Contents

1 Introduction to GPL

The Basics ................................................................. 1
GPL Syntax Rules .......................................................... 3
GPL Concepts .............................................................. 3
  Brief Overview of GPL Algebra. ................................. 4
  How Coordinates and the GPL Algebra Interact .......... 7
Common Tasks ............................................................. 13
  How to Add Stacking to a Graph ................................. 13
  How to Add Faceting (Paneling) to a Graph ................ 15
  How to Add Clustering to a Graph .............................. 16
  How to Use Aesthetics .............................................. 18

2 GPL Statement and Function Reference

GPL Statements ........................................................ 21
  COMMENT Statement ............................................... 22
  PAGE Statement ..................................................... 22
  GRAPH Statement .................................................. 23
  SOURCE Statement ................................................. 24
  DATA Statement .................................................... 24
  TRANS Statement ................................................... 25
  COORD Statement .................................................. 25
  SCALE Statement ................................................... 30
  GUIDE Statement ................................................... 41
  ELEMENT Statement ............................................... 47
GPL Functions ........................................................ 59
  aestheticMaximum Function ................................. 61
  aestheticMinimum Function ................................. 62
  aestheticMissing Function ................................. 63
  alpha Function ................................................ 64
  base Function .................................................. 64
  base.aesthetic Function .................................. 65
  base.all Function ........................................... 65
  base.coordinate Function ................................. 66
  begin Function (For GPL Graphs) ......................... 67
  begin Function (For GPL Pages) ............................. 68
  beta Function ................................................ 68
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>format.date Function</td>
<td>118</td>
</tr>
<tr>
<td>format.dateTime Function</td>
<td>119</td>
</tr>
<tr>
<td>format.time Function</td>
<td>119</td>
</tr>
<tr>
<td>from Function</td>
<td>119</td>
</tr>
<tr>
<td>gamma Function</td>
<td>120</td>
</tr>
<tr>
<td>gap Function</td>
<td>120</td>
</tr>
<tr>
<td>gridlines Function</td>
<td>121</td>
</tr>
<tr>
<td>in Function</td>
<td>121</td>
</tr>
<tr>
<td>include Function</td>
<td>122</td>
</tr>
<tr>
<td>index Function</td>
<td>123</td>
</tr>
<tr>
<td>iter Function</td>
<td>123</td>
</tr>
<tr>
<td>jump Function</td>
<td>124</td>
</tr>
<tr>
<td>label Function (For GPL Graphic Elements)</td>
<td>124</td>
</tr>
<tr>
<td>label Function (For GPL Guides)</td>
<td>125</td>
</tr>
<tr>
<td>layout.circle Function</td>
<td>126</td>
</tr>
<tr>
<td>layout.dag Function</td>
<td>127</td>
</tr>
<tr>
<td>layout.data Function</td>
<td>129</td>
</tr>
<tr>
<td>layout.grid Function</td>
<td>130</td>
</tr>
<tr>
<td>layout.network Function</td>
<td>132</td>
</tr>
<tr>
<td>layout.random Function</td>
<td>133</td>
</tr>
<tr>
<td>layout.tree Function</td>
<td>135</td>
</tr>
<tr>
<td>link.alpha Function</td>
<td>136</td>
</tr>
<tr>
<td>link.complete Function</td>
<td>138</td>
</tr>
<tr>
<td>link.delaunay Function</td>
<td>139</td>
</tr>
<tr>
<td>link.distance Function</td>
<td>141</td>
</tr>
<tr>
<td>link.gabriel Function</td>
<td>142</td>
</tr>
<tr>
<td>link.hull Function</td>
<td>144</td>
</tr>
<tr>
<td>link.influence Function</td>
<td>145</td>
</tr>
<tr>
<td>link.join Function</td>
<td>147</td>
</tr>
<tr>
<td>link.mst Function</td>
<td>148</td>
</tr>
<tr>
<td>link.neighbor Function</td>
<td>150</td>
</tr>
<tr>
<td>link.relativeNeighborhood Function</td>
<td>151</td>
</tr>
<tr>
<td>link.sequence Function</td>
<td>153</td>
</tr>
<tr>
<td>link.tsp Function</td>
<td>154</td>
</tr>
<tr>
<td>logistic Function</td>
<td>156</td>
</tr>
<tr>
<td>map Function</td>
<td>156</td>
</tr>
<tr>
<td>marron Function</td>
<td>157</td>
</tr>
<tr>
<td>max Function</td>
<td>157</td>
</tr>
<tr>
<td>min Function</td>
<td>158</td>
</tr>
<tr>
<td>mirror Function</td>
<td>158</td>
</tr>
<tr>
<td>missing.gap Function</td>
<td>159</td>
</tr>
<tr>
<td>missing.interpulate Function</td>
<td>159</td>
</tr>
<tr>
<td>missing.listwise Function</td>
<td>160</td>
</tr>
<tr>
<td>missing.pairwise Function</td>
<td>160</td>
</tr>
<tr>
<td>missing.wings Function</td>
<td>161</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>sort.data Function</td>
<td>203</td>
</tr>
<tr>
<td>sort.natural Function</td>
<td>204</td>
</tr>
<tr>
<td>sort.statistic Function</td>
<td>204</td>
</tr>
<tr>
<td>sort.values Function</td>
<td>205</td>
</tr>
<tr>
<td>split Function</td>
<td>205</td>
</tr>
<tr>
<td>sqlSource Function</td>
<td>206</td>
</tr>
<tr>
<td>start Function</td>
<td>206</td>
</tr>
<tr>
<td>startAngle Function</td>
<td>207</td>
</tr>
<tr>
<td>studentizedRange Function</td>
<td>207</td>
</tr>
<tr>
<td>summary.count Function</td>
<td>208</td>
</tr>
<tr>
<td>summary.count.cumulative Function</td>
<td>210</td>
</tr>
<tr>
<td>summary.countTrue Function</td>
<td>211</td>
</tr>
<tr>
<td>summary.first Function</td>
<td>213</td>
</tr>
<tr>
<td>summary.kurtosis Function</td>
<td>214</td>
</tr>
<tr>
<td>summary.last Function</td>
<td>216</td>
</tr>
<tr>
<td>summary.max Function</td>
<td>217</td>
</tr>
<tr>
<td>summary.mean Function</td>
<td>219</td>
</tr>
<tr>
<td>summary.median Function</td>
<td>220</td>
</tr>
<tr>
<td>summary.min Function</td>
<td>222</td>
</tr>
<tr>
<td>summary.mode Function</td>
<td>223</td>
</tr>
<tr>
<td>summary.percent Function</td>
<td>225</td>
</tr>
<tr>
<td>summary.percent.count Function</td>
<td>226</td>
</tr>
<tr>
<td>summary.percent.count.cumulative Function</td>
<td>227</td>
</tr>
<tr>
<td>summary.percent.cumulative Function</td>
<td>229</td>
</tr>
<tr>
<td>summary.percent.sum Function</td>
<td>229</td>
</tr>
<tr>
<td>summary.percent.sum.cumulative Function</td>
<td>231</td>
</tr>
<tr>
<td>summary.percentile Function</td>
<td>233</td>
</tr>
<tr>
<td>summary.percentTrue Function</td>
<td>234</td>
</tr>
<tr>
<td>summary.proportion Function</td>
<td>236</td>
</tr>
<tr>
<td>summary.proportion.count Function</td>
<td>237</td>
</tr>
<tr>
<td>summary.proportion.count.cumulative Function</td>
<td>238</td>
</tr>
<tr>
<td>summary.proportion.cumulative Function</td>
<td>240</td>
</tr>
<tr>
<td>summary.proportion.sum Function</td>
<td>240</td>
</tr>
<tr>
<td>summary.proportion.sum.cumulative Function</td>
<td>242</td>
</tr>
<tr>
<td>summary.proportionTrue Function</td>
<td>244</td>
</tr>
<tr>
<td>summary.range Function</td>
<td>245</td>
</tr>
<tr>
<td>summary.sd Function</td>
<td>247</td>
</tr>
<tr>
<td>summary.se Function</td>
<td>248</td>
</tr>
<tr>
<td>summary.se.kurtosis Function</td>
<td>250</td>
</tr>
<tr>
<td>summary.se.skewness Function</td>
<td>251</td>
</tr>
<tr>
<td>summary.sum Function</td>
<td>253</td>
</tr>
<tr>
<td>summary.sum.cumulative Function</td>
<td>254</td>
</tr>
<tr>
<td>summary.variance Function</td>
<td>256</td>
</tr>
<tr>
<td>t Function</td>
<td>257</td>
</tr>
<tr>
<td>texture.pattern Function</td>
<td>258</td>
</tr>
</tbody>
</table>
3 GPL Examples

