GPL for simple range bar for two variables

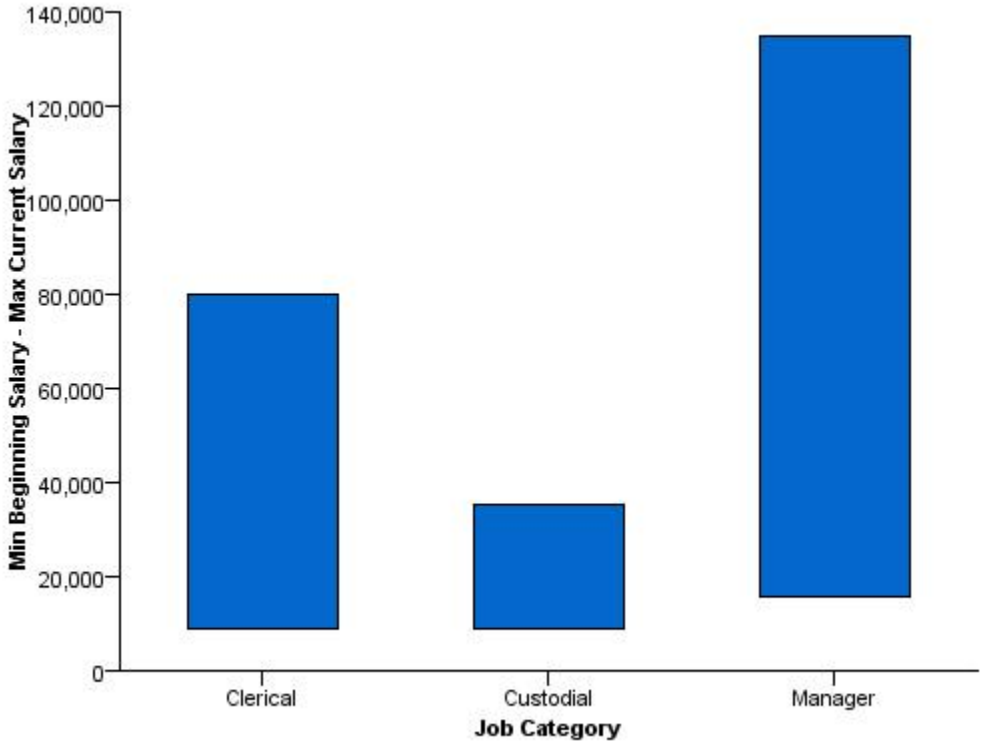


Figure 422. Simple range bar for two variables

High-Low-Close Chart

```
SOURCE: s = userSource(id("stocks"))
DATA: Date = col(source(s), name("Date"), unit.time(), format("MM/dd/yy"))
DATA: Close = col(source(s), name("Close"))
DATA: High = col(source(s), name("High"))
DATA: Low = col(source(s), name("Low"))
GUIDE: axis(dim(1), label("Date"))
GUIDE: axis(dim(2), label("Close"))
SCALE: time(dim(1), dataMaximum())
ELEMENT: interval(position(region.spread.range(Date*(Low+High))))
ELEMENT: point(position(Date*Close), color.exterior(color.red), size(size."2px"))
```

Figure 423. GPL for high-low-close chart

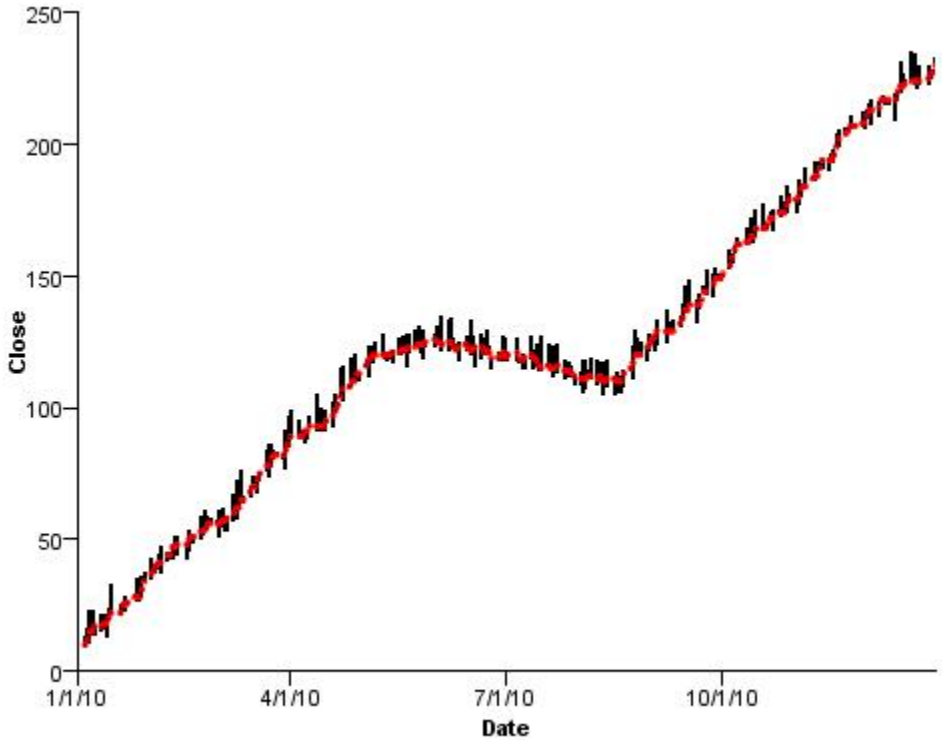


Figure 424. High-low-close chart

Scatter/Dot Examples

This section provides examples of different types of scatterplots and dot plots.

Simple 1-D Scatterplot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
COORD: rect(dim(1))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: point(position(salary))
```

Figure 425. GPL for simple 1-D scatterplot

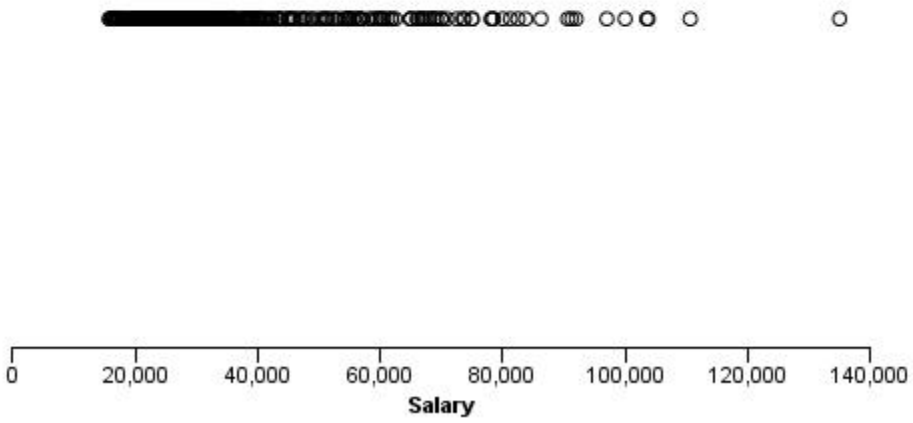


Figure 426. Simple 1-D scatterplot

Simple 2-D Scatterplot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: point(position(salbegin*salary))
```

Figure 427. GPL for simple 2-D scatterplot

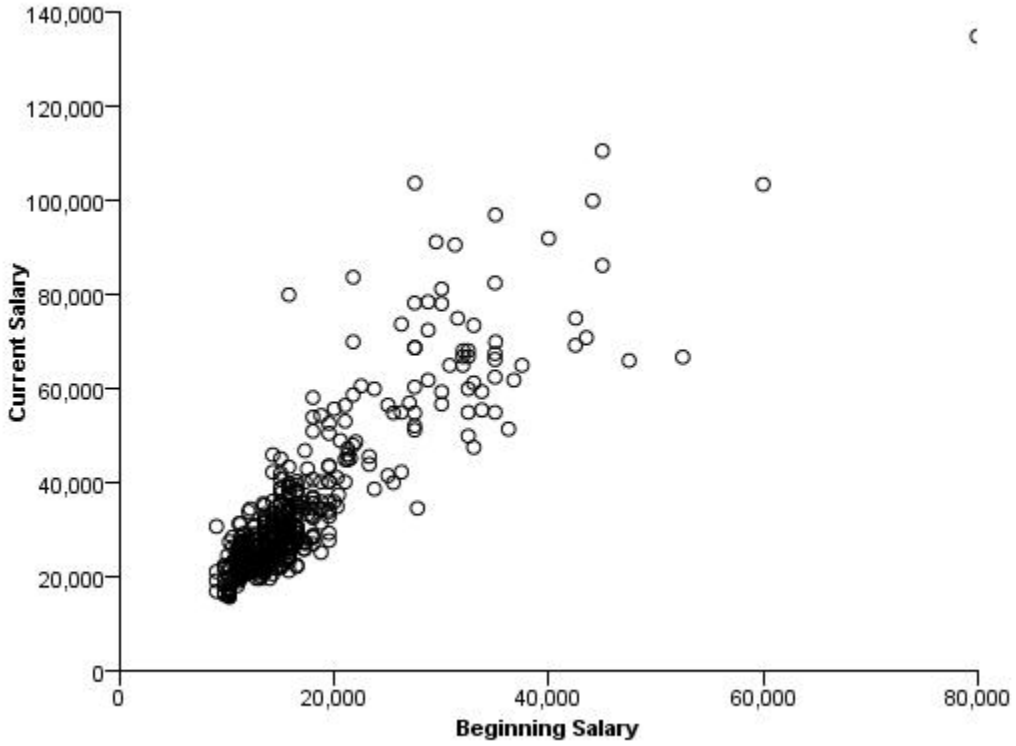


Figure 428. Simple 2-D scatterplot

Simple 2-D Scatterplot with Fit Line

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: point(position(salbegin*salary))
ELEMENT: line(position(smooth.linear(salbegin*salary)))
```

Figure 429. GPL for simple scatterplot with fit line

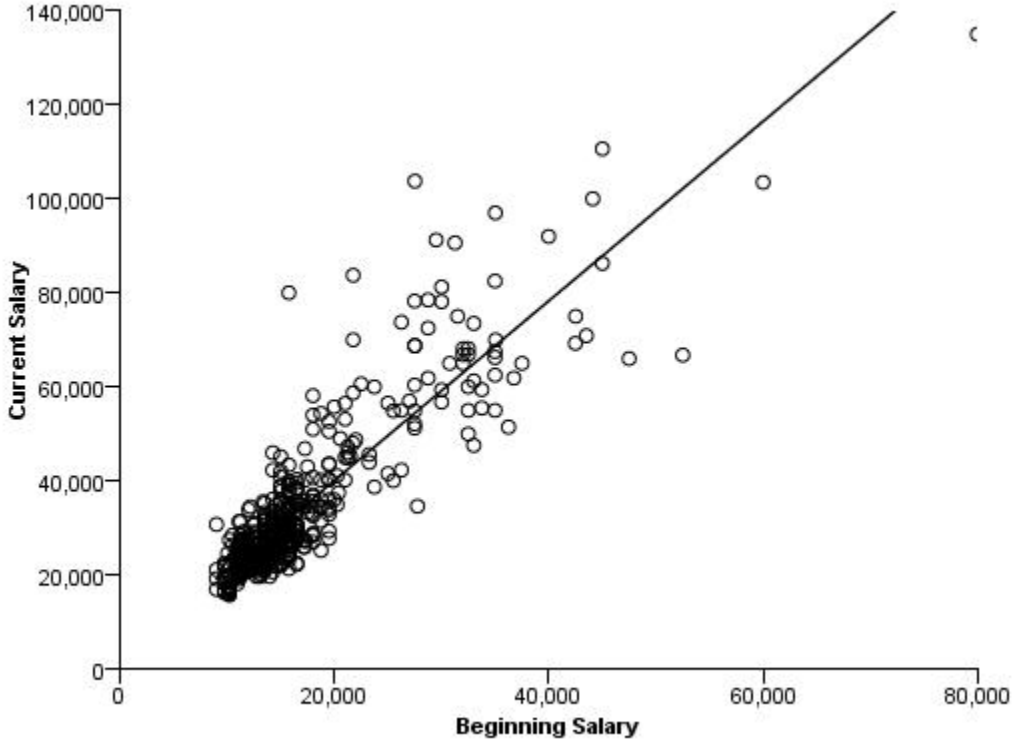


Figure 430. Simple scatterplot with fit line

Grouped Scatterplot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
DATA: gender=col(source(s), name("gender"), unit.category())
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: point(position(salbegin*salary), color(gender))
```

Figure 431. GPL for grouped scatterplot

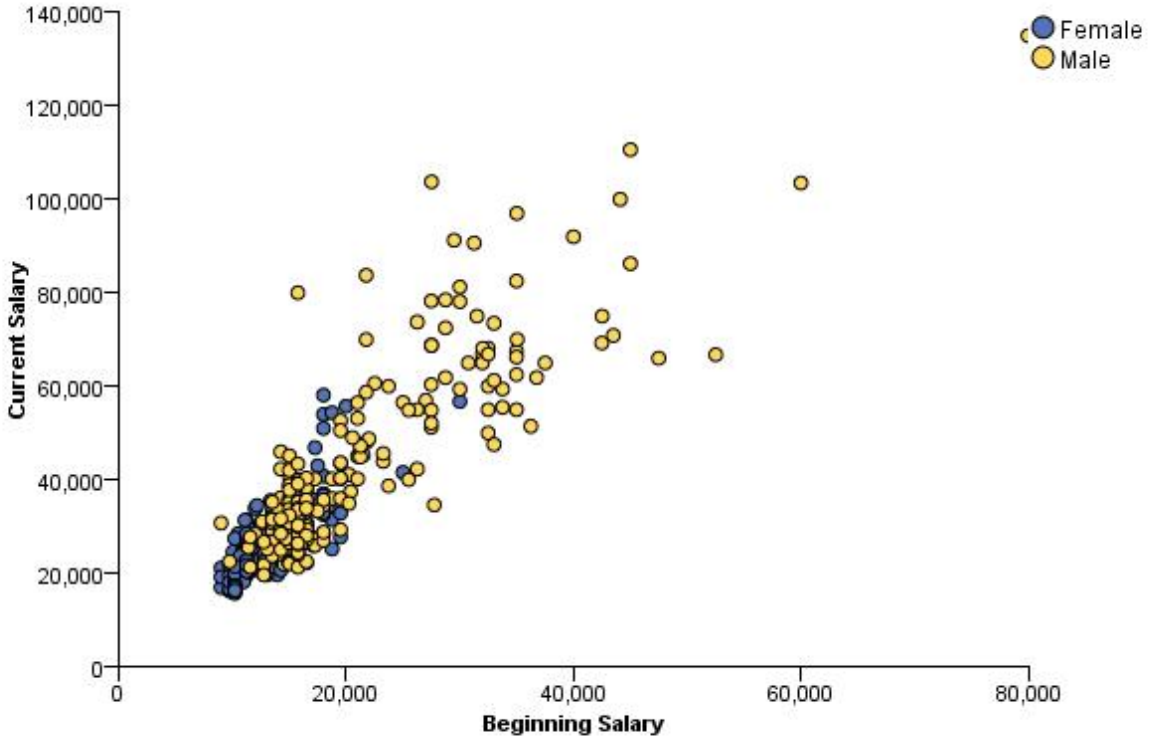


Figure 432. Grouped scatterplot

Grouped Scatterplot with Convex Hull

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
DATA: gender=col(source(s), name("gender"), unit.category())
GUIDE: axis(dim(1), label("Beginning Salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: legend(aesthetic(aesthetic.color.exterior), label("Gender"))
GUIDE: legend(aesthetic(aesthetic.color.interior), null())
ELEMENT: point(position(salbegin*salary), color.exterior(gender))
ELEMENT: edge(position(link.hull(salbegin*salary)), color.interior(gender))
```

Figure 433. GPL for grouped scatterplot with convex hull

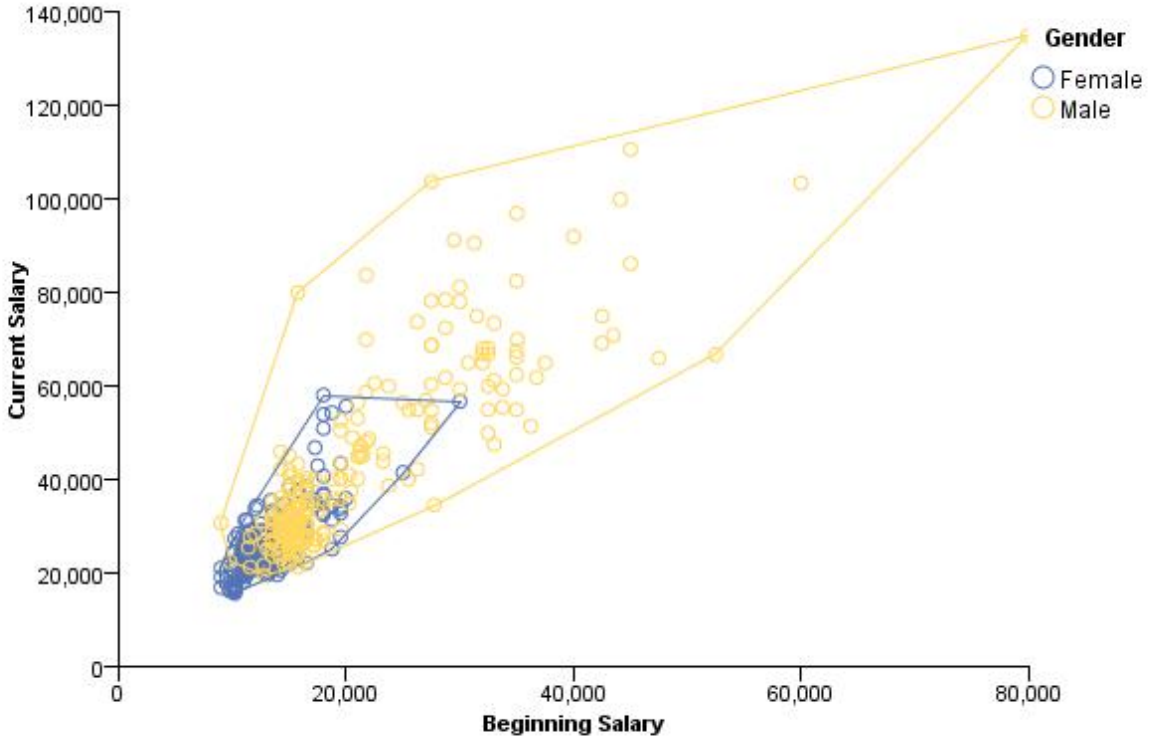


Figure 434. Grouped scatterplot with convex hull

Scatterplot Matrix (SPLOM)

```
SOURCE: s = userSource(id("customer_subset"))
DATA: age=col(source(s), name("age"))
DATA: income=col(source(s), name("income"))
DATA: creddebt=col(source(s), name("creddebt"))
GUIDE: axis(dim(1.1), ticks(null()))
GUIDE: axis(dim(2.1), ticks(null()))
GUIDE: axis(dim(1), gap(0px))
GUIDE: axis(dim(2), gap(0px))
ELEMENT: point(position((age/"Age"+income/"Income"+creddebt/"CC Debt")*
                        (age/"Age"+income/"Income"+creddebt/"CC Debt")))
```

Figure 435. GPL for scatterplot matrix

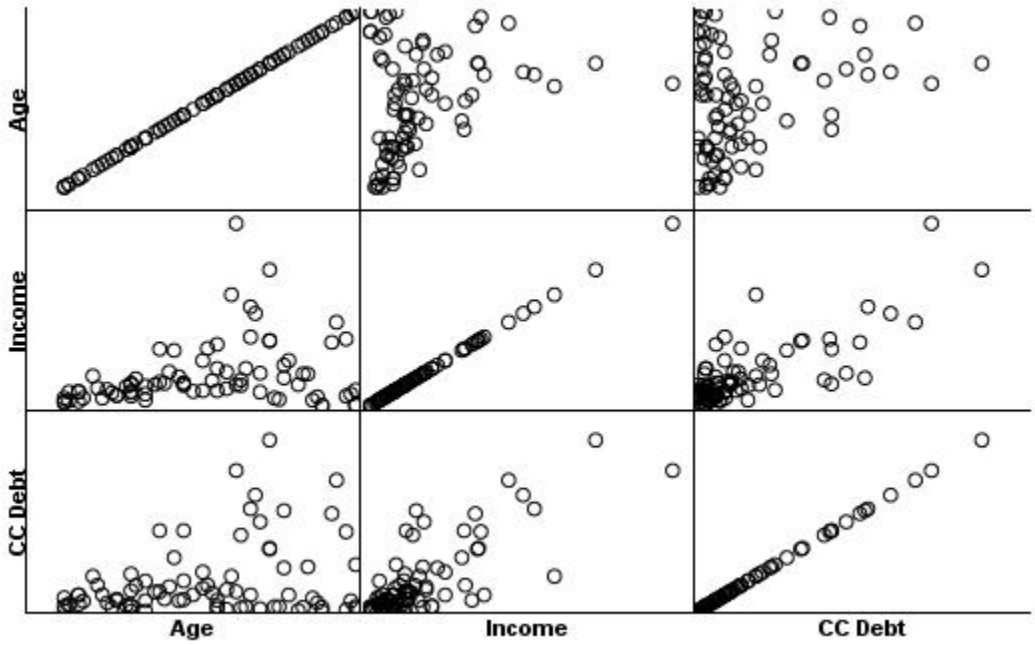


Figure 436. Scatterplot Matrix

Bubble Plot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
DATA: prevexp=col(source(s), name("prevexp"))
SCALE: linear(aesthetic(aesthetic.size),
            aestheticMinimum(size."5px"), aestheticMaximum(size."35px"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
GUIDE: legend(aesthetic(aesthetic.size), label("Previous Experience (months)"))
ELEMENT: point(position(salbegin*salary), size(prevexp))
```

Figure 437. GPL for bubble plot

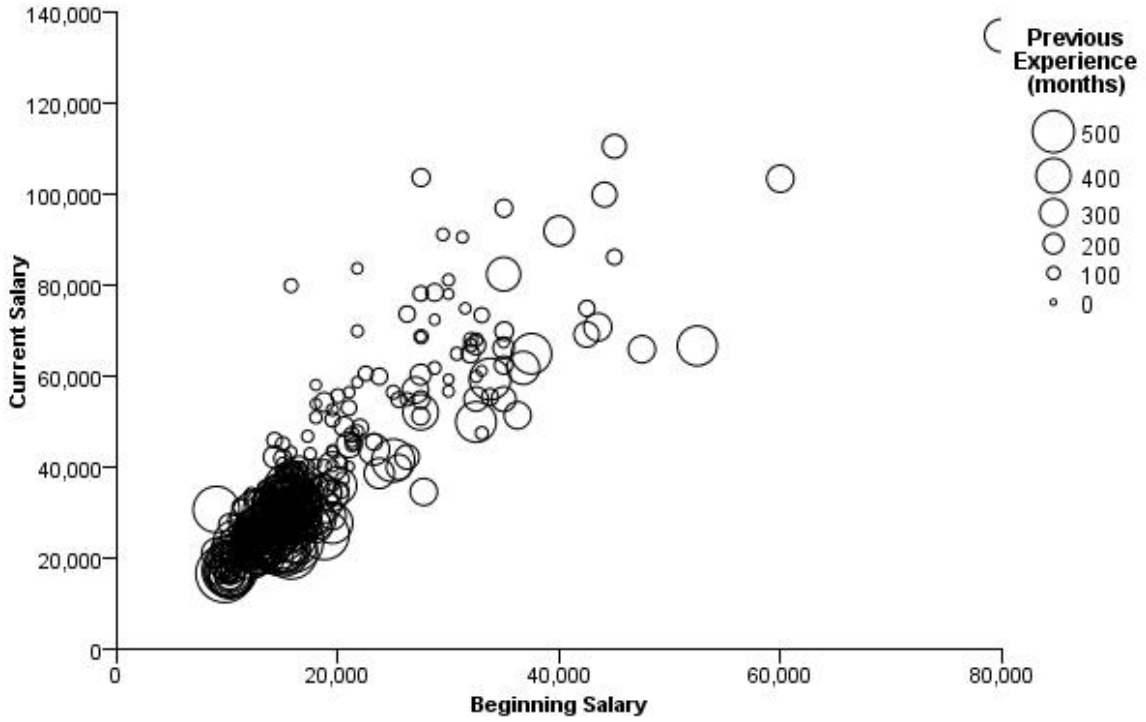


Figure 438. Bubble plot

Binned Scatterplot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
GUIDE: legend(aesthetic(aesthetic.color), label("Count"))
ELEMENT: point(position(bin.rect(salbegin*salary, dim(1,2))),
               color(summary.count()))
```

Figure 439. GPL for binned scatterplot

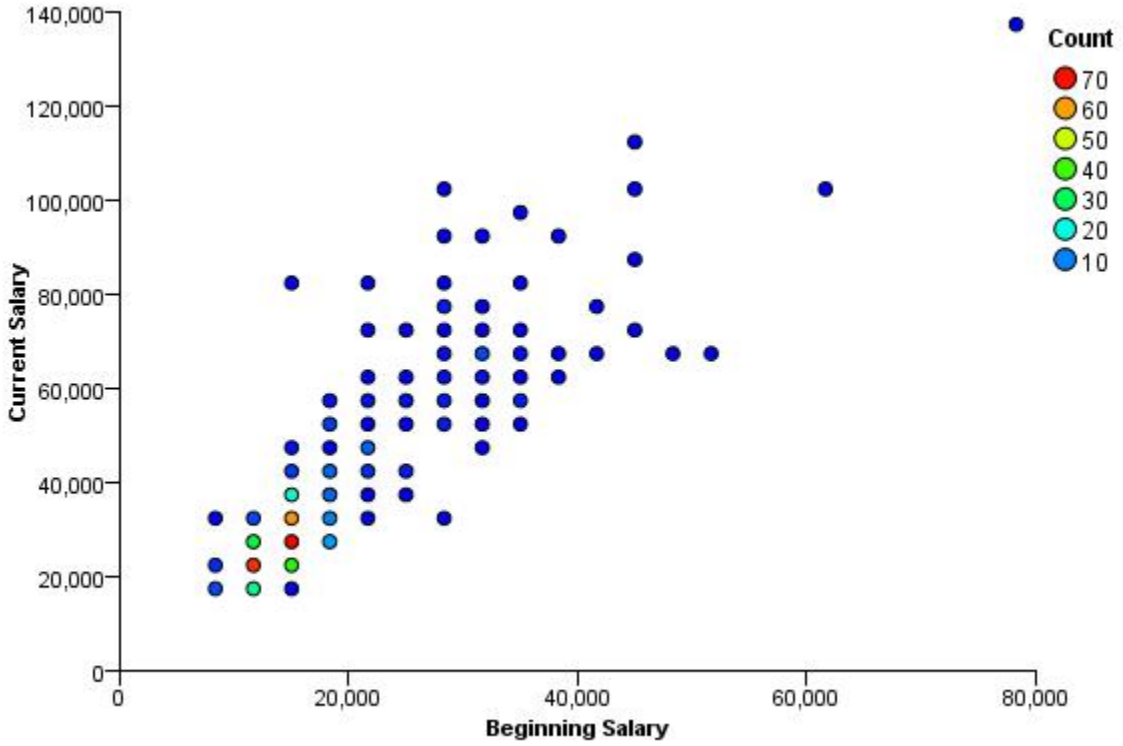


Figure 440. Binned scatterplot

Binned Scatterplot with Polygons

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
GUIDE: legend(aesthetic(aesthetic.color), label("Count"))
ELEMENT: polygon(position(bin.hex(salbegin*salary, dim(1,2))),
                 color(summary.count()))
```

Figure 441. GPL for binned scatterplot with polygons

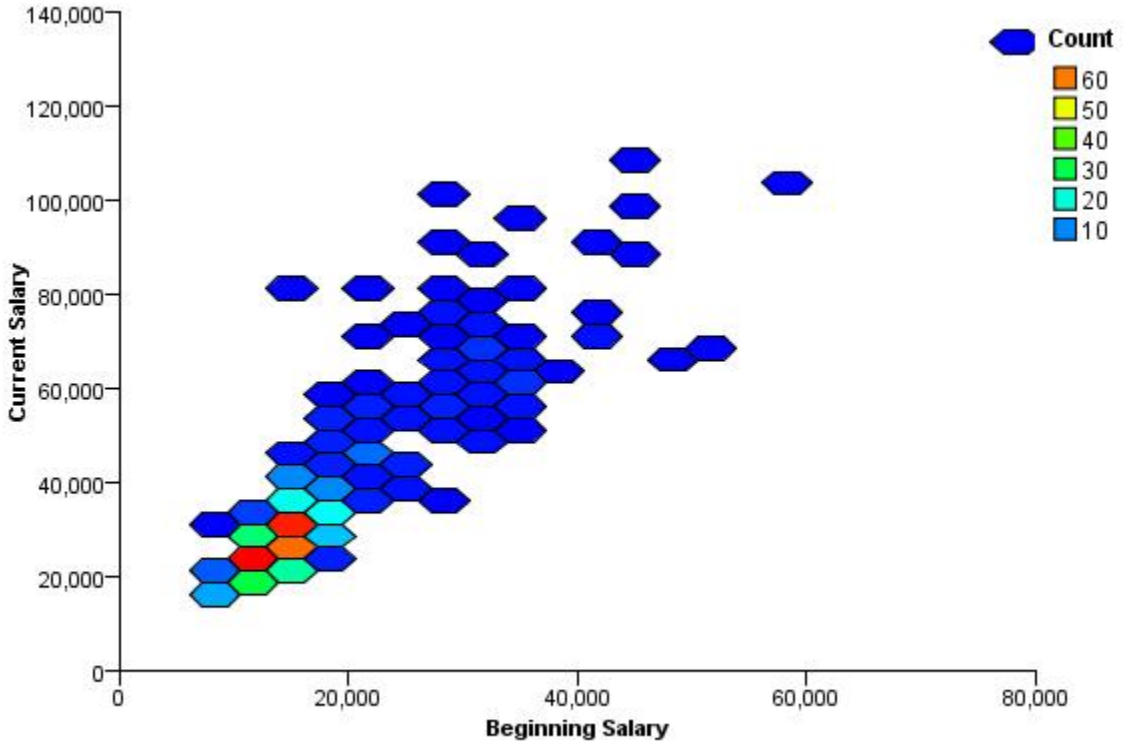
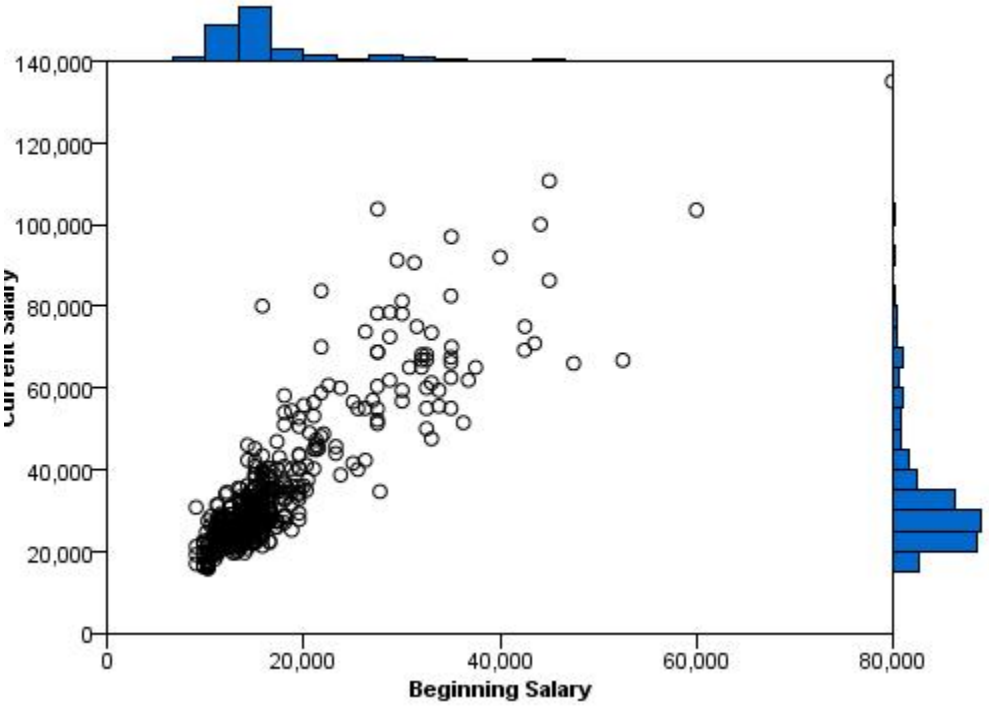


Figure 442. Binned scatterplot with polygons

Scatterplot with Border Histograms

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
DATA: salbegin = col(source(s), name("salbegin"))
GRAPH: begin(origin(5%, 10%), scale(85%, 85%))
GUIDE: axis(dim(1), label("Beginning Salary"))
GUIDE: axis(dim(2), label("Current Salary"))
ELEMENT: point(position(salbegin*salary))
GRAPH: end()
GRAPH: begin(origin(5%, 0%), scale(85%, 10%))
GUIDE: axis(dim(1), ticks(null()))
GUIDE: axis(dim(2), null())
ELEMENT: interval(position(summary.count(bin.rect(salbegin))))
GRAPH: end()
GRAPH: begin(origin(90%, 10%), scale(10%, 85%))
COORD: rect(dim(1, 2), transpose())
GUIDE: axis(dim(1), ticks(null()))
GUIDE: axis(dim(2), null())
ELEMENT: interval(position(summary.count(bin.rect(salary))))
GRAPH: end()
```

Figure 443. GPL for scatterplot with border histograms



Scatterplot with Border Boxplots

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
DATA: salbegin = col(source(s), name("salbegin"))
GRAPH: begin(origin(5%, 10%), scale(85%, 85%))
GUIDE: axis(dim(1), label("Beginning Salary"))
GUIDE: axis(dim(2), label("Current Salary"))
ELEMENT: point(position(salbegin*salary))
GRAPH: end()
GRAPH: begin(origin(5%, 0%), scale(85%, 10%))
COORD: rect(dim(1))
GUIDE: axis(dim(1), ticks(null()))
ELEMENT: schema(position(bin.quantile.letter(salbegin)), size(size."80%"))
GRAPH: end()
GRAPH: begin(origin(90%, 10%), scale(10%, 85%))
COORD: transpose(rect(dim(1)))
GUIDE: axis(dim(1), ticks(null()))
ELEMENT: schema(position(bin.quantile.letter(salary)), size(size."80%"))
GRAPH: end()
```

Figure 444. GPL for scatterplot with border boxplots

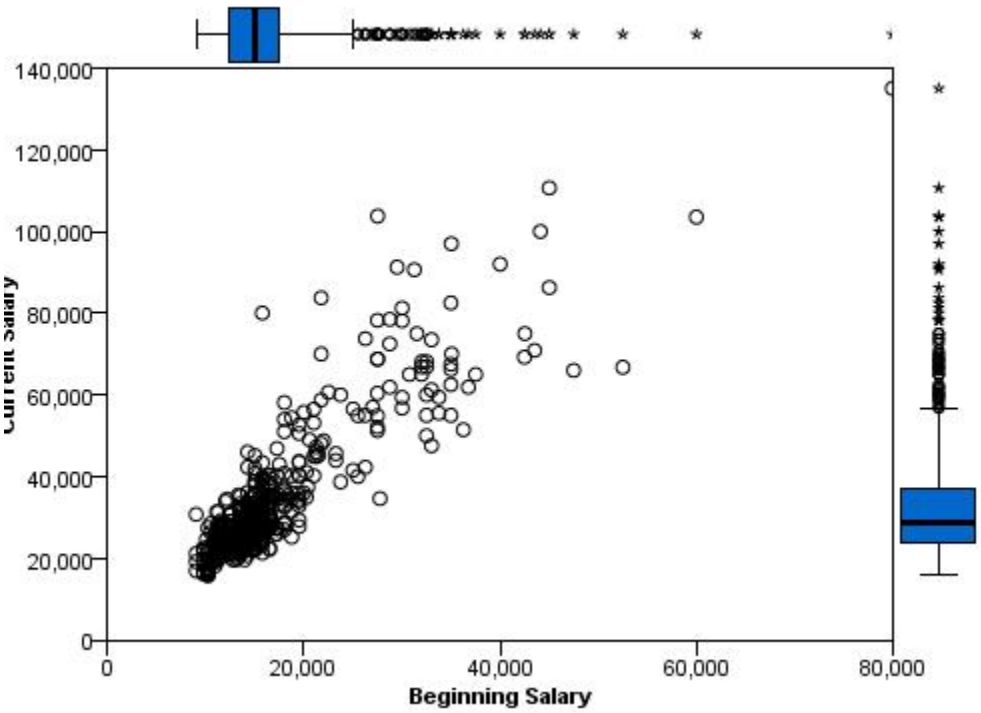


Figure 445. Scatterplot with border boxplots

Dot Plot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
COORD: rect(dim(1))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: point.dodge.asymmetric(position(bin.dot(salary)))
```

Figure 446. GPL for dot plot

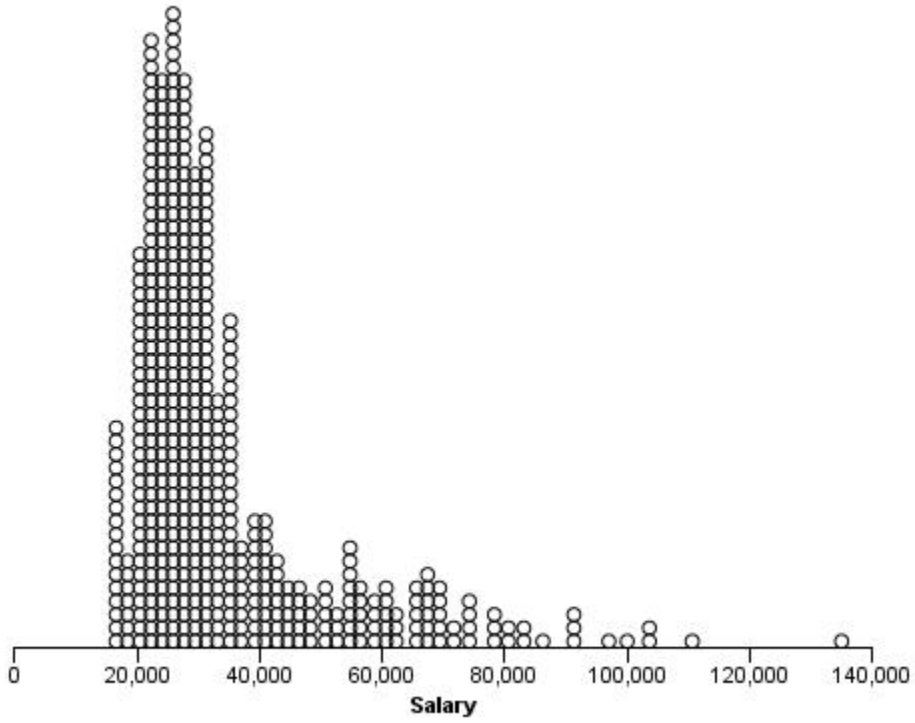


Figure 447. Dot plot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: salary = col(source(s), name("salary"))
GUIDE: axis(dim(1), label("Salary"))
COORD: rect(dim(1))
ELEMENT: point.dodge.asymmetric(position(bin.dot(salary)),
                                color(gender), shape(shape.square))
```