Using the Examples in Your Application .................................................. 267
Summary Bar Chart Examples ................................................................. 268
   Simple Bar Chart .............................................................................. 269
   Simple Bar Chart of Counts ............................................................ 269
   Simple Horizontal Bar Chart ........................................................... 270
   Simple Bar Chart With Error Bars .................................................... 271
   Simple Bar Chart with Bar for All Categories .................................... 272
   Stacked Bar Chart .......................................................................... 273
   Clustered Bar Chart ......................................................................... 274
   Clustered and Stacked Bar Chart ...................................................... 276
   Bar Chart Using an Evaluation Function ........................................... 277
   Bar Chart with Mapped Aesthetics .................................................... 279
   Faceted (Paneled) Bar Chart ............................................................ 280
   3-D Bar Chart .................................................................................. 282
   Error Bar Chart ................................................................................ 283

Histogram Examples ................................................................................ 284
   Histogram ....................................................................................... 284
   Histogram with Distribution Curve .................................................. 285
   Percentage Histogram .................................................................... 287
   Frequency Polygon .......................................................................... 288
   Stacked Histogram .......................................................................... 289
   Faceted (Paneled) Histogram .......................................................... 290
   Population Pyramid .......................................................................... 291
   Cumulative Histogram ..................................................................... 292
   3-D Histogram ................................................................................. 293
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-Low Chart Examples</td>
<td>294</td>
</tr>
<tr>
<td>Simple Range Bar for One Variable</td>
<td>294</td>
</tr>
<tr>
<td>Simple Range Bar for Two Variables</td>
<td>295</td>
</tr>
<tr>
<td>High-Low-Close Chart</td>
<td>296</td>
</tr>
<tr>
<td>Scatter/Dot Examples</td>
<td>297</td>
</tr>
<tr>
<td>Simple 1-D Scatterplot</td>
<td>297</td>
</tr>
<tr>
<td>Simple 2-D Scatterplot</td>
<td>298</td>
</tr>
<tr>
<td>Simple 2-D Scatterplot with Fit Line</td>
<td>299</td>
</tr>
<tr>
<td>Grouped Scatterplot</td>
<td>300</td>
</tr>
<tr>
<td>Grouped Scatterplot with Convex Hull</td>
<td>301</td>
</tr>
<tr>
<td>Scatterplot Matrix (SPLOM)</td>
<td>302</td>
</tr>
<tr>
<td>Bubble Plot</td>
<td>303</td>
</tr>
<tr>
<td>Binned Scatterplot</td>
<td>304</td>
</tr>
<tr>
<td>Binned Scatterplot with Polygons</td>
<td>305</td>
</tr>
<tr>
<td>Scatterplot with Border Histograms</td>
<td>306</td>
</tr>
<tr>
<td>Scatterplot with Border Boxplots</td>
<td>307</td>
</tr>
<tr>
<td>Dot Plot</td>
<td>308</td>
</tr>
<tr>
<td>2-D Dot Plot</td>
<td>310</td>
</tr>
<tr>
<td>Jittered Categorical Scatterplot</td>
<td>312</td>
</tr>
<tr>
<td>Line Chart Examples</td>
<td>313</td>
</tr>
<tr>
<td>Simple Line Chart</td>
<td>313</td>
</tr>
<tr>
<td>Simple Line Chart with Points</td>
<td>314</td>
</tr>
<tr>
<td>Line Chart of Date Data</td>
<td>315</td>
</tr>
<tr>
<td>Line Chart With Step Interpolation</td>
<td>316</td>
</tr>
<tr>
<td>Fit Line</td>
<td>317</td>
</tr>
<tr>
<td>Line Chart from Equation</td>
<td>318</td>
</tr>
<tr>
<td>Line Chart with Separate Scales</td>
<td>320</td>
</tr>
<tr>
<td>Line Chart in Parallel Coordinates</td>
<td>320</td>
</tr>
<tr>
<td>Pie Chart Examples</td>
<td>322</td>
</tr>
<tr>
<td>Pie Chart</td>
<td>322</td>
</tr>
<tr>
<td>Paneled Pie Chart</td>
<td>324</td>
</tr>
<tr>
<td>Stacked Pie Chart</td>
<td>324</td>
</tr>
<tr>
<td>Boxplot Examples</td>
<td>325</td>
</tr>
<tr>
<td>1-D Boxplot</td>
<td>325</td>
</tr>
<tr>
<td>Boxplot</td>
<td>326</td>
</tr>
<tr>
<td>Clustered Boxplot</td>
<td>328</td>
</tr>
<tr>
<td>Boxplot With Overlaid Dot Plot</td>
<td>329</td>
</tr>
<tr>
<td>Multi-Graph Examples</td>
<td>330</td>
</tr>
<tr>
<td>Scatterplot with Border Histograms</td>
<td>330</td>
</tr>
<tr>
<td>Scatterplot with Border Boxplots</td>
<td>331</td>
</tr>
<tr>
<td>Stocks Line Chart with Volume Bar Chart</td>
<td>332</td>
</tr>
<tr>
<td>Dual Axis Graph</td>
<td>333</td>
</tr>
<tr>
<td>Histogram with Dot Plot</td>
<td>334</td>
</tr>
</tbody>
</table>
Appendix