Figure 448. GPL for grouped dot plot

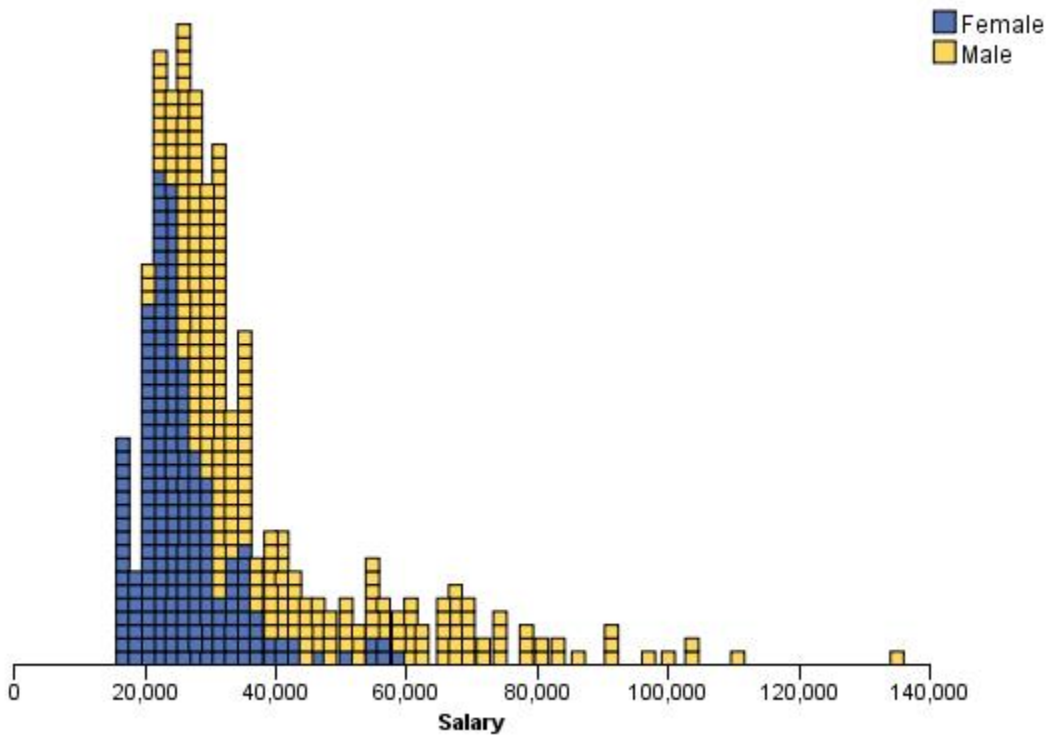


Figure 449. Grouped dot plot

2-D Dot Plot

```
SOURCE: s = userSource(id("customer_subset"))
DATA: region = col(source(s), name("region"), unit.category())
DATA: income = col(source(s), name("income"))
GUIDE: axis(dim(1), label("Income"))
GUIDE: axis(dim(2), label("Region"))
ELEMENT: point.dodge.symmetric(position(bin.dot(income*region)))
```

Figure 450. GPL for 2-D dot plot

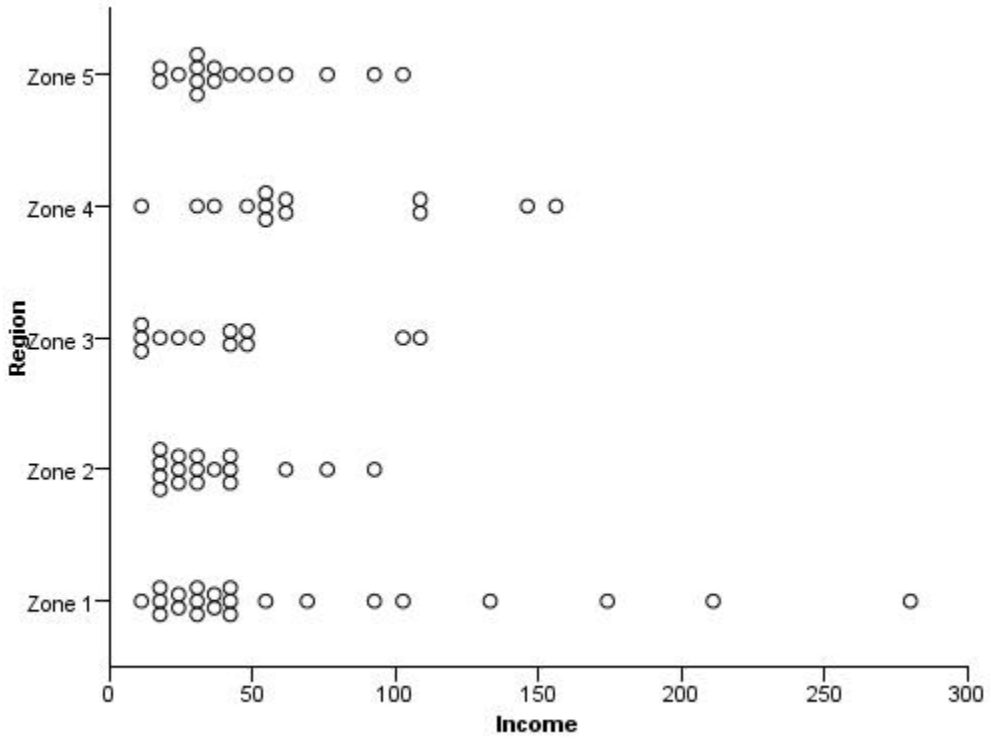


Figure 451. 2-D dot plot

```
SOURCE: s = userSource(id("customer_subset"))
DATA: region = col(source(s), name("region"), unit.category())
DATA: income = col(source(s), name("income"))
GUIDE: axis(dim(1), label("Region"))
GUIDE: axis(dim(2), label("Income"))
ELEMENT: point.dodge.symmetric(position(bin.dot(region*income, dim(2))))
```

Figure 452. GPL for alternate 2-D dot plot

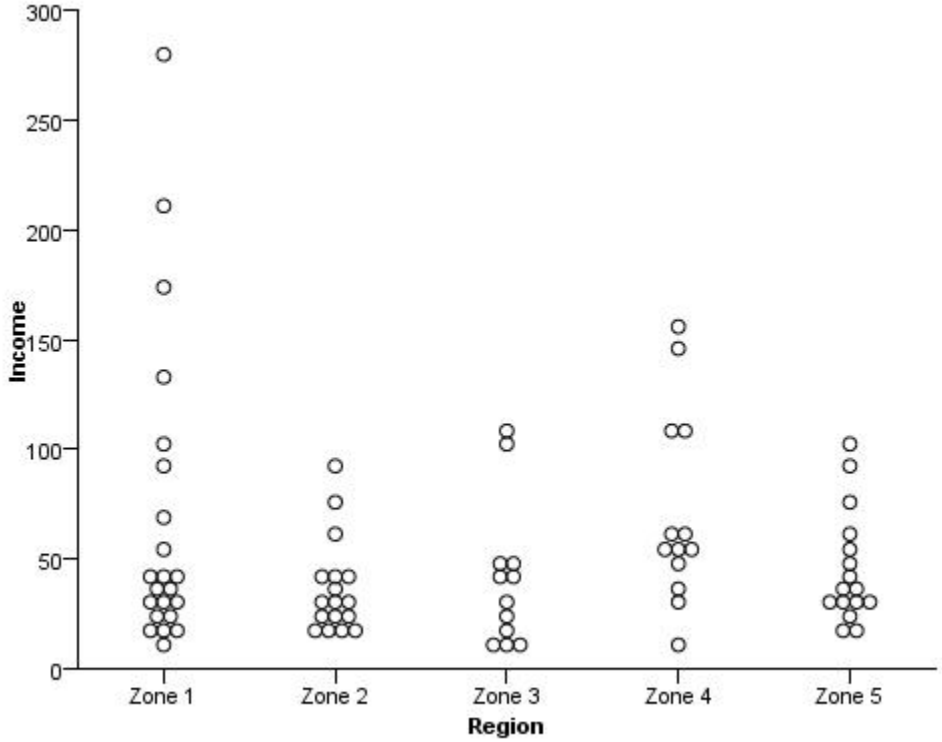


Figure 453. Alternate 2-D dot plot

Jittered Categorical Scatterplot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
GUIDE: axis(dim(2), label("Gender"))
GUIDE: axis(dim(1), label("Employment Category"))
ELEMENT: point.jitter(position(jobcat*gender))
```

Figure 454. GPL for jittered categorical scatterplot

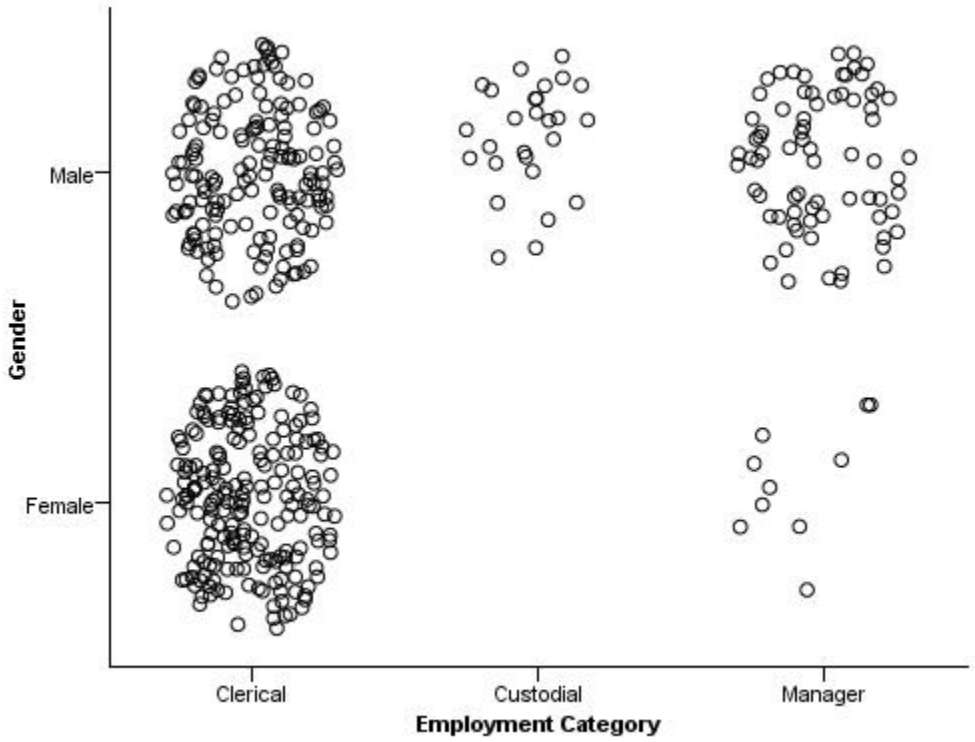


Figure 455. Jittered categorical scatterplot

Line Chart Examples

This section provides examples of different types of line charts.

Simple Line Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: line(position(salbegin*salary))
```

Figure 456. GPL for simple line chart

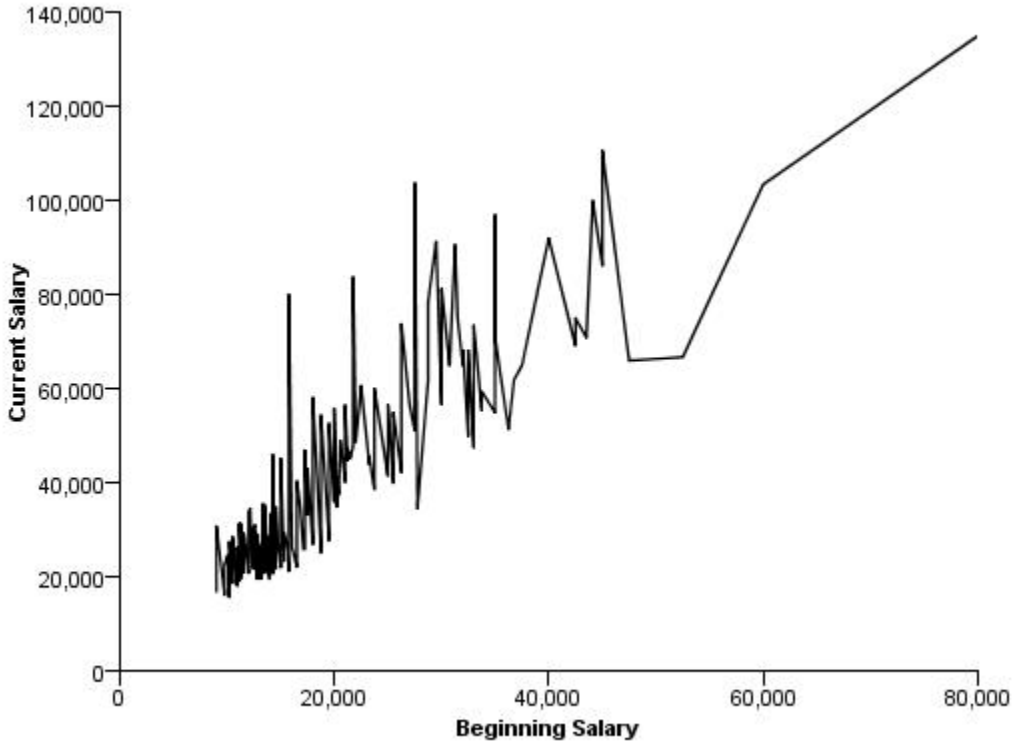


Figure 457. Simple line chart

Simple Line Chart with Points

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: line(position(salbegin*salary))
ELEMENT: point(position(salbegin*salary))
```

Figure 458. GPL for simple line chart with points

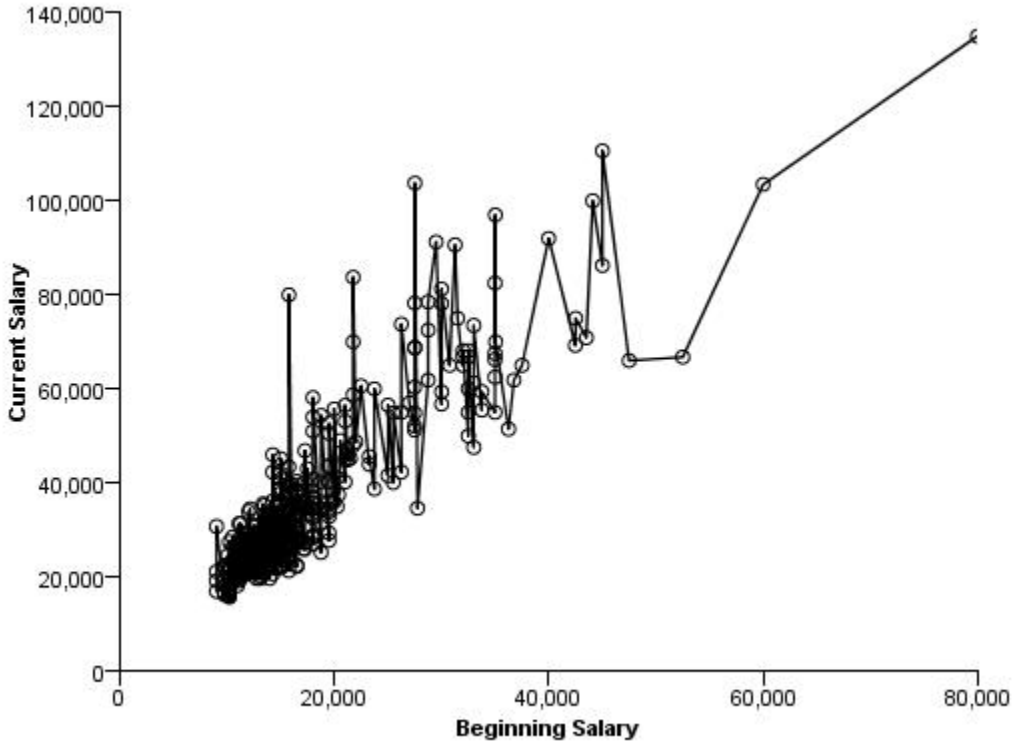


Figure 459. Simple line chart with points

Line Chart of Date Data

```
SOURCE: s = userSource(id("stocks"))
DATA: Date = col(source(s), name("Date"), unit.time(), format("MM/dd/yy"))
DATA: Close = col(source(s), name("Close"))
GUIDE: axis(dim(1), label("Date"))
GUIDE: axis(dim(2), label("Close"))
SCALE: time(dim(1), dataMaximum())
ELEMENT: line(position(Date*Close))
```

Figure 460. GPL for line chart of date data

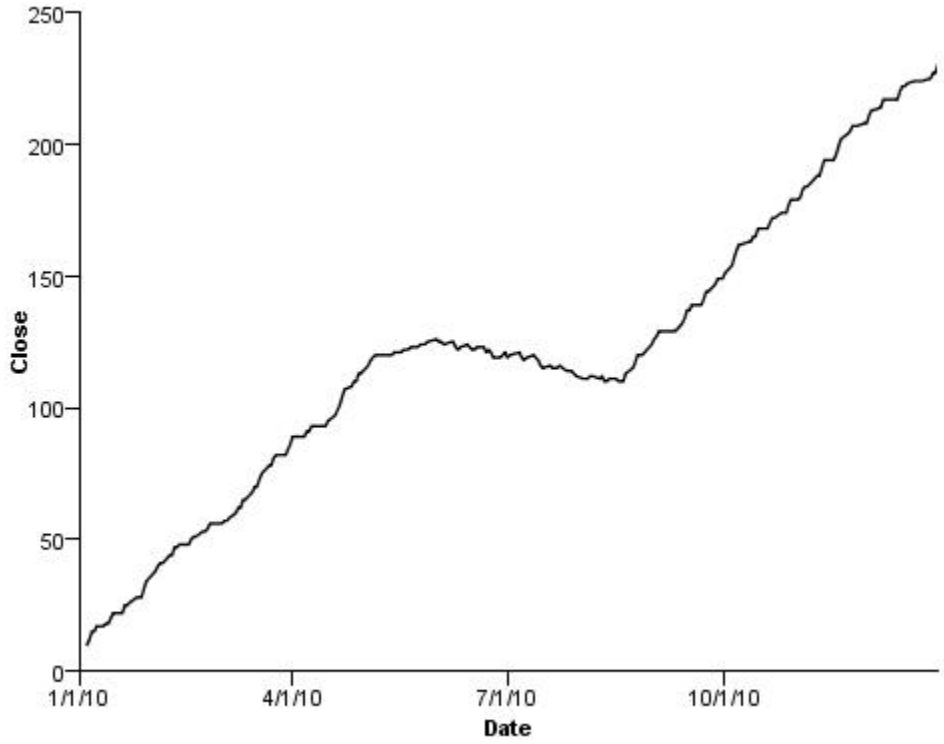


Figure 461. Line chart of date data

Line Chart With Step Interpolation

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: line(position(smooth.step.center(salbegin*salary)))
```