A GPL Constants

Color Constants ............................................................. 344
Shape Constants ............................................................ 345
Size Constants .............................................................. 345
Pattern Constants ............................................................ 346

Bibliography

Index
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* (Wilkinson, 2005).

The Basics

The GPL example below creates a simple bar chart. (See Figure 1-2 on p. 2.) 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* in Chapter 3 on p. 267.

Figure 1-1
GPL for a simple bar chart

```plaintext
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)))
```
Chapter 1

Figure 1-2
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 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. For more information, see the topic Brief Overview of GPL Algebra on p. 4.

Details about all of the statements and functions appear in GPL Statement and Function Reference on p. 21.

**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 GPL Examples on p. 267.
Chapter 1

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 p. 7 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,

\[ X \times Y \neq Y \times X \]
\[ X / Y \neq Y / X \]

**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:

\[(A+B) \times C\]

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

\[(A\times C) + (B\times D)\]

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 \((A+B) \times (C+D)\) won’t work. \((A\times C) + (B\times D)\) 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)))`
**Chapter 1**

**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 p. 320.

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 p. 272.

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 1-3
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 1-4
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)))
Introduction to GPL

Figure 1-5
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)))
```
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**

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

**Full Specification**

```plaintext
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)))
```
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)))
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))
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 p. 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 p. 1 as a “baseline” for the changes.
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())
```

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))
```

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:

```plaintext
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 1-10
*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 p. 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 1-11
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 p. 1 as a “baseline” for the changes.

- 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())
  ```

- 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.

- 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))
  ```
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.

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

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

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. Compare this to “Faceted bar chart” on p. 15.

Figure 1-12
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.

Figure 1-13
Baseline GPL for example

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))

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())

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"))

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.
Finally add the aesthetic for the continuous variable:

```plaintext
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.

```plaintext
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.

**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:

```plaintext
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.
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 (GPL), DATA Statement (GPL), TRANS Statement (GPL)

Specification Statements

COORD Statement (GPL), SCALE Statement (GPL), GUIDE Statement (GPL), ELEMENT Statement (GPL)

Control Statements

PAGE Statement (GPL), GRAPH Statement (GPL)
Comment Statements

COMMENT Statement (GPL)

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

Figure 2-1
Defining a comment

COMMENT: This graph shows counts for each job category.

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.

Examples

Figure 2-2
Example: Defining a page

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 2-3
Example: Defining a page with multiple graphs

```
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()
```

Valid Functions
begin Function (For GPL Pages), end Function (GPL)

**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.

**Examples**

**Figure 2-4**
*Scaling a graph*

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

**Figure 2-5**
*Example: Scatterplot with border histograms*

```
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()
```
Chapter 2

Valid Functions

begin Function (For GPL Graphs), end Function (GPL)

SOURCE Statement

Syntax

SOURCE: <source name> = <function>

<source name>. User-defined name for the data source. Refer to GPL Syntax Rules on p. 3 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

Figure 2-6
Example: Reading a CSV file

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

Valid Functions

csvSource Function (GPL), savSource Function (GPL), sqlSource Function (GPL), userSource Function (GPL)

DATA Statement

Syntax

DATA: <variable name> = <function>

<variable name>. User-defined name for the variable. Refer to GPL Syntax Rules on p. 3 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

Figure 2-7  
Example: Specifying a variable from a data source

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

*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 (GPL), iter Function (GPL)

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 p. 3 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

Figure 2-8  
Example: Creating a transformation variable from other variables

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

Figure 2-9  
Example: Creating an index variable

TRANS: casenum = index()

Valid Functions

collapse Function (GPL), eval Function (GPL), index Function (GPL)

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**

**Figure 2-10**
Example: Polar coordinates for pie charts

COORD: polar.theta()

**Figure 2-11**
Example: 3-D rectangular coordinates

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

**Figure 2-12**
Example: Embedded coordinate systems for paneled pie chart

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

**Figure 2-13**
Example: Transposed coordinate system

COORD: transpose()

**Coordinate Types and Transformations**

parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), mirror Function (GPL), project Function (GPL), reflect Function (GPL), transpose Function (GPL), wrap Function (GPL)

**GPL Coordinate Types**

There are several coordinate types available in GPL.

**Coordinate Types**

parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), mirror Function (GPL), project Function (GPL), reflect Function (GPL), transpose Function (GPL), wrap Function (GPL)

**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. For more information, see the topic dim Function on p. 109.

<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**

Figure 2-14
Example: Parallel coordinates graph

```
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.