Figure 462. GPL for line chart with step interpolation

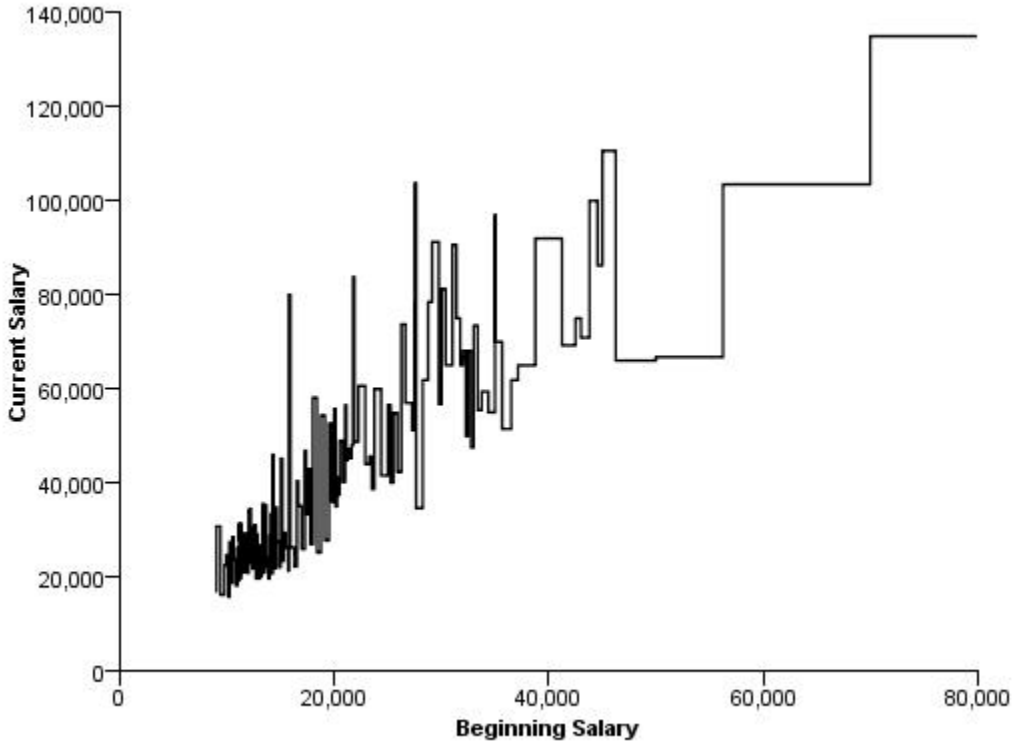


Figure 463. Line chart with step interpolation

Fit Line

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: point(position(salbegin*salary))
ELEMENT: line(position(smooth.linear(salbegin*salary)))
```

Figure 464. GPL for linear fit line overlaid on scatterplot

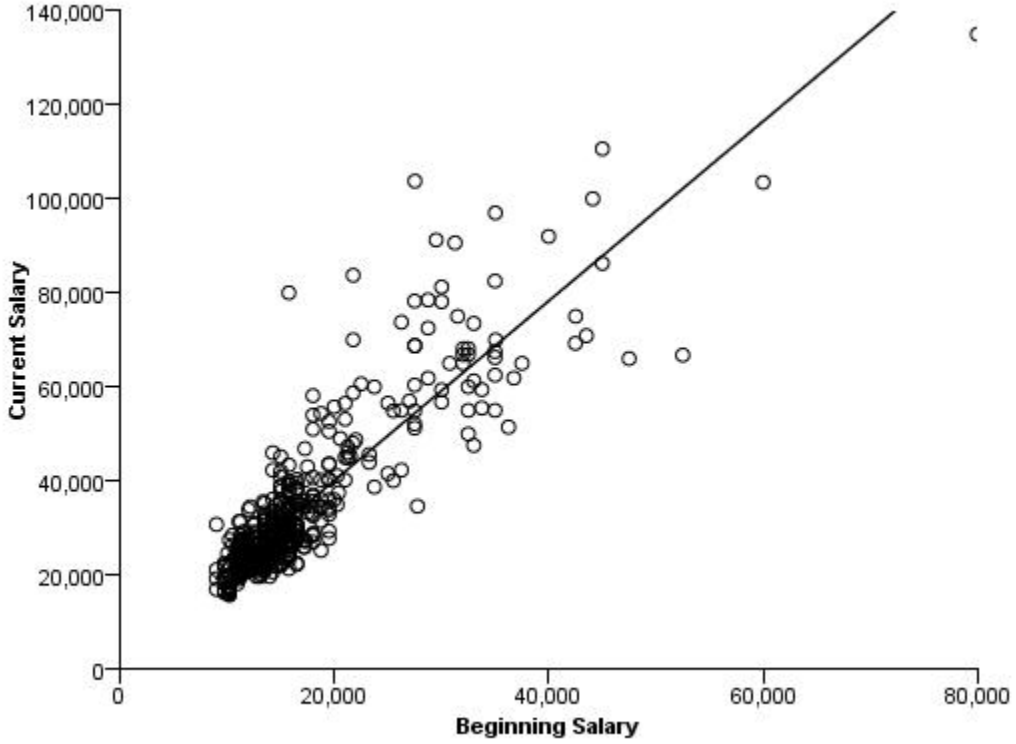


Figure 465. Linear fit line overlaid on scatterplot

Line Chart from Equation

```
DATA: x = iter(0,10,0.01)
TRANS: y = eval(sin(x)*cos(x)*(x-5))
ELEMENT: line(position(x*y))
```

Figure 466. GPL for line chart from equation

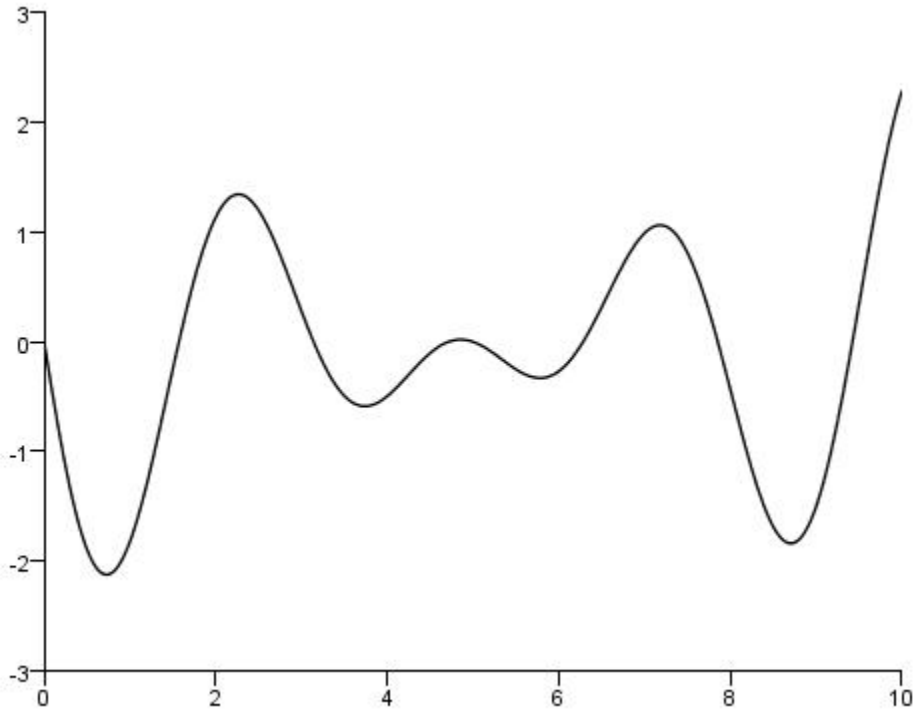


Figure 467. Line chart from equation

```
DATA: x = iter(0,6.28,0.01)
TRANS: y = eval(cos(6*x))
COORD: polar()
SCALE: linear(dim(1), min(0.0), max(6.28))
GUIDE: axis(dim(2), null())
ELEMENT: line(position(x*y))
```

Figure 468. GPL for line chart from equation in polar coordinates

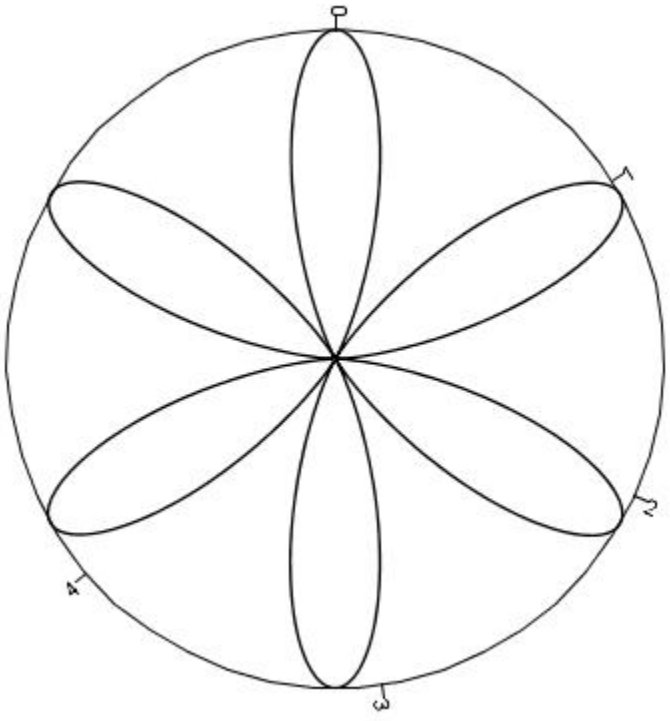


Figure 469. Line chart from equation in polar coordinates

Line Chart with Separate Scales

```
SOURCE: s = userSource(id("stocks"))
DATA: date = col(source(s), name("Date"), unit.time(), format("MM/dd/yy"))
DATA: close = col(source(s), name("Close"))
DATA: volume = col(source(s), name("Volume"))
SCALE: time(dim(1), dataMaximum())
ELEMENT: line(position(date*(close/"Close"+volume/"Volume")))
```

Figure 470. GPL for line chart with separate scales

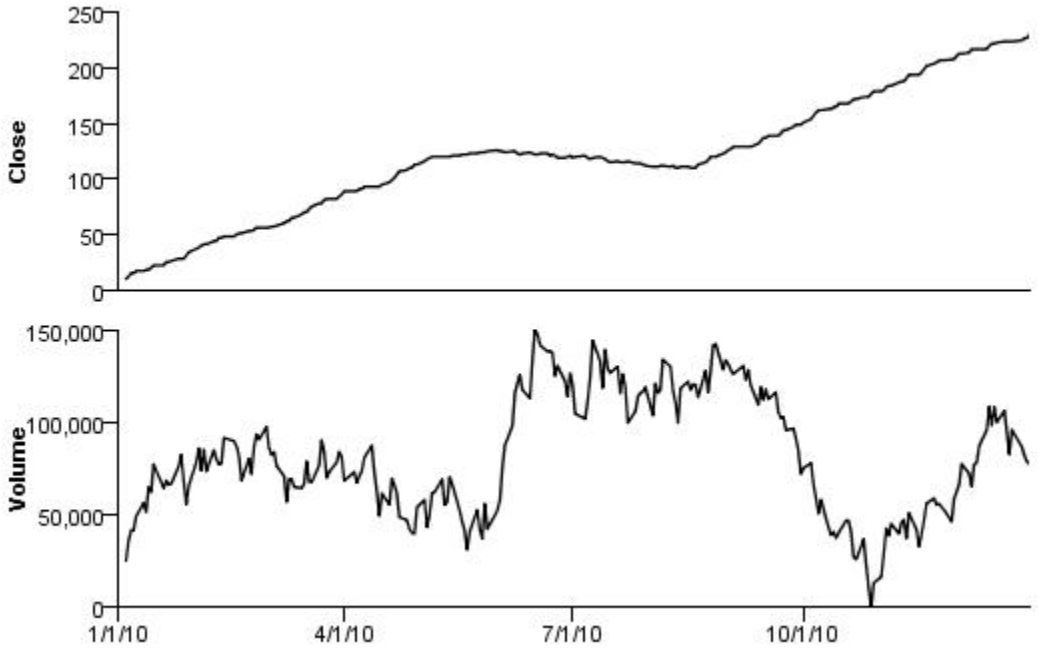


Figure 471. Line chart with separate scales

Pie Chart Examples

This section provides examples of different types of pie charts.

Pie Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
COORD: polar.theta()
SCALE: linear(dim(1), dataMinimum(), dataMaximum())
GUIDE: axis(dim(1), null())
ELEMENT: interval.stack(position(summary.count(1)), color(jobcat))
```

Figure 472. GPL for pie chart

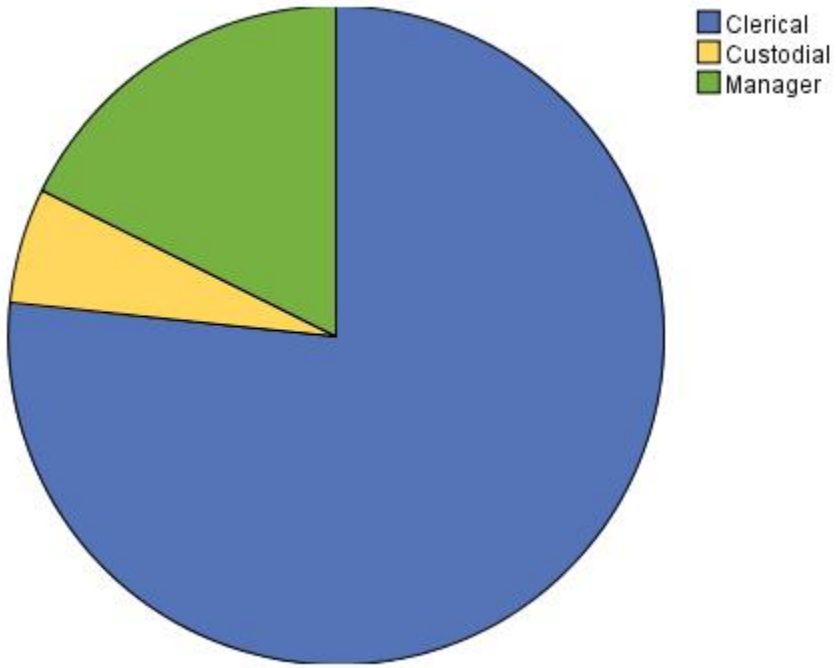


Figure 473. Pie chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
COORD: polar.theta()
SCALE: linear(dim(1), dataMinimum(), dataMaximum())
GUIDE: axis(dim(1), null())
GUIDE: legend(aesthetic(aesthetic.color), null())
ELEMENT: interval.stack(position(summary.count(1)), color(jobcat),
                        label(jobcat))
```

Figure 474. GPL for pie chart with labels

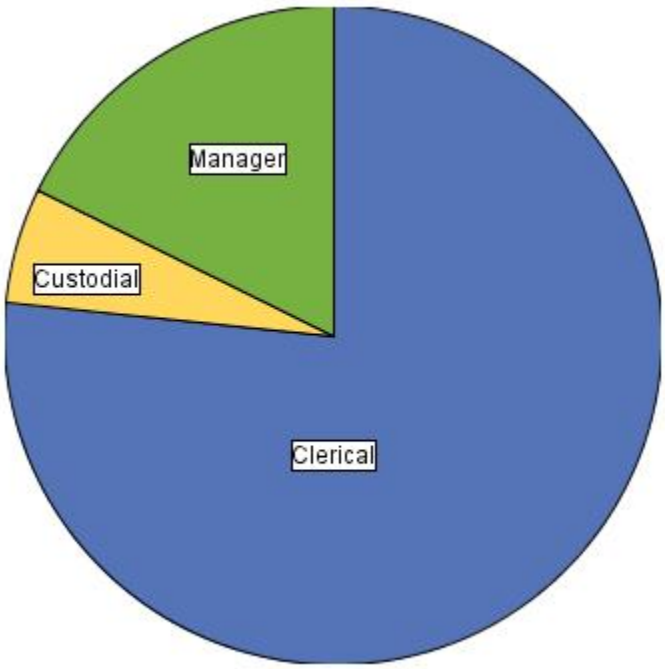


Figure 475. Pie chart with labels

Paneled Pie Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
COORD: polar.theta()
SCALE: linear(dim(1), dataMinimum(), dataMaximum())
GUIDE: axis(dim(1), null())
GUIDE: axis(dim(2), label("Gender"))
ELEMENT: interval.stack(position(summary.percent.count(1*gender))),
                        color(jobcat))
```

Figure 476. GPL for paneled pie chart

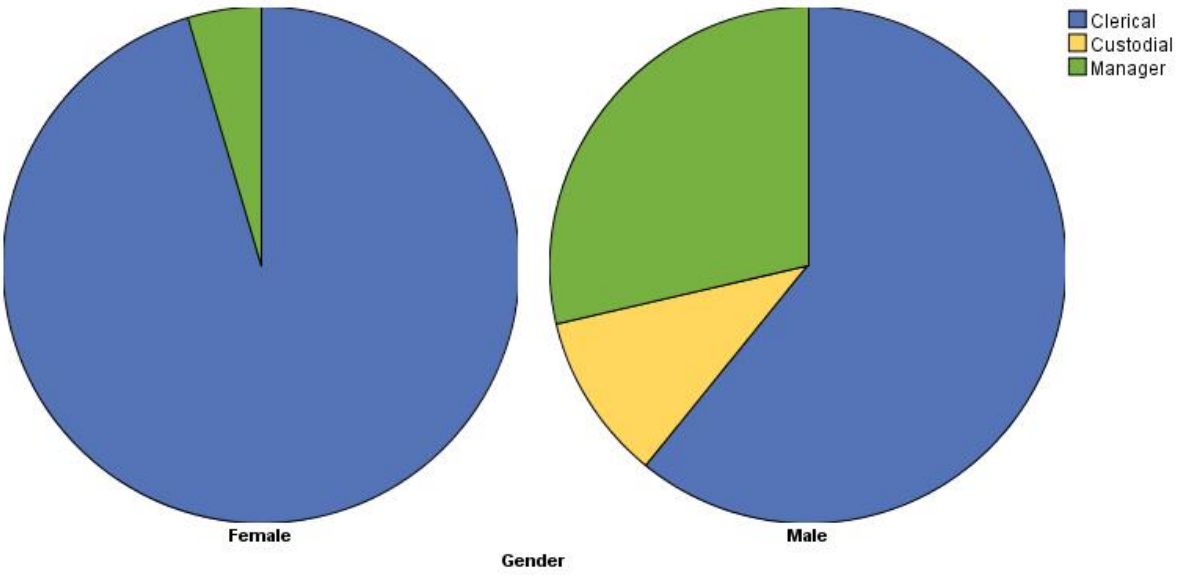


Figure 477. Paneled pie chart

Stacked Pie Chart

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
COORD: polar(transpose())
GUIDE: axis(dim(1), null())
GUIDE: axis(dim(2), null())
GUIDE: legend(aesthetic(aesthetic.texture.pattern.interior), label("Employment Category"))
GUIDE: legend(aesthetic(aesthetic.color.interior), label("Gender"))
ELEMENT: interval.stack(position(summary.percent.count(jobcat*1, base.coordinate(dim(1)))),
                        texture.pattern(jobcat), color(gender), size(size."100%"))
```

Figure 478. GPL for stacked pie chart

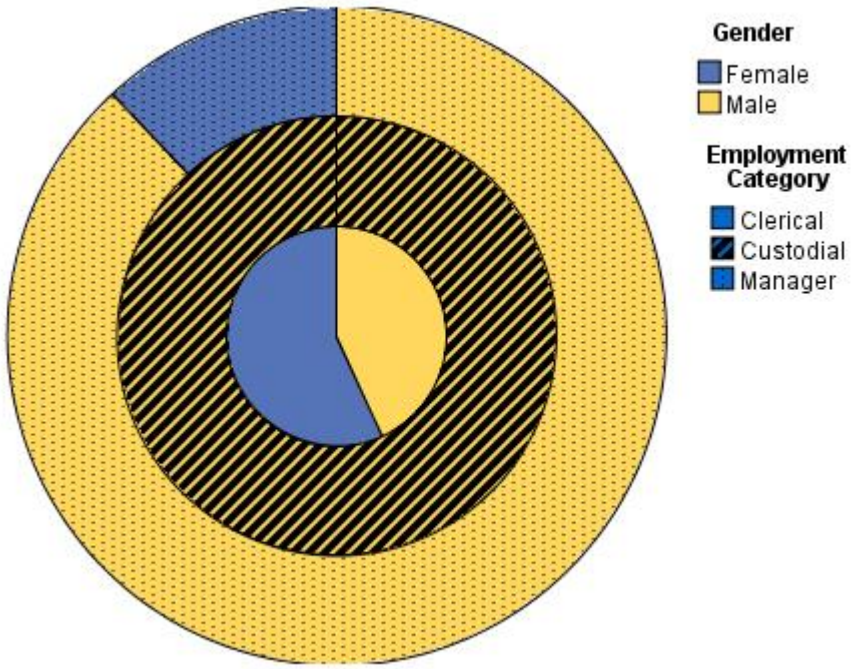


Figure 479. Stacked pie chart

Boxplot Examples

This section provides examples of different types of box plots.

1-D Boxplot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
COORD: rect(dim(1))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: schema(position(bin.quantile.letter(salary)), size(size."50%"))
```

Figure 480. GPL for 1-D box plot

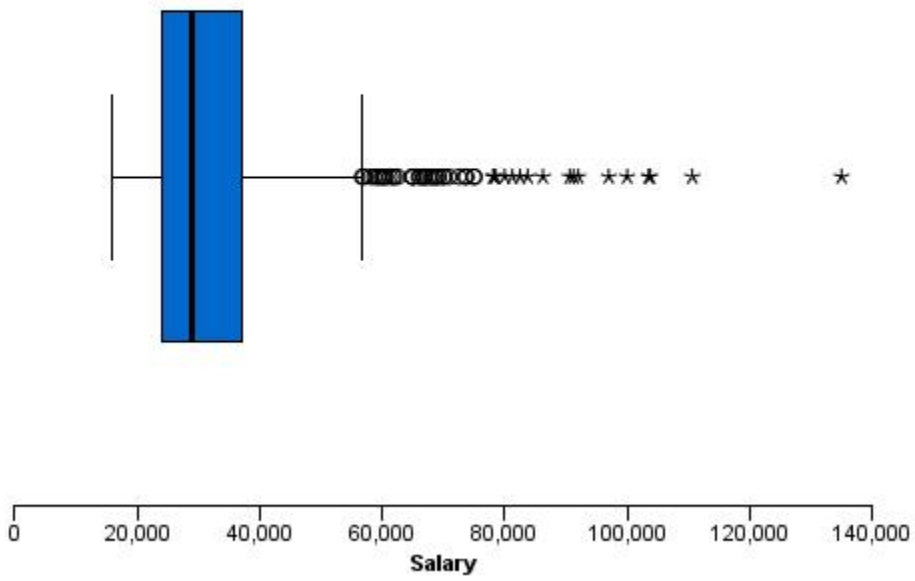


Figure 481. 1-D boxplot

Boxplot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
GUIDE: axis(dim(2), label("Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: schema(position(bin.quantile.letter(jobcat*salary)))
```

Figure 482. GPL for boxplot

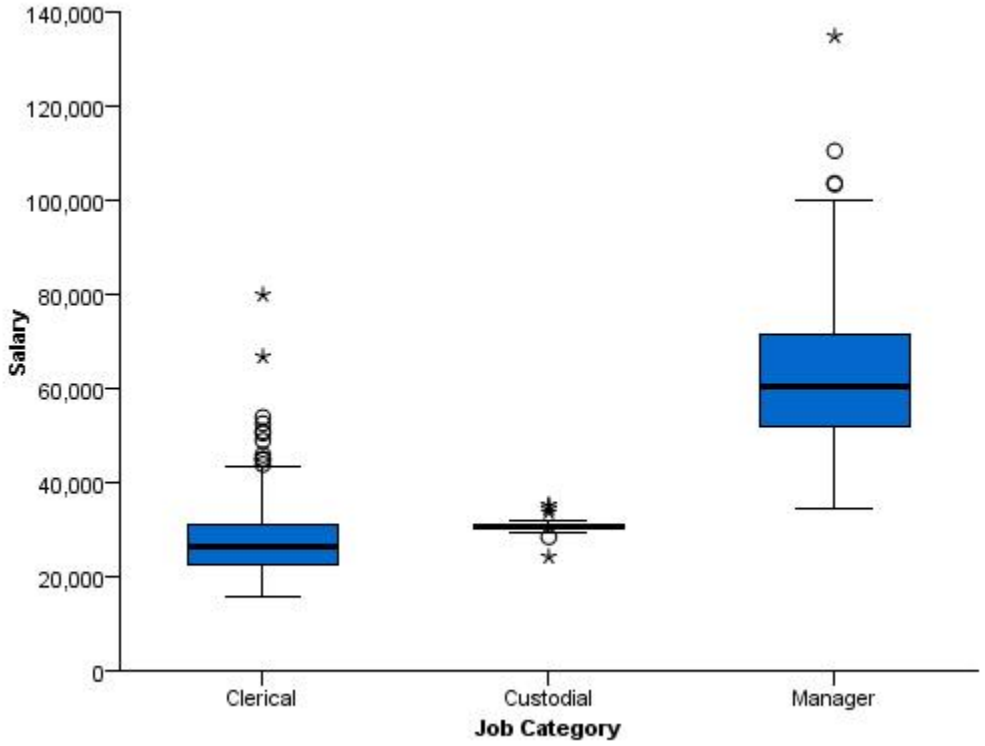


Figure 483. Boxplot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: salary=col(source(s), name("salary"))
DATA: id = col(source(s), name("id"), unit.category())
GUIDE: axis(dim(2), label("Salary"))
GUIDE: axis(dim(1), label("Job Category"))
ELEMENT: schema(position(bin.quantile.letter(jobcat*salary)), label(id))
```

Figure 484. GPL for boxplot with labeled outliers

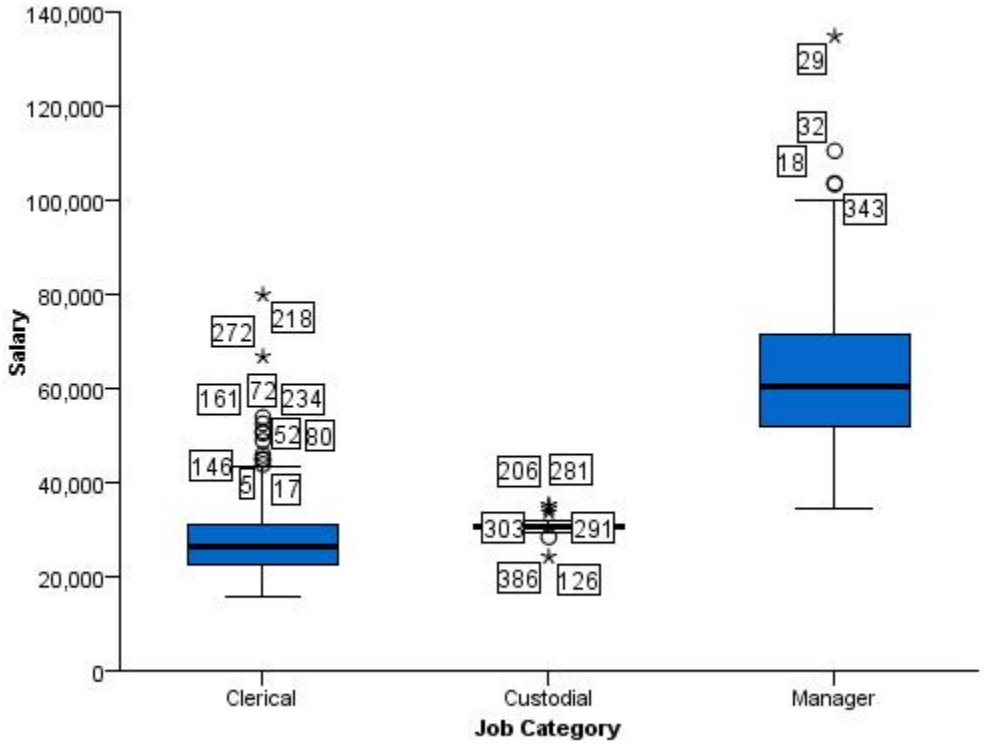


Figure 485. Boxplot with labeled outliers

Clustered Boxplot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat=col(source(s), name("jobcat"), unit.category())
DATA: gender=col(source(s), name("gender"), unit.category())
DATA: salary=col(source(s), name("salary"))
COORD: rect(dim(1,2), cluster(3))
GUIDE: axis(dim(2), label("Salary"))
GUIDE: axis(dim(3), label("Job Category"))
ELEMENT: schema(position(bin.quantile.letter(gender*salary*jobcat)), color(gender))
```

Figure 486. GPL for clustered boxplot

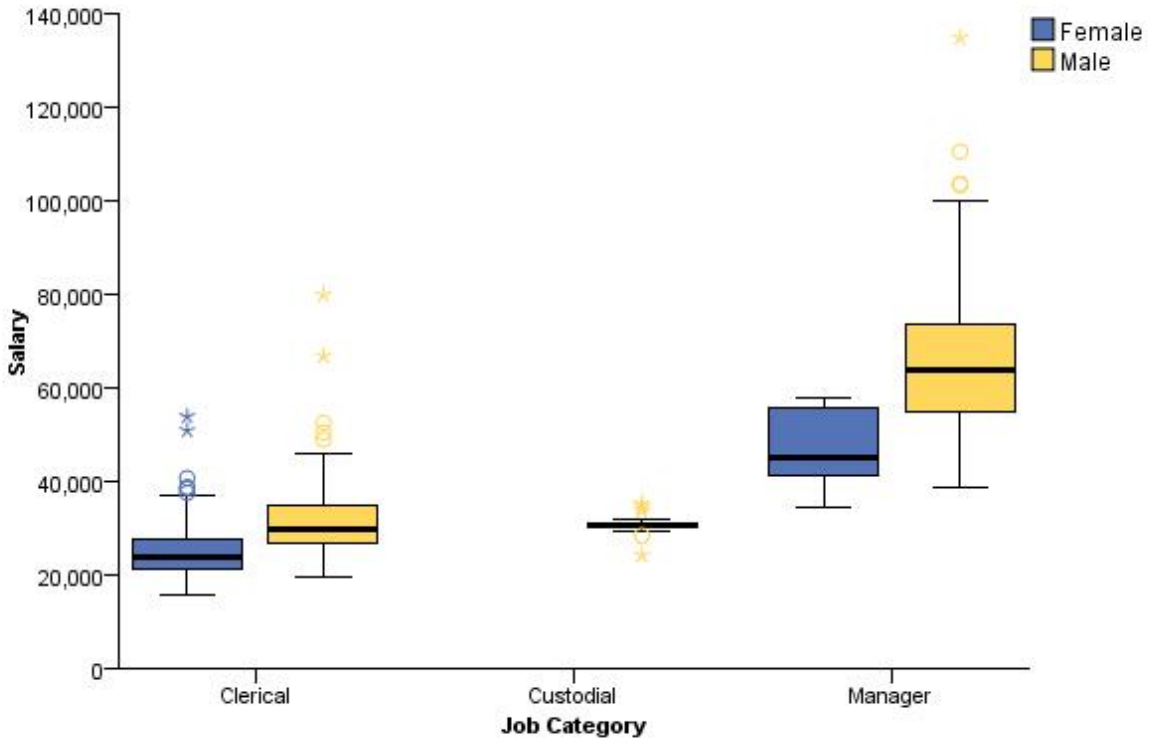


Figure 487. Clustered boxplot

Boxplot With Overlaid Dot Plot

```
SOURCE: s = userSource(id("customer_subset"))
DATA: region = col(source(s), name("region"), unit.category())
DATA: income = col(source(s), name("income"))
GUIDE: axis(dim(2), label("Income"))
GUIDE: axis(dim(1), label("Region"))
SCALE: linear(dim(2), include(0))
ELEMENT: schema(position(bin.quantile.letter(region*income)))
ELEMENT: point.dodge.symmetric(position(bin.dot(region*income, dim(2))),
                               color(color.red))
```

Figure 488. GPL for boxplot with overlaid dot plot

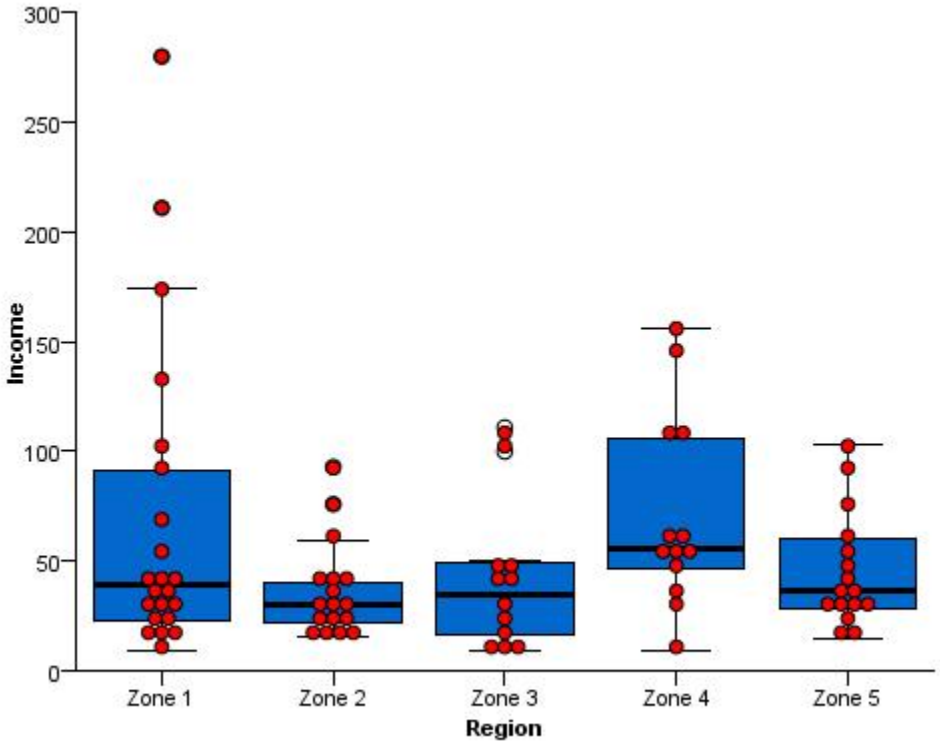


Figure 489. Boxplot with overlaid dot plot

Multi-Graph Examples

This section provides examples of multiple graphs in the same page display.

Scatterplot with Border Histograms

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
DATA: salbegin = col(source(s), name("salbegin"))
GRAPH: begin(origin(5%, 10%), scale(85%, 85%))
GUIDE: axis(dim(1), label("Beginning Salary"))
GUIDE: axis(dim(2), label("Current Salary"))
ELEMENT: point(position(salbegin*salary))
GRAPH: end()
GRAPH: begin(origin(5%, 0%), scale(85%, 10%))
GUIDE: axis(dim(1), ticks(null()))
GUIDE: axis(dim(2), null())
ELEMENT: interval(position(summary.count(bin.rect(salbegin))))
GRAPH: end()
GRAPH: begin(origin(90%, 10%), scale(10%, 85%))
COORD: rect(dim(1, 2), transpose())
GUIDE: axis(dim(1), ticks(null()))
GUIDE: axis(dim(2), null())
ELEMENT: interval(position(summary.count(bin.rect(salary))))
GRAPH: end()
```