**Coordinate Types and Transformations**

polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), mirror Function (GPL), project Function (GPL), reflect Function (GPL), transpose Function (GPL), wrap Function (GPL)

**Applies To**

COORD Statement (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), project Function (GPL)

**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. For more information, see the topic dim Function on p. 109.

<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

Figure 2-15
Example: Polar line chart

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

Valid Functions

reverse Function (GPL), startAngle Function (GPL)

Coordinate Types and Transformations

parallel Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), mirror Function (GPL), project Function (GPL), reflect Function (GPL), transpose Function (GPL), wrap Function (GPL)

Applies To

COORD Statement (GPL), parallel Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), project Function (GPL)

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. For more information, see the topic dim Function on p. 109.

<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**

Figure 2-16
Example: Pie chart

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

**Valid Functions**

reverse Function (GPL), startAngle Function (GPL)

**Coordinate Types and Transformations**

parallel Coordinate Type (GPL), polar Coordinate Type (GPL), rect Coordinate Type (GPL), mirror Function (GPL), project Function (GPL), reflect Function (GPL), transpose Function (GPL), wrap Function (GPL)

**Applies To**

COORD Statement (GPL), parallel Coordinate Type (GPL), polar Coordinate Type (GPL), rect Coordinate Type (GPL), project Function (GPL)

**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)). For more information, see the topic `dim Function` on p. 109.

**<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**

Figure 2-17
Example: 2-D bar chart

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

Figure 2-18
Example: 3-D bar chart

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

**Valid Functions**

cluster Function (GPL), sameRatio Function (GPL)

**Coordinate Types and Transformations**

parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), mirror Function (GPL), project Function (GPL), reflect Function (GPL), transpose Function (GPL), wrap Function (GPL)

**Applies To**

COORD Statement (GPL), parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), project Function (GPL)

**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 p. 3 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

Figure 2-19
Example: Specifying a linear dimension scale

\[
\text{SCALE: linear}(\text{dim}(2), \text{max}(50000))
\]

Figure 2-20
Example: Specifying a log aesthetic scale

\[
\text{SCALE: log(aesthetic(aesthetic.color))}
\]
\[
\text{ELEMENT: point(position(x*y), color(z))}
\]

Figure 2-21
Example: Creating a graph with dual scales

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

Scale Types

asn Scale (GPL), atanh Scale (GPL), cat Scale (GPL), linear Scale (GPL), log Scale (GPL), logit Scale (GPL), pow Scale (GPL), prob Scale (GPL), probit Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

GPL Scale Types

There are several scale types available in GPL.

Scale Types

asn Scale (GPL), atanh Scale (GPL), cat Scale (GPL), linear Scale (GPL), log Scale (GPL), logit Scale (GPL), pow Scale (GPL), prob Scale (GPL), probit Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

asn Scale

Syntax

\[
\text{asn(dim(<numeric>), <function>)}
\]

or

\[
\text{asn(aesthetic(aesthetic.<aesthetic type>), <function>)}
\]

<numeric>. A numeric value indicating the dimension to which the scale applies. For more information, see the topic dim Function on p. 109.

<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.
Chapter 2

**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**

Figure 2-22  
Example: Specifying an arcsine scale

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

**Valid Functions**

-aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing Function (GPL)

**Applies To**

SCALE Statement (GPL)

**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. For more information, see the topic dim Function on p. 109.
- **<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 hyperbolic 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**

Figure 2-23  
Example: Specifying a Fisher’s z scale

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

aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing Function (GPL)

Applies To
SCALE Statement (GPL)

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. For more information, see the topic dim Function on p. 109.

<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

Figure 2-24
Example: Specifying the scale for a dimension and sorting categories
SCALE: cat(dim(1), sort.natural())

Figure 2-25
Example: Specifying the scale for an aesthetic and including categories
SCALE: cat(aesthetic(aesthetic.color), include("IL"))

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

Valid Functions

aestheticMissing Function (GPL), values Function (GPL), exclude Function (GPL), include Function (GPL), map Function (GPL), reverse Function (GPL), sort.data Function (GPL), sort.natural Function (GPL), sort.statistic Function (GPL), sort.values Function (GPL)
Applies To

SCALE Statement (GPL)

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. For more information, see the topic dim Function on p. 109.

<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

Figure 2.26
Example: Specifying a maximum value for the linear scale

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

Valid Functions

aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing Function (GPL), dataMaximum Function (GPL), dataMinimum Function (GPL), include Function (GPL), max Function (GPL), min Function (GPL), origin Function (For GPL Scales), reverse Function (GPL)

Applies To

SCALE Statement (GPL)

log Scale

Syntax

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

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

<numeric>. A numeric value indicating the dimension to which the scale applies. For more information, see the topic dim Function on p. 109.

<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 scale. If a base is not explicitly specified, the default is base 10. Data values for this scale must be greater than 0.

**Example**

Figure 2-27
*Example: Specifying the scale and a base*

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

**Valid Functions**

aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing Function (GPL), dataMaximum Function (GPL), dataMinimum Function (GPL), include Function (GPL), base Function (GPL), max Function (GPL), min Function (GPL), origin Function (For GPL Scales), reverse Function (GPL)

**Applies To**

SCALE Statement (GPL)

**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. For more information, see the topic dim Function on p. 109.

<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.
Chapter 2

**Description**

Creates a logit scale. Data values for this scale must fall in the open interval (0, 1). That is, for any data value \( x \), \( 0 < x < 1 \).

**Example**

Figure 2-28
*Example: Specifying a logit scale*

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

**Valid Functions**

- aestheticMaximum Function (GPL)
- aestheticMinimum Function (GPL)
- aestheticMissing Function (GPL)

**Applies To**

- SCALE Statement (GPL)

**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. For more information, see the topic dim Function on p. 109.

**<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. If an exponent is not explicitly specified, the default is 0.5.