Figure 490. GPL for scatterplot with border histograms

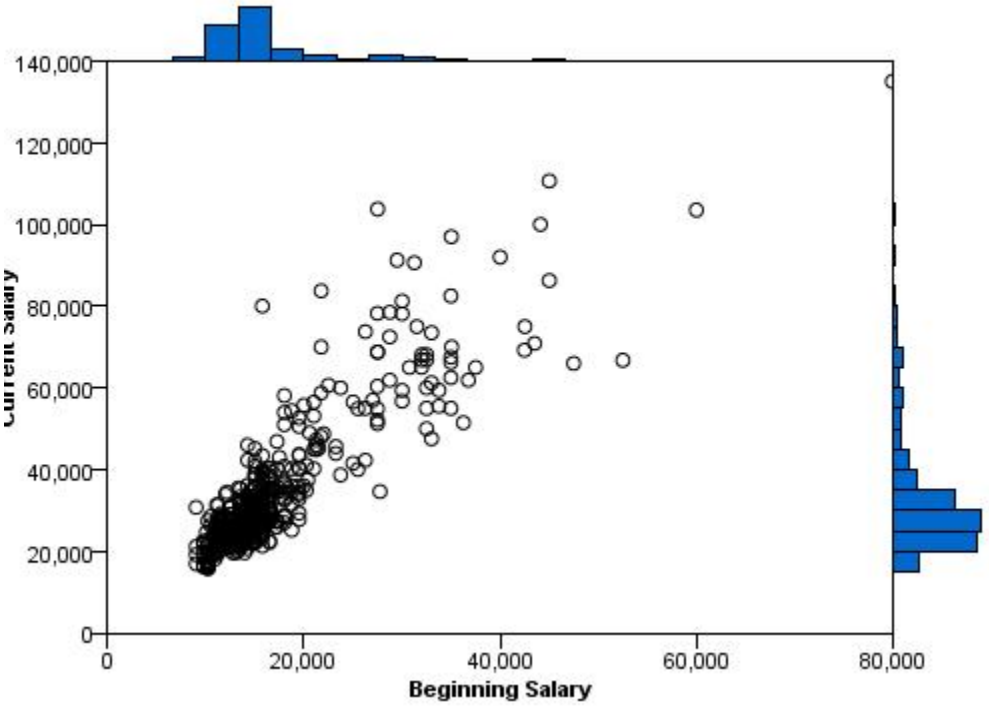


Figure 491. Scatterplot with border histograms

Scatterplot with Border Boxplots

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
DATA: salbegin = col(source(s), name("salbegin"))
GRAPH: begin(origin(5%, 10%), scale(85%, 85%))
GUIDE: axis(dim(1), label("Beginning Salary"))
GUIDE: axis(dim(2), label("Current Salary"))
ELEMENT: point(position(salbegin*salary))
GRAPH: end()
GRAPH: begin(origin(5%, 0%), scale(85%, 10%))
COORD: rect(dim(1))
GUIDE: axis(dim(1), ticks(null()))
ELEMENT: schema(position(bin.quantile.letter(salbegin)), size(size."80%"))
GRAPH: end()
GRAPH: begin(origin(90%, 10%), scale(10%, 85%))
COORD: transpose(rect(dim(1)))
GUIDE: axis(dim(1), ticks(null()))
ELEMENT: schema(position(bin.quantile.letter(salary)), size(size."80%"))
GRAPH: end()
```

Figure 492. GPL for scatterplot with border boxplots

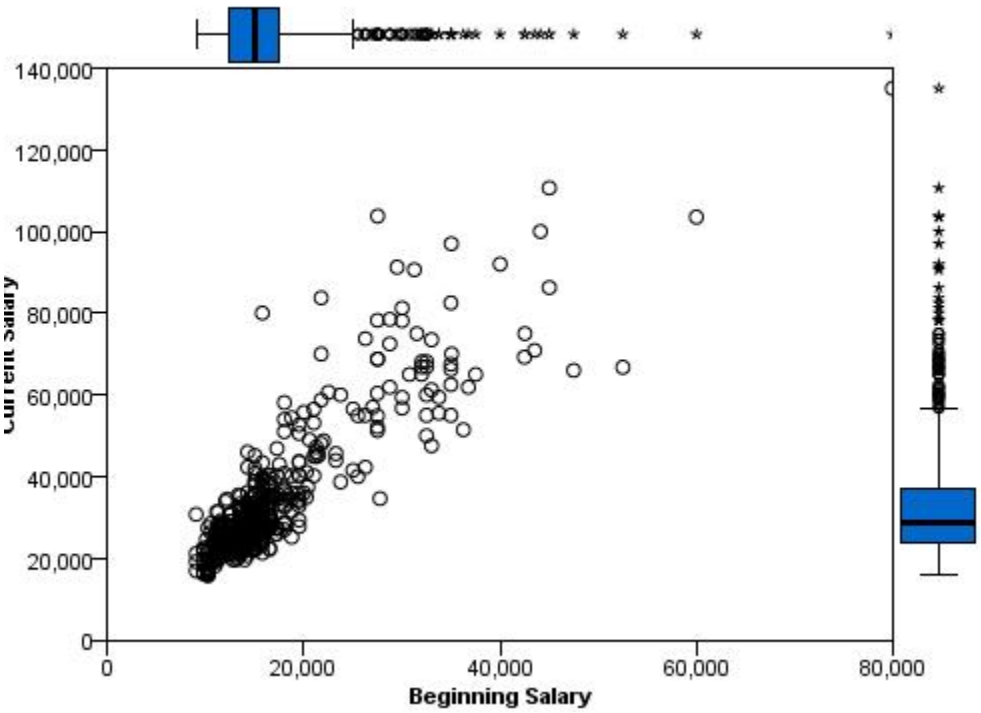


Figure 493. Scatterplot with border boxplots

Stocks Line Chart with Volume Bar Chart

```
SOURCE: s = userSource(id("stocks"))
DATA: date = col(source(s), name("Date"), unit.time(), format("MM/dd/yy"))
DATA: close = col(source(s), name("Close"))
DATA: volume = col(source(s), name("Volume"))
GRAPH: begin(origin(10%, 0%), scale(90%, 60%))
GUIDE: axis(dim(1), ticks(null()))
GUIDE: axis(dim(2), label("Close"))
ELEMENT: line(position(date*close))
GRAPH: end()
GRAPH: begin(origin(10%, 70%), scale(90%, 25%))
GUIDE: axis(dim(1), label("Date"))
GUIDE: axis(dim(2), label("Volume"))
ELEMENT: interval(position(date*volume))
GRAPH: end()
```

Figure 494. GPL for stocks and volume chart

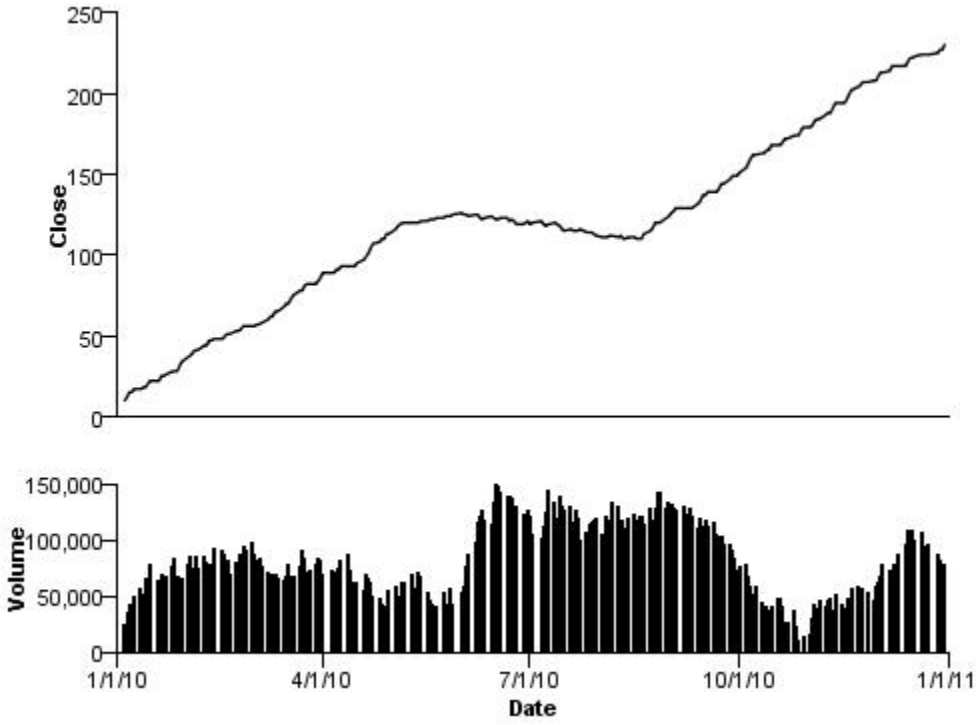


Figure 495. Stocks and volume chart

Dual Axis Graph

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: salary = col(source(s), name("salary"))
SCALE: y1 = linear(dim(2), include(0.0))
SCALE: y2 = linear(dim(2), include(0.0))
GUIDE: axis(dim(1), label("Employment Category"))
GUIDE: axis(scale(y1), label("Mean Salary"))
GUIDE: axis(scale(y2), label("Count"), opposite(), color(color.red))
ELEMENT: interval(position(summary.mean(jobcat*salary)), scale(y1))
ELEMENT: line(position(summary.count(jobcat)), color(color.red), scale(y2))
```

Figure 496. GPL for dual axis graph

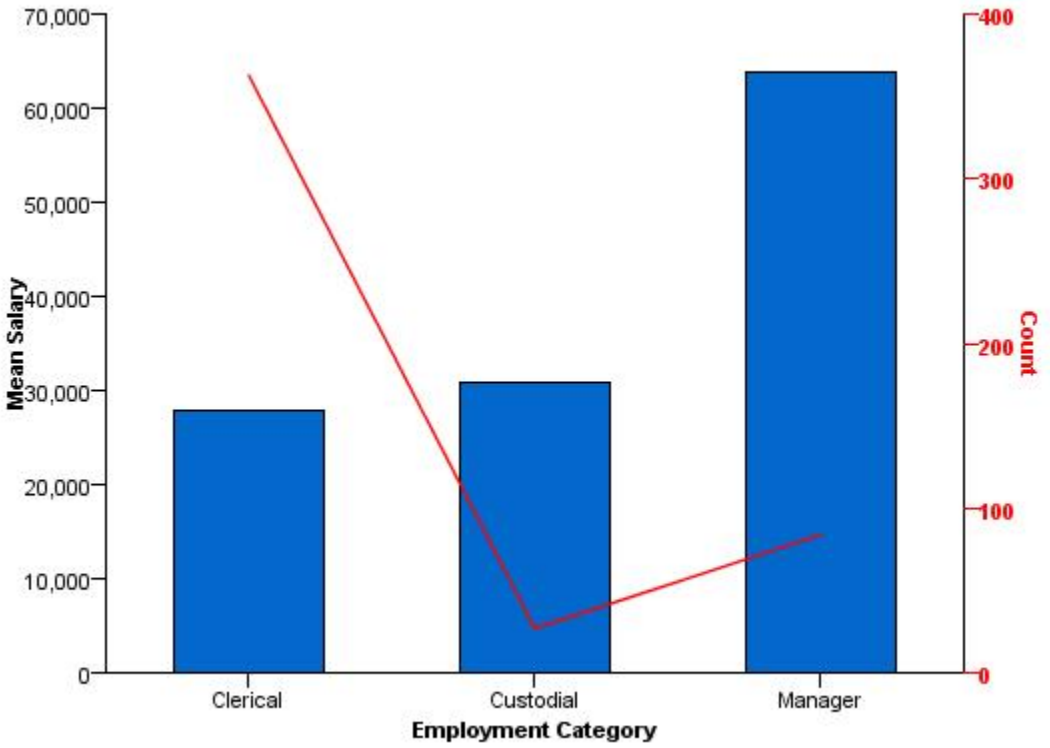


Figure 497. Dual axis graph

Histogram with Dot Plot

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
GRAPH: begin(origin(5.0%, 5.0%), scale(90.0%, 90.0%))
COORD: rect(dim(1, 2))
ELEMENT: interval(position(summary.count(bin.rect(salary))),
                  transparency.interior(transparency."0.9"))
GRAPH: end()
GRAPH: begin(origin(5.0%, 5.0%), scale(90.0%, 90.0%))
COORD: rect(dim(1))
GUIDE: axis(dim(1), ticks(null()))
GUIDE: axis(dim(2), ticks(null()))
ELEMENT: point.dodge.asymmetric(position(bin.dot(salary)))
GRAPH: end()
```

Figure 498. GPL for histogram with dot plot

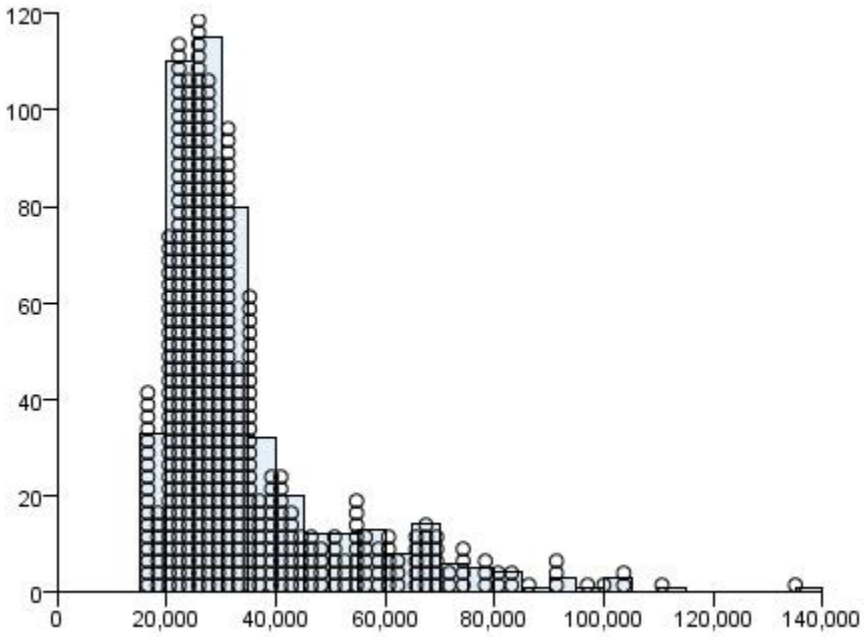


Figure 499. Histogram with dot plot

Other Examples

This section provides examples that demonstrate other features of GPL that are not specific to chart types.

Collapsing Small Categories

If you are creating a graph with several, small categories (that is, categories with a small sum), you may want to collapse those categories into a larger, common category. Following is an example that collapses small categories in a pie chart.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: educ = col(source(s), name("educ"), unit.category())
DATA: salary = col(source(s), name("salary"))
TRANS: educ_collapsed = collapse(category(educ), minimumPercent(5.0),
                                 sumVariable(salary), otherValue("Other"))
COORD: polar.theta()
SCALE: linear(dim(1), dataMinimum(), dataMaximum())
GUIDE: axis(dim(1), null())
GUIDE: legend(aesthetic(aesthetic.color.interior),
                        label("Educational Level (years)"))
ELEMENT: interval.stack(position(summary.sum(salary)),
                        color.interior(educ_collapsed))
```

Figure 500. GPL for collapsing small categories

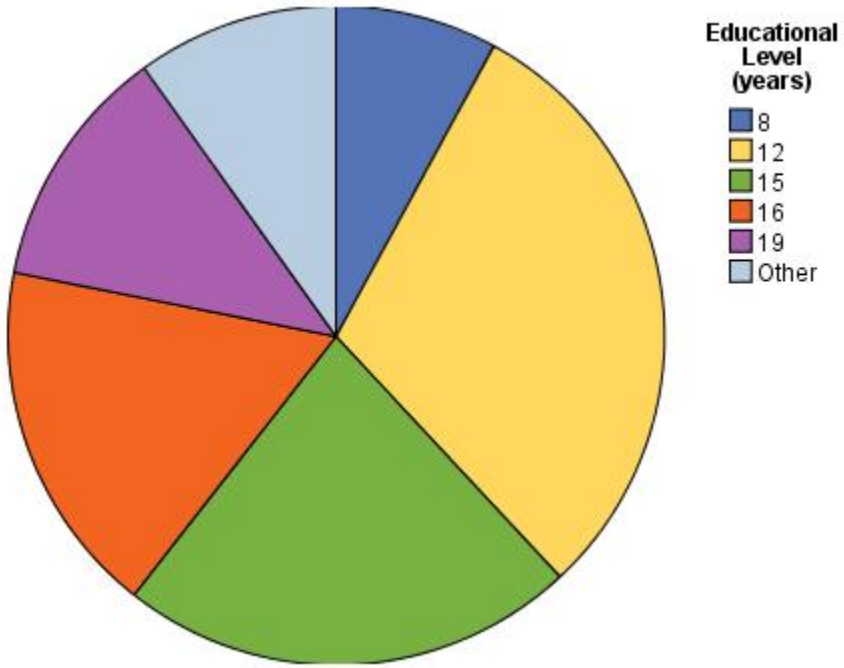


Figure 501. Graph with collapsed categories

Mapping Aesthetics

This example demonstrates how you can map a specific categorical value in the graph to a specific aesthetic value. In this case, "Female" bars are colored green in the resulting graph.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salary = col(source(s), name("salary"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: gender = col(source(s), name("gender"), unit.category())
COORD: rect(dim(1, 2), cluster(3))
SCALE: cat(aesthetic(aesthetic.color), map(("Female", color.green)))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(3), label("Job Category"))
ELEMENT: interval(position(summary.mean(gender*salary*jobcat)),
                  color(gender))
SCALE: cat(aesthetic(aesthetic.color), map(("f", color.green)))
GUIDE: axis(dim(2), label("Mean Salary"))
GUIDE: axis(dim(3), label("Job Category"))
ELEMENT: interval(position(summary.mean(gender*salary*jobcat)),
                  color(gender))
```

Figure 502. GPL for mapping aesthetics

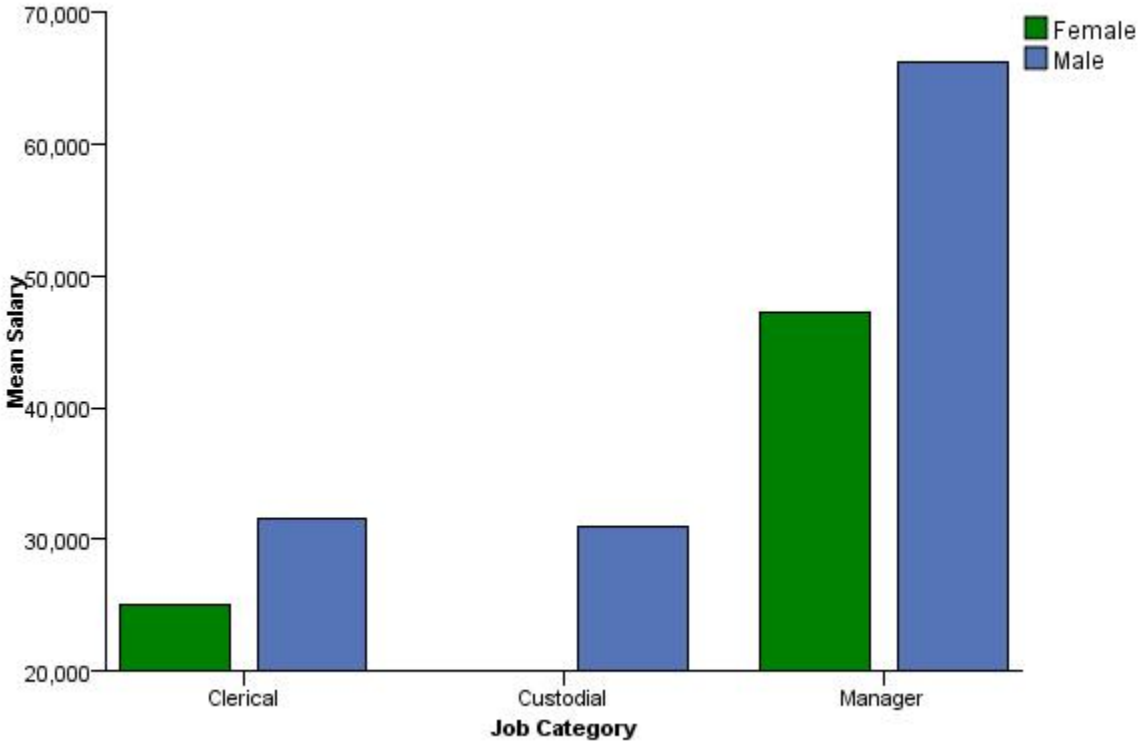


Figure 503. Graph with mapped aesthetics

Faceting by Separate Variables

This example demonstrates how you can create facets based on separate variables, so that each facet shows the different variable information. Note that you can do something similar with nesting. See the topic “Line Chart with Separate Scales ” on page 360 for more information.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: salary = col(source(s), name("salary"))
DATA: salbegin = col(source(s), name("salbegin"))
ELEMENT: schema(position(bin.quantile.letter(jobcat*salbegin*"Beginning Salary"+
                                             jobcat*salary*"Current Salary")))
```