**Example**

Figure 2-29
*Example: Specifying the scale and an exponent*

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

aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing
Function (GPL), dataMaximum Function (GPL), dataMinimum Function (GPL), exponent
Function (GPL), include Function (GPL), max Function (GPL), min Function (GPL), origin
Function (For GPL Scales), reverse Function (GPL)

Applies To
SCALE Statement (GPL)

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. For more
information, see the topic dim Function on p. 109.
<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 a probability 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

Figure 2-30
Example: Specifying a beta distribution for the probability scale

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

Distribution Functions

beta Function (GPL), chiSquare Function (GPL), exponential Function (GPL), f Function (GPL),
gamma Function (GPL), logistic Function (GPL), normal Function (GPL), poisson Function
(GPL), studentizedRange Function (GPL), t Function (GPL), uniform Function (GPL), weibull
Function (GPL)

Valid Functions

aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing
Function (GPL)
Applies To

SCALE Statement (GPL)

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. For more information, see the topic dim Function on p. 109.

<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. 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

Figure 2-31
Example: Specifying a probit scale

probit(dim(2))

Valid Functions

aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing Function (GPL)

Applies To

SCALE Statement (GPL)

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. For more information, see the topic dim Function on p. 109.

<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 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.

The safe log formula is:

\[ \text{sign}(x) \times \log(1 + \text{abs}(x)) \]

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

\[ \text{sign}(-99) \times \log(1 + \text{abs}(-99)) = -1 \times \log(1 + 99) = -1 \times 2 = -2 \]

Example

Figure 2-32
Example: Specifying the scale and a base

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

Valid Functions

aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing Function (GPL), dataMaximum Function (GPL), dataMinimum Function (GPL), include Function (GPL), base Function (GPL), max Function (GPL), min Function (GPL), origin Function (For GPL Scales), reverse Function (GPL)

Applies To

SCALE Statement (GPL)

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. For more information, see the topic dim Function on p. 109.

<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 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.

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

If \(x \geq 0\):

\[
pow(1+x, \text{exponent}) - 1
\]

If \(x < 0\):

\[
1 - pow(1-x, \text{exponent})
\]

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

If \(x \geq 0\):

\[
1 - pow(1+x, \text{exponent})
\]

If \(x < 0\):

\[
pow(1-x, \text{exponent}) - 1
\]

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**

Figure 2-33

*Example: Specifying the scale and an exponent*

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

**Valid Functions**

- `aestheticMaximum Function (GPL)`
- `aestheticMinimum Function (GPL)`
- `aestheticMissing Function (GPL)`
- `dataMaximum Function (GPL)`
- `dataMinimum Function (GPL)`
- `exponent Function (GPL)`
- `include Function (GPL)`
- `max Function (GPL)`
- `min Function (GPL)`
- `origin Function (For GPL Scales)`
- `reverse Function (GPL)`

**Applies To**

- `SCALE Statement (GPL)`

**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. For more information, see the topic dim Function on p. 109.

<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**

Figure 2-34

*Example: Specifying a maximum value for the time scale*

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

**Valid Functions**

aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing Function (GPL), dataMaximum Function (GPL), dataMinimum Function (GPL), include Function (GPL), max Function (GPL), min Function (GPL), origin Function (For GPL Scales), reverse Function (GPL)

**Applies To**

SCALE Statement (GPL)

**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**

Figure 2-35

*Example: Axis*

GUIDE: axis(dim(2), label("Mean Current Salary"))
**Guide Types**

axis Guide Type (GPL), form.line Guide Type (GPL), legend Guide Type (GPL), text.footnote Guide Type (GPL), text.subfootnote Guide Type (GPL), text.subsubfootnote Guide Type (GPL), text.subtitle Guide Type (GPL), text.subsubtitle Guide Type (GPL), text.title Guide Type (GPL)

**GPL Guide Types**

There are several different types of guides.

**Guide Types**

axis Guide Type (GPL), form.line Guide Type (GPL), legend Guide Type (GPL), text.footnote Guide Type (GPL), text.subfootnote Guide Type (GPL), text.subsubfootnote Guide Type (GPL), text.subtitle Guide Type (GPL), text.subsubtitle Guide Type (GPL), text.title Guide Type (GPL)

**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. For more information, see the topic dim Function on p. 109.

**<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. For more information, see the topic scale Function (For GPL Axes) on p. 184.

**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 p. 30.

**Examples**

**Figure 2-36**

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

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

**Figure 2-37**

Example: Specifying axis labels for a dimension with two scales

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))

Valid Functions

color Function (For GPL Guides), delta Function (GPL), gap Function (GPL), gridlines Function (GPL), label Function (For GPL Guides), opposite Function (GPL), format.date Function (GPL), format.dateTime Function (GPL), format.time Function (GPL), unit.percent Function (GPL), start Function (GPL), ticks Function (GPL)

Applies To

GUIDE Statement (GPL)

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 p. 167.

Examples

Figure 2-38
Example: Horizontal reference line

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

Figure 2-39
Example: Vertical reference line

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

Valid Functions

color.brightness Function (For GPL Guides), color Function (For GPL Guides), color.hue Function (For GPL Guides), label Function (For GPL Guides), position Function (For GPL Guides), color.saturation Function (For GPL Guides), scale Function (For GPL Graphic Elements and form.line), shape Function (For GPL Guides), size Function (For GPL Guides), transparency Function (For GPL Guides)

Applies To

GUIDE Statement (GPL)
**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**

Figure 2-40
Example: Specifying a legend title

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

**Valid Functions**

label Function (For GPL Guides)

**Applies To**

GUIDE Statement (GPL)

**text.footnote Guide Type**

**Syntax**

text.footnote(<function>)

<function>. One or more valid functions.

**Description**

Specifies the footnote for the graph.
Examples

Figure 2-41
Example: Specifying the footnote text

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

Valid Functions

label Function (For GPL Guides)

Applies To

GUIDE Statement (GPL)

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

Figure 2-42
Example: Specifying the second-level footnote text

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

Valid Functions

label Function (For GPL Guides)

Applies To

GUIDE Statement (GPL)

text.subsubfootnote Guide Type

Syntax

text.subsubfootnote(<function>)

<function>. One or more valid functions.
Chapter 2

Description

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

Examples

Figure 2-43
Example: Specifying the third-level footnote text

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

Valid Functions

label Function (For GPL Guides)

Applies To

GUIDE Statement (GPL)

text.subtitle Guide Type

Syntax

text.subtitle(<function>)

<function>. One or more valid functions.

Description

Specifies the subtitle for the graph.

Examples

Figure 2-44
Example: Specifying the subtitle text

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

Valid Functions

label Function (For GPL Guides)

Applies To

GUIDE Statement (GPL)

text.subsubtitle Guide Type

Syntax

text.subsubtitle(<function>)
<function>. One or more valid functions.

**Description**

Specifies the second-level subtitle for the graph.

**Examples**

Figure 2-45
*Example: Specifying the second-level subtitle text*

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

**Valid Functions**

*label Function (For GPL Guides)*

**Applies To**

GUIDE Statement (GPL)

---

**text.title Guide Type**

**Syntax**

text.title(<function>)

<function>. One or more valid functions.

**Description**

Specifies the title for the graph.

**Examples**

Figure 2-46
*Example: Specifying the title text*

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

**Valid Functions**

*label Function (For GPL Guides)*

**Applies To**

GUIDE Statement (GPL)

---

**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:

```plaintext
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:

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

**Examples**

**Figure 2-47**
Example: Scatterplot

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

**Figure 2-48**
Example: Line chart

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

**Figure 2-49**
Example: Bar chart of means

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

**Graphic Element Types**

area Element (GPL), edge Element (GPL), interval Element (GPL), line Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL)

**GPL Graphic Element Types**

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

**area Element**

**Syntax**

```plaintext
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

Figure 2-50
Example: Area chart

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

Collision Modifiers

difference Collision Modifier, stack Collision Modifier

Valid Functions

color.brightness Function (For GPL Graphic Elements), closed Function (GPL), color Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), jump Function (GPL), label Function (For GPL Graphic Elements), missing.gap Function (GPL), missing.interpolate Function (GPL), missing.wings Function (GPL), position Function (For GPL Graphic Elements), preserveStraightLines Function (GPL), color.saturation Function (For GPL Graphic Elements), scale Function (For GPL Graphic Elements and form.line), split Function (GPL), transparency Function (For GPL Graphic Elements), visible Function (GPL)

Applies To

ELEMENT Statement (GPL)

bar Element

Description

bar is an alias for interval. For the syntax and other examples, see interval Element on p. 50.

Examples

Figure 2-51
Example: Bar chart

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

Syntax

dedge(<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

Figure 2-52
Example: Minimum spanning tree

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

Figure 2-53
Example: Convex hull

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

Valid Functions

color.brightness Function (For GPL Graphic Elements), closed Function (GPL), color Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), label Function (For GPL Graphic Elements), missing.gap Function (GPL), missing.interpolate Function (GPL), missing.wings Function (GPL), position Function (For GPL Graphic Elements), preserveStraightLines Function (GPL), color.saturation Function (For GPL Graphic Elements), scale Function (For GPL Graphic Elements and form.line), shape Function (For GPL Graphic Elements), size Function (For GPL Graphic Elements), split Function (GPL), transparency Function (For GPL Graphic Elements), visible Function (GPL)

Applies To

ELEMENT Statement (GPL)

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

Figure 2-54
Example: Bar chart

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

Figure 2-55
Example: Histogram

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

Figure 2-56
Example: Error bar chart

ELEMENT: interval(position(region.conf.i.mean(jobcat*salary)), shape(shape.ibeam))

Figure 2-57
Example: Stacked bar chart

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

Collision Modifiers

dodge Collision Modifier, stack Collision Modifier

Valid Functions

color.brightness Function (For GPL Graphic Elements), color Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), label Function (For GPL Graphic Elements), texture.pattern Function (GPL), position Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), scale Function (For GPL Graphic Elements and form.line), shape Function (For GPL Graphic Elements), size Function (For GPL Graphic Elements), split Function (GPL), transparency Function (For GPL Graphic Elements), visible Function (GPL)

Applies To

ELEMENT Statement (GPL)

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).
Chapter 2

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 p. 52 for more information about the path graphic element.

Examples

Figure 2-58
Example: Line chart of continuous variables

ELEMENT: line(position(salbegin*salary))

Figure 2-59
Example: Line chart of a summary statistic

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

Collision Modifiers

stack Collision Modifier

Valid Functions

color.brightness Function (For GPL Graphic Elements), closed Function (GPL), color Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), jump Function (GPL), label Function (For GPL Graphic Elements), missing.gap Function (GPL), missing.interpolate Function (GPL), missing.wings Function (GPL), position Function (For GPL Graphic Elements), preserveStraightLines Function (GPL), color.saturation Function (For GPL Graphic Elements), scale Function (For GPL Graphic Elements and form.line), shape Function (For GPL Graphic Elements), size Function (For GPL Graphic Elements), split Function (GPL), transparency Function (For GPL Graphic Elements), visible Function (GPL)

Applies To

ELEMENT Statement (GPL)

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**

Figure 2-60
*Example: Line chart drawn through all values*

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

Figure 2-61
*Example: Creating a line chart with variable widths*

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

**Valid Functions**

- color.brightness Function (For GPL Graphic Elements)
- closed Function (GPL)
- color Function (For GPL Graphic Elements)
- color.hue Function (For GPL Graphic Elements)
- label Function (For GPL Graphic Elements)
- missing.gap Function (GPL)
- missing.interpolate Function (GPL)
- missing.wings Function (GPL)
- position Function (For GPL Graphic Elements)
- preserveStraightLines Function (GPL)
- color.saturation Function (For GPL Graphic Elements)
- scale Function (For GPL Graphic Elements and form.line)
- shape Function (For GPL Graphic Elements)
- size Function (For GPL Graphic Elements)
- split Function (GPL)
- transparency Function (For GPL Graphic Elements)
- visible Function (GPL)

**Applies To**

- ELEMENT Statement (GPL)

**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**

Figure 2-62
*Example: Scatterplot*

```
ELEMENT: point(position(salbegin*salary))
```
Chapter 2

Figure 2-63
Example: Dot plot

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

**Collision Modifiers**

dodge.asymmetric Collision Modifier, dodge.symmetric Collision Modifier, jitter Collision Modifier, stack Collision Modifier

**Valid Functions**

color.brightness Function (For GPL Graphic Elements), color Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), label Function (For GPL Graphic Elements), texture.pattern Function (GPL), position Function (For GPL Graphic Elements),
color.saturation Function (For GPL Graphic Elements and form.line), shape Function (For GPL Graphic Elements), size Function (For GPL Graphic Elements), split Function (GPL), transparency Function (For GPL Graphic Elements),
visible Function (GPL)