Figure 504. GPL for faceting by separate variables

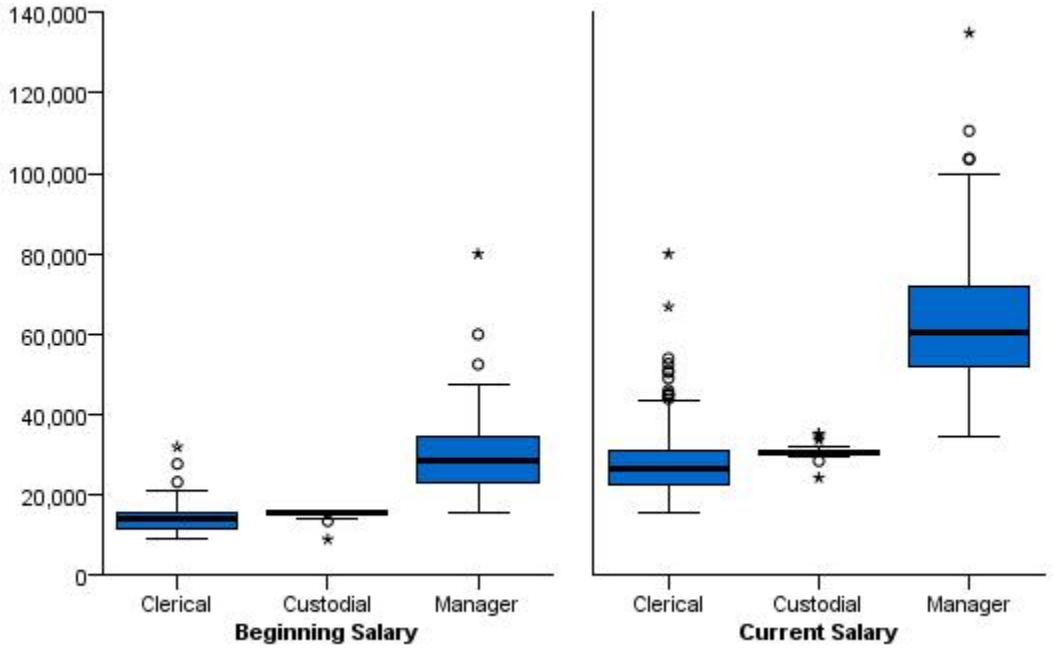


Figure 505. Faceting by separate variables

Grouping by Separate Variables

This example demonstrates how you can create groups based on separate variables, so that each instance of the graphic element shows the different variable information. In this example, you are comparing the results of two variables across a categorical variable.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin=col(source(s), name("salbegin"))
DATA: salary=col(source(s), name("salary"))
DATA: educ=col(source(s), name("educ"), unit.category())
GUIDE: axis(dim(1), label("Educational Level (years)"))
ELEMENT: line(position(summary.mean(educ*salary)),color("Current Salary"))
ELEMENT: line(position(summary.mean(educ*salbegin)),color("Beginning Salary"))
```

Figure 506. GPL for grouping by separate variables

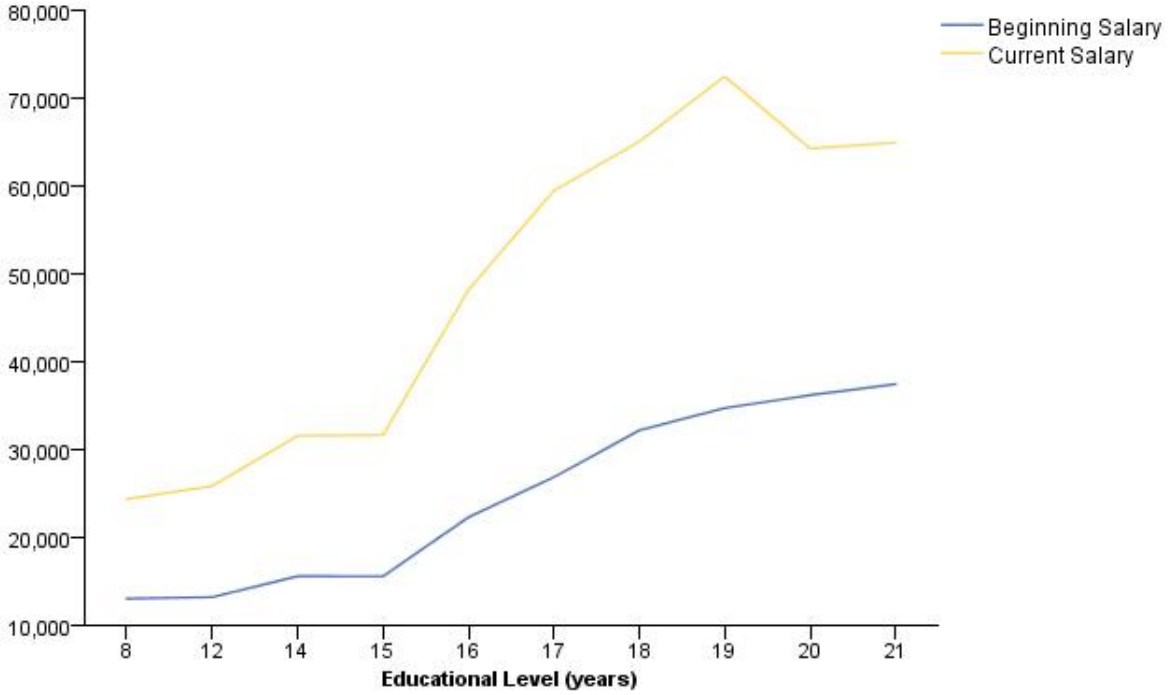


Figure 507. Grouping by separate variables

Clustering Separate Variables

This example demonstrates how you can cluster separate variables.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: salary = col(source(s), name("salary"))
DATA: salbegin = col(source(s), name("salbegin"))
COORD: rect(cluster(3))
SCALE: linear(dim(2), include(0.0))
GUIDE: axis(dim(2), label("Mean"))
GUIDE: axis(dim(3), label("Job Category"))
ELEMENT: interval(position(summary.mean("Beginning Salary"*salbegin*jobcat)),
                  color("Beginning Salary"))
ELEMENT: interval(position(summary.mean("Current Salary"*salary*jobcat)),
                  color("Current Salary"))
```

Figure 508. GPL for clustering separate variables

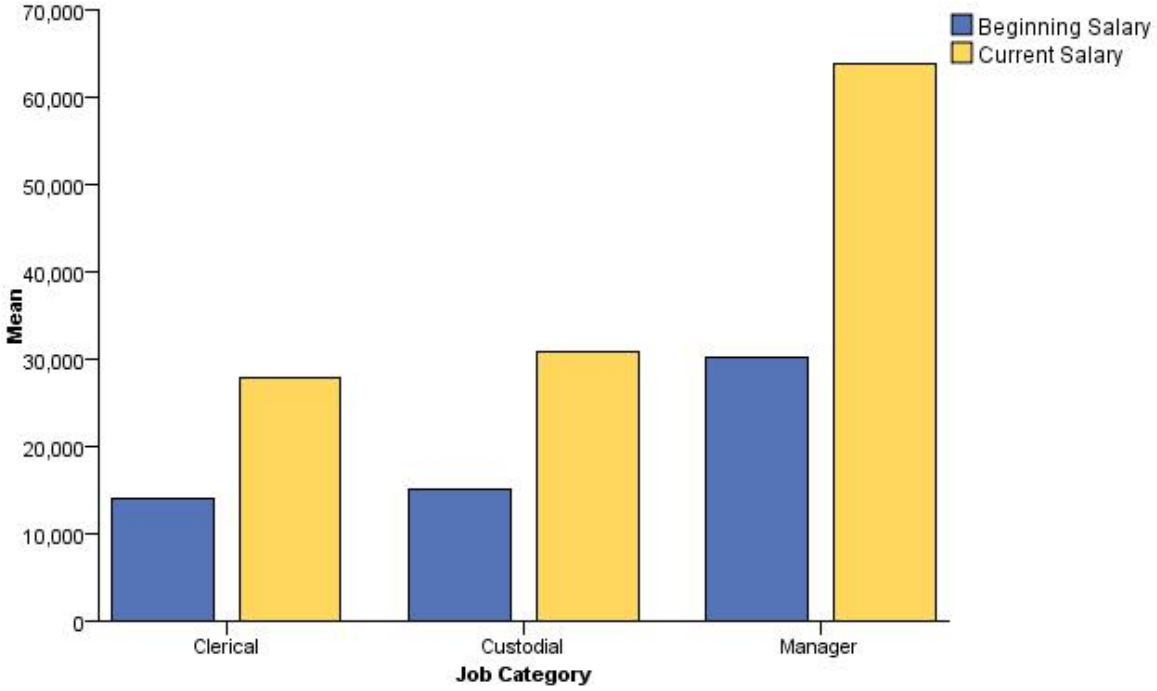


Figure 509. Clustering separate variables

Binning over Categorical Values

This example demonstrates how you use binning to show the distribution of a continuous variable over categorical values.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: jobcat = col(source(s), name("jobcat"), unit.category())
DATA: salary = col(source(s), name("salary"))
GUIDE: axis(dim(1), label("Employment Category"))
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: legend(aesthetic(aesthetic.color), label("Count"))
ELEMENT: polygon(position(bin.rect(jobcat*salary, dim(2))), color(summary.count()))
```

Figure 510. GPL for binning over categorical values

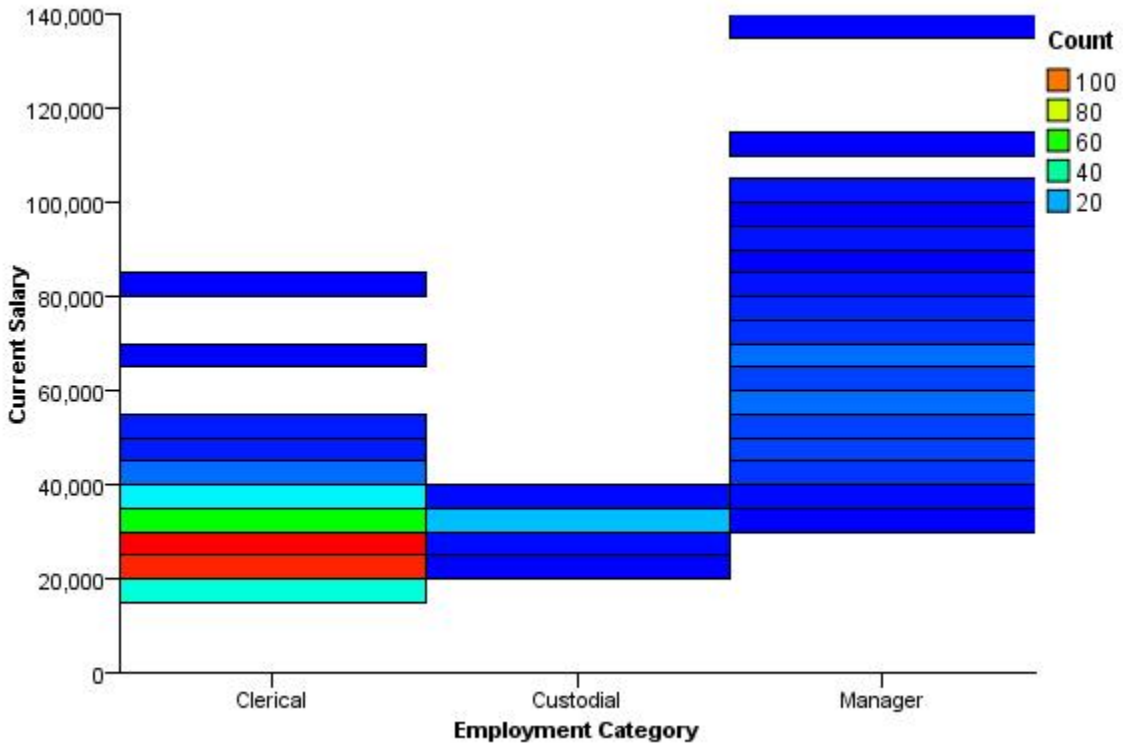


Figure 511. Binning over categorical values

Categorical Heat Map

```
SOURCE: s = userSource(id("Employeedata"))
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: educ = col(source(s), name("educ"), unit.category())
DATA: salary = col(source(s), name("salary"))
GUIDE: axis(dim(1), label("Educational Level"))
GUIDE: axis(dim(2), label("Gender"))
GUIDE: legend(aesthetic(aesthetic.color), label("Mean Salary"))
ELEMENT: point(position(educ*gender), shape(shape.square), size(size."10%"),
               color(summary.mean(salary)))
```

Figure 512. GPL for categorical heat map

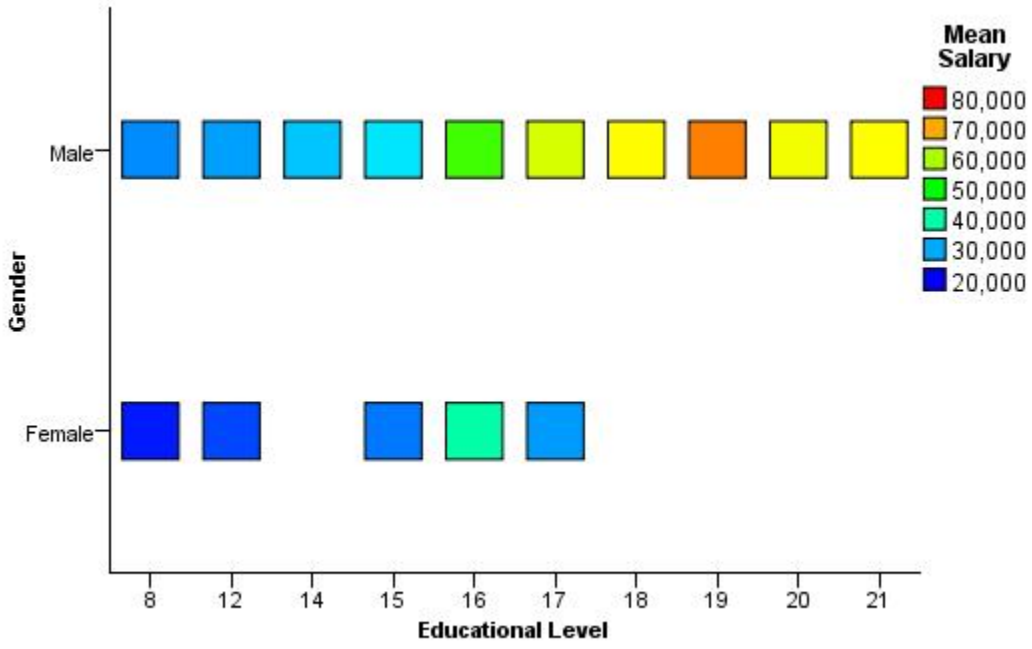


Figure 513. Categorical heat map

```
SOURCE: s = userSource(id("Employeedata"))
DATA: gender = col(source(s), name("gender"), unit.category())
DATA: educ = col(source(s), name("educ"), unit.category())
DATA: salary = col(source(s), name("salary"))
GUIDE: axis(dim(1), label("Educational Level"))
GUIDE: axis(dim(2), label("Gender"))
GUIDE: legend(aesthetic(aesthetic.color), label("Mean Salary"))
ELEMENT: polygon(position(educ*gender), color(summary.mean(salary)))
```

Figure 514. GPL for alternate categorical heat map

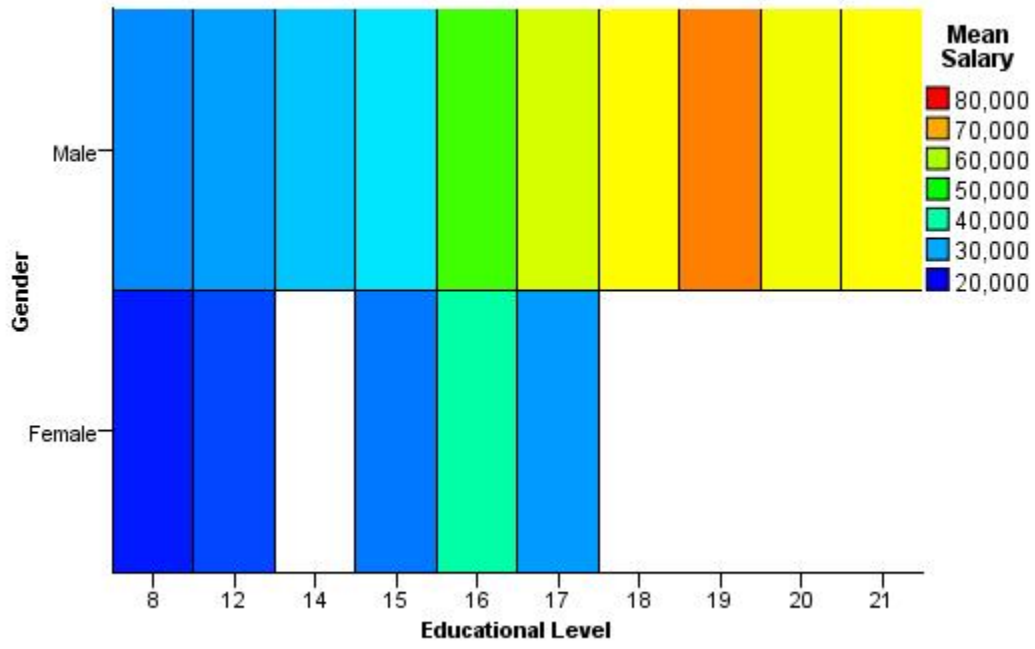


Figure 515. Alternate categorical heat map

Creating Categories Using the eval Function

This example demonstrates how you can use the `eval` function to create categories based on an expression.