**Applies To**

ELEMENT Statement (GPL)

**polygon Element**

**Syntax**

```plaintext
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**

Figure 2-64
Example: Hexagonal binning

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

**Valid Functions**

color.brightness Function (For GPL Graphic Elements), color Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), label Function (For GPL Graphic Elements), position Function (For GPL Graphic Elements), preserveStraightLines Function (GPL),
color.saturation Function (For GPL Graphic Elements), scale Function (For GPL Graphic Elements),
visible Function (GPL)
Elements and form.line), shape Function (For GPL Graphic Elements), size Function (For GPL Graphic Elements), split Function (GPL), transparency Function (For GPL Graphic Elements), visible Function (GPL)

**Applies To**

ELEMENT Statement (GPL)

**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**

Figure 2-65

*Example: Boxplot*

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

**Valid Functions**

color.brightness Function (For GPL Graphic Elements), color Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), label Function (For GPL Graphic Elements), texture.pattern Function (GPL), position Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), scale Function (For GPL Graphic Elements and form.line), size Function (For GPL Graphic Elements), split Function (GPL), transparency Function (For GPL Graphic Elements), visible Function (GPL)

**Applies To**

ELEMENT Statement (GPL)

**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

Figure 2-66
Example: Difference area chart

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

Valid Element Types

area Element

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 p. 57.)

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

Figure 2-67
Example: 2-D Dot plot

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

Figure 2-68
Example: Dodged bar chart

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

Valid Element Types

interval Element, point Element
**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**

Figure 2-69

Example: 1-D Dot plot

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

**Valid Element Types**

point Element

**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**

Figure 2-70

Example: 2-D Dot plot

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

**Valid Element Types**

point Element
**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**

Figure 2-71
Example: Jittered categorical point chart

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

**Table 2-1**

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>joint</code></td>
<td>Same as <code>joint.uniform</code></td>
</tr>
<tr>
<td><code>conditional</code></td>
<td>Same as <code>conditional.uniform</code></td>
</tr>
<tr>
<td><code>normal</code></td>
<td>Same as <code>joint.normal</code></td>
</tr>
<tr>
<td><code>uniform</code></td>
<td>Same as <code>joint.uniform</code></td>
</tr>
<tr>
<td><code>joint.normal</code></td>
<td>Jitter points in all dimensions using a normal distribution</td>
</tr>
<tr>
<td><code>joint.uniform</code></td>
<td>Jitter points in all dimensions using a uniform distribution</td>
</tr>
<tr>
<td><code>conditional.normal</code></td>
<td>Jitter points in the analysis dimension using a normal distribution</td>
</tr>
<tr>
<td><code>conditional.uniform</code></td>
<td>Jitter points in the analysis dimension using a uniform distribution</td>
</tr>
</tbody>
</table>

**Valid Element Types**

point Element

**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

Figure 2-72  
Example: Stacked bar chart

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

Valid Element Types

area Element, interval Element, line Element, point Element

GPL Functions

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

Aesthetic Functions

color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), shape Function (For GPL Graphic Elements), size Function (For GPL Graphic Elements), texture.pattern Function (GPL), transparency Function (For GPL Graphic Elements), visible Function (GPL)

Aesthetic Mapping Functions

aestheticMaximum Function (GPL), aestheticMinimum Function (GPL), aestheticMissing Function (GPL), map Function (GPL)

Guide Aesthetic Functions

color Function (For GPL Guides), color.brightness Function (For GPL Guides), color.hue Function (For GPL Guides), color.saturation Function (For GPL Guides), shape Function (For GPL Guides), size Function (For GPL Guides), transparency Function (For GPL Guides)

Data Functions

col Function (GPL), iter Function (GPL)

Data Source Functions

csvSource Function (GPL), savSource Function (GPL), sqlSource Function (GPL), userSource Function (GPL)

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)
Chapter 2

**Graph Control Functions**

begin Function (For GPL Graphs), end Function (GPL)

**Missing Value Functions for Lines and Areas**

missing.gap Function (GPL), missing.interpolate Function (GPL), missing.wings Function (GPL)

**Percentage Base Functions**

base.aesthetic Function (GPL), base.all Function (GPL), base.coordinate Function (GPL)

**Probability Scale Distribution Functions**

beta Function (GPL), chiSquare Function (GPL), exponential Function (GPL), f Function (GPL),
gamma Function (GPL), logistic Function (GPL), normal Function (GPL), poisson Function (GPL),
studentizedRange Function (GPL), t Function (GPL), uniform Function (GPL), weibull Function (GPL)

**Sort Functions**

sort.data Function (GPL), sort.natural Function (GPL), sort.statistic Function (GPL), sort.values Function (GPL)

**Statistic Functions**

density.beta Function (GPL), density.chiSquare Function (GPL), density.exponential Function (GPL),
density.f Function (GPL), density.gamma Function (GPL), density.kernel Function (GPL),
density.logistic Function (GPL), density.normal Function (GPL), density.poisson Function (GPL),
density.studentizedRange Function (GPL), density.t Function (GPL), density.uniform Function (GPL),
density.weibull Function (GPL), layout.circle Function (GPL), layout.dag Function (GPL),
layout.data Function (GPL), layout.grid Function (GPL), layout.network Function (GPL),
layout.random Function (GPL), layout.tree Function (GPL), link.alpha Function (GPL),
link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL),
link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL),
link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL),
link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL),
region.conf.interval Function (GPL), region.conf.mean Function (GPL),
region.conf.percent.interval Function (GPL), region.conf.proportion Function (GPL),
region.conf.smooth Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL),
region.spread.se Function (GPL), smooth.cubic Function (GPL), smooth.linear Function (GPL),
smooth.loess Function (GPL), smooth.mean Function (GPL), smooth.median Function (GPL),
smooth.quadratic Function (GPL), smooth.spline Function (GPL), smooth.step Function (GPL),
summary.count Function (GPL), summary.cumulative Function (GPL),
summary.first Function (GPL), summary.last Function (GPL), summary.max Function (GPL),
summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL),
summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.cumulative Function (GPL),
summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative
Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL)

Transformation Functions

collapse Function (GPL), eval Function (GPL), index Function (GPL)

Other Functions

alpha Function (GPL), base Function (GPL), begin Function (For GPL Graphs), begin Function (For GPL Pages), binCount Function (GPL), binStart Function (GPL), binWidth Function (GPL), closed Function (GPL), cluster Function (GPL), dataMaximum Function (GPL), dataMinimum Function (GPL), delta Function (GPL), dim Function (GPL), end Function (GPL), exclude Function (GPL), exponent Function (GPL), format Function (GPL), format.date Function (GPL), format.dateTime Function (GPL), format.time Function (GPL), from Function (GPL), gap Function (GPL), gridlines Function (GPL), in Function (GPL), include Function (GPL), jump Function (GPL), label Function (For GPL Graphic Elements), label Function (For GPL Guides), marron Function (GPL), max Function (GPL), min Function (GPL), missing.listwise Function (GPL), missing.pairwise Function (GPL), multiple Function (GPL), noConstant Function (GPL), node Function (GPL), origin Function (GPL), origin Function (For GPL Graphs), origin Function (For GPL Scales), position Function (For GPL Graphic Elements), position Function (For GPL Guides), preserveStraightLines Function (GPL), proportion Function (GPL), reverse Function (GPL), root Function (GPL), sameRatio Function (GPL), scale Function (For GPL Axes), scale Function (For GPL Graphs), scale Function (For GPL Graphic Elements and form.line), scale Function (For GPL Pages), segments Function (GPL), showAll Function (GPL), split Function (GPL), start Function (GPL), startDate Function (GPL), ticks Function (GPL), to Function (GPL), unit.percent Function (GPL), values Function (GPL), weight Function (GPL)

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**

Figure 2-73
Example: Specifying a minimum and maximum size for points in a bubble plot

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

**Applies To**

asn Scale (GPL), atanh Scale (GPL), linear Scale (GPL), log Scale (GPL), logit Scale (GPL), pow Scale (GPL), prob Scale (GPL), probit Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

**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

Figure 2-74
Example: Specifying a minimum and maximum size for points in a bubble plot

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

Applies To

asn Scale (GPL), atanh Scale (GPL), linear Scale (GPL), log Scale (GPL), logit Scale (GPL), pow Scale (GPL), prob Scale (GPL), probit Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

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

Figure 2-75
Example: Specifying a missing value aesthetic for a scale

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

**Applies To**

asn Scale (GPL), atanh Scale (GPL), cat Scale (GPL), linear Scale (GPL), log Scale (GPL), logit Scale (GPL), pow Scale (GPL), prob Scale (GPL), probit Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

### 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**

Figure 2-76

*Example: Specifying a 99% confidence interval*

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

**Applies To**

region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.confi.smooth Function (GPL), summary.percentile Function (GPL)

### 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**

Figure 2-77

*Example: Specifying a different base for a logarithmic scale*

SCALE: log(dim(2), base(2))
Applies To
log Scale (GPL), safeLog Scale (GPL)

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
Figure 2-78
Example: Using a variable across clusters as the percentage base

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

Applies To
region.conf.percent.count Function (GPL), region.conf.proportion.count Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL)

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

Figure 2-79
Example: Specifying the total count as the percentage base

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

Figure 2-80
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())))

Figure 2-81
Example: Specifying the total count across all panels as the percentage base

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

Applies To

region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL),
summary.percent Function (GPL), summary.percent.count Function (GPL),
summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL),
summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL),
summary.proportion Function (GPL), summary.proportion.count Function (GPL),
summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL),
summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL)

base.coordinate Function

Syntax

base.coordinate(dim(<numeric>))

<numeric>. A numeric value indicating the dimension to which the scale applies. For more
information, see the topic dim Function on p. 109.

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

Figure 2-82
Example: Making each stack equal 100%

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

Figure 2-83
Example: Making each cluster 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))
```