```
SOURCE: s = userSource(id("Employeedata"))
DATA: salbegin = col(source(s), name("salbegin"))
DATA: salary = col(source(s), name("salary"))
DATA: educ = col(source(s), name("educ"))
TRANS: college = eval(educ>12 ? "College" : "No College")
GUIDE: axis(dim(2), label("Current Salary"))
GUIDE: axis(dim(1), label("Beginning Salary"))
ELEMENT: point(position(salbegin*salary), color(college))
```

Figure 516. GPL for creating categories with the eval function

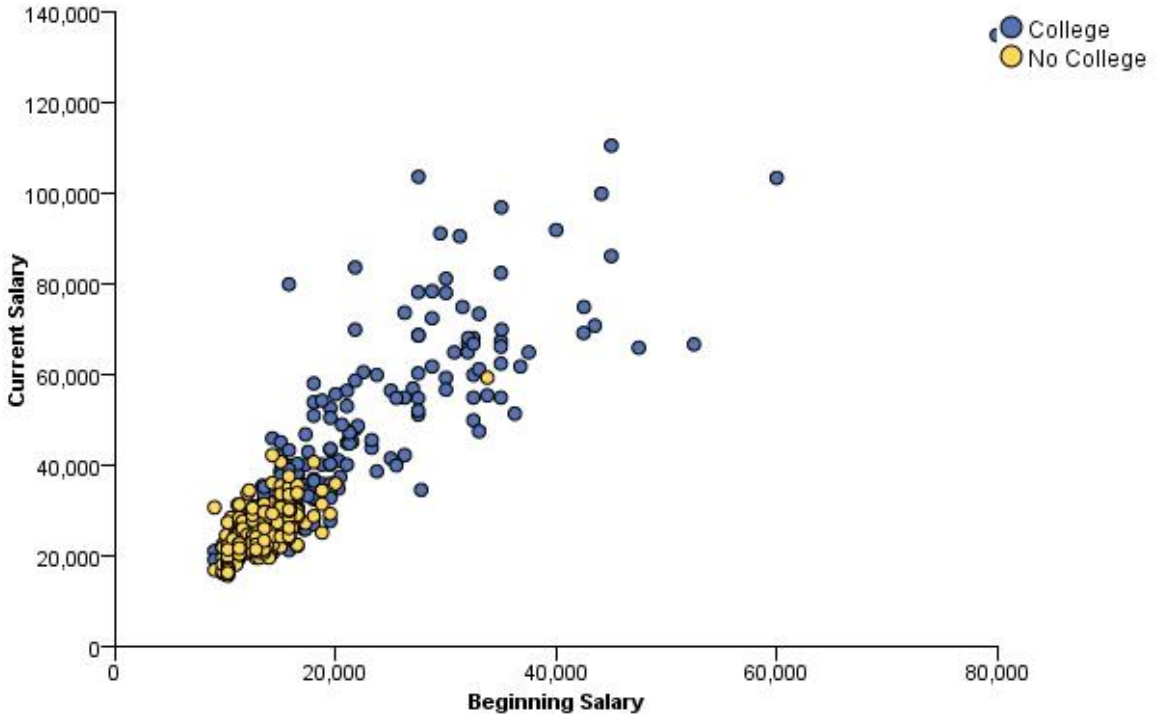


Figure 517. Creating categories with the eval function

Chapter 4. GPL Constants

Color Constants

Color constants: aliceblue, aqua, azure, bisque, black, blanchedalmond, blue, blueviolet, brown, burlywood, cadetblue, chartreuse, chocolate, coral, cornflowerblue, cornsilk, crimson, cyan, darkblue, darkcyan, darkgoldenrod, darkgray, darkgreen, darkgrey, darkkhaki, darkmagenta, darkolivegreen, darkorange, darkorchid, darkred, darksalmon, darkseagreen, darkslateblue, darkslategray, darkslategrey, darkturquoise, darkviolet, deeppink, deepskyblue, dimgray, dimgrey, dodgerblue, firebrick, floralwhite, forestgreen, fuchsia, gainsboro, ghostwhite, gold, goldenrod, gray, grey, green, greenyellow, honeydew, hotpink, indianred, indigo, ivory, khaki, lavender, lavenderblush, lawngreen, lemonchiffon, lightblue, lightcoral, lightcyan, lightgoldenrodyellow, lightgray, lightgreen, lightgrey, lightpink, lightsalmon, lightseagreen, lightskyblue, lightslategray, lightslategrey, lightsteelblue, lightyellow, lime, limegreen, linen, magenta, maroon, mediumaquamarine, mediumblue, mediumorchid, mediumpurple, mediumseagreen, mediumslateblue, mediumspringgreen, mediumturquoise, mediumvioletred, midnightblue, mintcream, mistyrose, moccasin, navajowhite, navy, oldlace, olive, olivedrab, orange, orangered, orchid, palegoldenrod, palegreen, paleturquoise, palevioletred, papayawhip, peachpuff, peru, pink, plum, powderblue, purple, red, rosybrown, royalblue, saddlebrown, salmon, sandybrown, seagreen, seashell, sienna, silver, skyblue, slateblue, slategray, slategrey, snow, springgreen, steelblue, tan, teal, thistle, tomato, turquoise, violet, wheat, white, whitesmoke, yellow, yellowgreen

Shape Constants

The following constants are all the valid constants for the different graphic element types. Note that the constants for the edge graphic element appear in multiple tables.

shape.interior constants for interval elements: ibeam, line, square

shape.interior constants for edge and point elements: arrow, bowtie, circle, cross, decagon, elbow, elbowArrow, female, flower, flower3, flower4, flower5, flower6, flower7, flower8, flower9, flower10, heptagon, hexagon, hollowBowtie, hollowCircle, hollowDecagon, hollowHeptagon, hollowHexagon, hollowNonagon, hollowOctagon, hollowPentagon, hollowPolygon, hollowRoundRectangle, hollowSquare, hollowStar, hollowStar3, hollowStar4, hollowStar5, hollowStar6, hollowStar7, hollowStar8, hollowStar9, hollowStar10, hollowTriangle, ibeam, line, male, nonagon, octagon, pentagon, plus, polygon, roundRectangle, square, star, star3, star4, star5, star6, star7, star8, star9, star10, triangle

The following constants can also be used with the `shape.exterior` function to specify the border dashing of all other graphic elements. (An exception is the edge element, for which `shape.interior` defines the overall shape *or* dashing.) These constants are also used with the `shape` function that specifies the dashing of `form.line` guides.

shape.interior constants for edge, line, and path elements: dash, dash_1_dot, dash_2_dots, dash_3_dots, dash_2x, dash_3x, dash_4_dots, dash_dash2x, half_dash, solid

Size Constants

Size constants: tiny, small, medium, large, huge

Pattern Constants

Pattern constants: checkered, grid, grid2, grid3, grid4, grid5, mesh, mesh2, mesh3, mesh4, mesh5, pluses, pluses2, solid

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBMproducts. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© Copyright IBM Corp. 2021. Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. 1989 - 2021. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Index

A

aestheticMaximum function (GPL) 59
aestheticMinimum function (GPL) 60
aestheticMissing function (GPL) 61
algebra (GPL)
    rules 3
alpha function (GPL) 62
analysis variable (GPL) 3
area element (GPL) 44
asn scale (GPL) 28
atanh scale (GPL) 28
axis guide (GPL) 39

B

bar element (GPL) 45
base function (GPL) 62
base.aesthetic function (GPL) 63
base.all function (GPL) 63
base.coordinate function (GPL) 64
begin function (GPL) 65
beta function (GPL) 66
bin.dot function (GPL) 66
bin.hex function (GPL) 68
bin.quantile.letter function (GPL) 70
bin.rect function (GPL) 72
binCount function (GPL) 75
binStart function (GPL) 75
binWidth function (GPL) 76
blend operator (GPL) 3
boxplot (GPL) 50

C

cat (categorical) scale (GPL) 29
chiSquare function (GPL) 77
cLoglog scale (GPL) 30
closed function (GPL) 77
cluster function (GPL) 77
col function (GPL) 79
collapse function (GPL) 80
collision modifiers (GPL)
    difference 51
    dodge 51
    dodge.asymmetric 51
    dodge.symmetric 52
    jitter 52
    stack 53
color function (GPL) 81, 82
color.brightness function (GPL) 82, 83
color.hue function (GPL) 84
color.saturation function (GPL) 85, 86
COMMENT statement (GPL) 19
constants (GPL)
    color 385
    pattern 385
constants (GPL) *(continued)*
    shape 385
    size 385
    user 3
COORD statement (GPL) 22
coordinate types (GPL)
    parallel 23
    polar 24
    polar.theta 25
    project 182
    rect (rectangular) 26
coordinates (GPL) 5
cross operator (GPL) 3
csvSource function (GPL) 86

D

DATA statement (GPL) 21
dataMaximum function (GPL) 87
dataMinimum function (GPL) 87
delta function (GPL) 88
density.beta function (GPL) 88
density.chiSquare function (GPL) 90
density.exponential function (GPL) 92
density.f function (GPL) 94
density.gamma function (GPL) 96
density.kernel function (GPL) 98
density.logistic function (GPL) 101
density.normal function (GPL) 103
density.poisson function (GPL) 105
density.studentizedRange function (GPL) 107
density.t function (GPL) 109
density.uniform function (GPL) 111
density.weibull function (GPL) 113
difference collision modifier (GPL) 51
dim function (GPL) 115
dodge collision modifier (GPL) 51
dodge.asymmetric collision modifier (GPL) 51
dodge.symmetric collision modifier (GPL) 52

E

edge element (GPL) 45
ELEMENT statement (GPL) 43
element types (GPL)
    area 44
    bar 45
    edge 45
    interval 46
    line 47
    path 48
    point 48
    polygon 49
    schema 50
end function (GPL) 117
eval function (GPL) 118
exclude function (GPL) 122

exponent function (GPL) 123
exponential function (GPL) 123

F

f function (GPL) 123
footnote (GPL) 41, 42
form.line guide (GPL) 40
format function (GPL) 124
format.date function (GPL) 124
format.dateTime function (GPL) 124
format.time function (GPL) 125
from function (GPL) 125
functions (GPL)
    aestheticMaximum 59
    aestheticMinimum 60
    aestheticMissing 61
    alpha 62
    base 62
    base.aesthetic 63
    base.all 63
    base.coordinate 64
    begin 65
    beta 66
    bin.dot 66
    bin.hex 68
    bin.quantile.letter 70
    bin.rect 72
    binCount 75
    binStart 75
    binWidth 76
    chiSquare 77
    closed 77
    cluster 77
    col 79
    collapse 80
    color 81, 82
    color.brightness 82, 83
    color.hue 84
    color.saturation 85, 86
    csvSource 86
    dataMaximum 87
    dataMinimum 87
    delta 88
    density.beta 88
    density.chiSquare 90
    density.exponential 92
    density.f 94
    density.gamma 96
    density.kernel 98
    density.logistic 101
    density.normal 103
    density.poisson 105
    density.studentizedRange 107
    density.t 109
    density.uniform 111
    density.weibull 113
    dim 115
    end 117
    eval 118
    exclude 122
    exponent 123
    exponential 123
    f 123
functions (GPL) *(continued)*
    format 124
    format.date 124
    format.dateTime 124
    format.time 125
    from 125
    gamma 126
    gap 126
    gridlines 126
    in 127
    include 127
    index 128
    iter 128
    jump 128
    label 129, 130
    layout.circle 131
    layout.dag 133
    layout.data 135
    layout.grid 137
    layout.network 139
    layout.random 141
    layout.tree 143
    link.alpha 145
    link.complete 147
    link.delaunay 149
    link.distance 151
    link.gabriel 153
    link.hull 155
    link.influence 157
    link.join 159
    link.mst 162
    link.neighbor 164
    link.relativeNeighborhood 166
    link.sequence 168
    link.tsp 170
    logistic 172
    map 172
    marron 173
    max 173
    min 174
    mirror 174
    missing.gap 175
    missing.interpolate 175
    missing.listwise 175
    missing.pairwise 176
    missing.wings 176
    multiple 177
    noConstant 177
    node 177
    normal 178
    notIn 178
    opposite 178
    origin 179
    poisson 180
    position 180, 181
    preserveStraightLines 182
    proportion 183
    reflect 183
    region.confi.count 184
    region.confi.mean 186
    region.confi.percent.count 188
    region.confi.proportion.count 190
    region.confi.smooth 192
    region.spread.range 194

functions (GPL) *(continued)*
region.spread.sd 196
region.spread.se 198
reverse 201
root 201
savSource 202
scale 202–204
scaledToData 204
segments 205
shape 205, 206
showAll 207
size 207, 208
smooth.cubic 208
smooth.linear 211
smooth.loess 213
smooth.mean 215
smooth.median 217
smooth.quadratic 220
smooth.spline 222
smooth.step 224
sort.data 225
sort.natural 226
sort.statistic 226
sort.values 226
split 227
sqlSource 227
start 228
startAngle 228
studentizedRange 229
summary.count 229
summary.count.cumulative 231
summary.countTrue 234
summary.first 236
summary.kurtosis 238
summary.last 240
summary.max 242
summary.mean 244
summary.median 246
summary.min 248
summary.mode 250
summary.percent 252, 258
summary.percent.count 254, 256
summary.percent.sum 260, 262
summary.percentile 264
summary.percentTrue 267
summary.proportion 269
summary.proportion.count 270, 272
summary.proportion.cumulative 275
summary.proportion.sum 275, 277
summary.proportionTrue 279
summary.range 282
summary.sd 284
summary.se 286
summary.se.kurtosis 288
summary.se.skewness 290
summary.sum 292
summary.sum.cumulative 294
summary.variance 296
t 298
texture.pattern 298
tick 299
functions (GPL) *(continued)*
to 300
transparency 300, 301
transpose 301
uniform 302
unit.percent 302
userSource 303
values 303
visible 304
weibull 304
weight 305
wrap 305

G

gamma function (GPL) 126
gap function (GPL) 126
GPL
algebra 3
algebra and coordinate interaction 5
algebra rules 3
analysis variable 3
blend operator 3
clustering 15
color constants 385
constants, color 385
constants, pattern 385
constants, shape 385
constants, size 385
cross operator 3
data sources for examples 307
example data sources 307
faceting 14
introduction 1
nest operator 3
operator precedence 3
operators 3
paneling 14
pattern constants 385
shape constants 385
size constants 385
stacking 12
syntax rules 2
unity variable 3
user constants 3
using aesthetics 16
GRAPH statement (GPL) 20
graphic element types (GPL)
area 44
bar 45
edge 45
interval 46
line 47
path 48
point 48
polygon 49
schema 50
graphic elements (GPL) 43
gridlines function (GPL) 126
GUIDE statement (GPL) 38
guide types (GPL)
axis 39
form.line 40
legend 40

guide types (GPL) *(continued)*
line 40
text.footnote 41
text.subfootnote 41
text.subsubfootnote 42
text.subsubtitle 42
text.subtitle 42
text.title 43

I

in function (GPL) 127
include function (GPL) 127
index function (GPL) 128
interval element (GPL) 46
iter function (GPL) 128

J

jitter collision modifier (GPL) 52
jump function (GPL) 128

L

label function (GPL) 129, 130
layout.circle function (GPL) 131
layout.dag function (GPL) 133
layout.data function (GPL) 135
layout.grid function (GPL) 137
layout.network function (GPL) 139
layout.random function (GPL) 141
layout.tree function (GPL) 143
legend guide (GPL) 40
line element (GPL) 47
line guide (GPL) 40
linear scale (GPL) 31
link.alpha function (GPL) 145
link.complete function (GPL) 147
link.delaunay function (GPL) 149
link.distance function (GPL) 151
link.gabriel function (GPL) 153
link.hull function (GPL) 155
link.influence function (GPL) 157
link.join function (GPL) 159
link.mst function (GPL) 162
link.neighbor function (GPL) 164
link.relativeNeighborhood function (GPL) 166
link.sequence function (GPL) 168
link.tsp function (GPL) 170
log scale (GPL) 32
logistic function (GPL) 172
logit scale (GPL) 33

M

map function (GPL) 172
marron function (GPL) 173
max function (GPL) 173
min function (GPL) 174
mirror function (GPL) 174
missing.gap function (GPL) 175
missing.interpolate function (GPL) 175
missing.listwise function (GPL) 175
missing.pairwise function (GPL) 176
missing.wings function (GPL) 176
multiple function (GPL) 177

N

nest operator (GPL) 3
noConstant function (GPL) 177
node function (GPL) 177
normal function (GPL) 178
notIn function (GPL) 178

O

operator precedence (GPL) 3
operators (GPL) 3
opposite function (GPL) 178
origin function (GPL) 179

P

PAGE statement (GPL) 20
parallel coordinates (GPL) 23
path element (GPL) 48
point element (GPL) 48
poisson function (GPL) 180
polar coordinates (GPL) 24
polar.theta coordinates (GPL) 25
polygon element (GPL) 49
position function (GPL) 180, 181
pow scale (GPL) 33
preserveStraightLines function (GPL) 182
prob scale (GPL) 34
probit scale (GPL) 35
project coordinates (GPL) 182
proportion function (GPL) 183

R

rect (rectangular) coordinates (GPL) 26
reference line (GPL) 40
reflect function (GPL) 183
region.confi.count function (GPL) 184
region.confi.mean function (GPL) 186
region.confi.percent.count function (GPL) 188
region.confi.proportion.count function (GPL) 190
region.confi.smooth function (GPL) 192
region.spread.range function (GPL) 194
region.spread.sd function (GPL) 196
region.spread.se function (GPL) 198
reverse function (GPL) 201
root function (GPL) 201

S

safeLog scale (GPL) 35
safePower scale (GPL) 36
savSource function (GPL) 202
scale function (GPL) 202–204
SCALE statement (GPL) 27
scale types (GPL)
asn 28
atanh 28

scale types (GPL) *(continued)*
cat (categorical) 29
cLoglog 30
linear 31
log 32
logit 33
pow 33
prob 34
probit 35
safeLog 35
safePower 36
time 37
scaledToData function (GPL) 204
schema element (GPL) 50
segments function (GPL) 205
shape function (GPL) 205, 206
showAll function (GPL) 207
size function (GPL) 207, 208
smooth.cubic function (GPL) 208
smooth.linear function (GPL) 211
smooth.loess function (GPL) 213
smooth.mean function (GPL) 215
smooth.median function (GPL) 217
smooth.quadratic function (GPL) 220
smooth.spline function (GPL) 222
smooth.step function (GPL) 224
sort.data function (GPL) 225
sort.natural function (GPL) 226
sort.statistic function (GPL) 226
sort.values function (GPL) 226
SOURCE statement (GPL) 21
split function (GPL) 227
sqlSource function (GPL) 227
stack collision modifier (GPL) 53
start function (GPL) 228
startAngle function (GPL) 228
statements (GPL)
COMMENT 19
COORD 22
DATA 21
ELEMENT 43
GRAPH 20
GUIDE 38
PAGE 20
SCALE 27
SOURCE 21
TRANS 22
studentizedRange function (GPL) 229
subfootnote guide (GPL) 41
subsubfootnote guide (GPL) 42
subsubtitle guide (GPL) 42
subtitle guide (GPL) 42
summary.count function (GPL) 229
summary.count.cumulative function (GPL) 231
summary.countTrue function (GPL) 234
summary.first function (GPL) 236
summary.kurtosis function (GPL) 238
summary.last function (GPL) 240
summary.max function (GPL) 242
summary.mean function (GPL) 244
summary.median function (GPL) 246
summary.min function (GPL) 248
summary.mode function (GPL) 250
summary.percent function (GPL) 252, 258
summary.percent.count function (GPL) 254, 256
summary.percent.sum function (GPL) 260, 262
summary.percentile function (GPL) 264
summary.percentTrue function (GPL) 267
summary.proportion function (GPL) 269
summary.proportion.count function (GPL) 270, 272
summary.proportion.cumulative function (GPL) 275
summary.proportion.sum function (GPL) 275, 277
summary.proportionTrue function (GPL) 279
summary.range function (GPL) 282
summary.sd function (GPL) 284
summary.se function (GPL) 286
summary.se.kurtosis function (GPL) 288
summary.se.skewness function (GPL) 290
summary.sum function (GPL) 292, 294
summary.variance function (GPL) 296
syntax rules
GPL 2

T

t function (GPL) 298
text.footnote guide (GPL) 41
text.subfootnote guide (GPL) 41
text.subsubfootnote guide (GPL) 42
text.subsubtitle guide (GPL) 42
text.subtitle guide (GPL) 42
text.title guide (GPL) 43
texture.pattern function (GPL) 298
ticks function (GPL) 299
time scale (GPL) 37
title 2 (GPL) 42
title 3 (GPL) 42
title guide (GPL) 43
to function (GPL) 300
TRANS statement (GPL) 22
transparency function (GPL) 300, 301
transpose function (GPL) 301

U

uniform function (GPL) 302
unit.percent function (GPL) 302
unity variable (GPL) 3
user constants (GPL) 3
userSource function (GPL) 303

V

values function (GPL) 303
visible function (GPL) 304

W

weibull function (GPL) 304
Weibull scale (GPL) 30
weight function (GPL) 305
wrap function (GPL) 305