Applies To

region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL),
summary.percent Function (GPL), summary.percent.count Function (GPL),
summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL),
summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL),
summary.proportion Function (GPL), summary.proportion.count Function (GPL),
summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL)

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

Figure 2-84
Example: Scaling a graph by 50%

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

Figure 2-85
Example: Defining a particular graph

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

Valid Functions

origin Function (For GPL Graphs), scale Function (For GPL Graphs)

Applies To

GRAPH Statement (GPL)
**begin Function (For GPL Pages)**

**Syntax**

\[ \text{begin(<function>)} \]

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

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

**Examples**

Figure 2-86

*Example: Scaling a visualization*

\[ \text{PAGE: begin(scale(50\%,50\%))} \]

**Valid Functions**

*scale Function (For GPL Pages)*

**Applies To**

PAGE Statement (GPL)

---

**beta Function**

**Syntax**

\[ \text{beta(<shape>, <shape>)} \]

*<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**

Figure 2-87

*Example: Specifying a beta distribution for the probability scale*

\[ \text{SCALE: prob(dim(2), beta(2, 5))} \]

**Applies To**

prob Scale (GPL)
**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 p. 4 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. For more information, see the topic *dim Function* on p. 109.

*<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**

*Figure 2-88 Example: Creating a 1-D dot plot*

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

**Statistic Functions**

See *GPL Functions* on p. 59.

**Valid Functions**

*binCount Function (GPL), binStart Function (GPL), binWidth Function (GPL)*
**Binning Functions**

bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), density.beta Function (GPL), density.chiSquare Function (GPL), density.exponential Function (GPL), density.f Function (GPL), density.gamma Function (GPL), density.logistic Function (GPL), density.normal Function (GPL), density.poisson Function (GPL), density.studentizedRange Function (GPL), density.t Function (GPL), density.uniform Function (GPL), density.weibull Function (GPL), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.spread.sd Function (GPL), region.spread.se Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL)

**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 p. 4 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. For more information, see the topic dim Function on p. 109.

**<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**

Figure 2-89

Example: Binned scatterplot

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

Figure 2-90

Example: Binned polygon

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

**Statistic Functions**

See GPL Functions on p. 59.

**Valid Functions**

binCount Function (GPL), binStart Function (GPL), binWidth Function (GPL)

**Binning Functions**

bin.dot Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), density.beta Function (GPL), density.chiSquare Function (GPL), density.exponential Function (GPL), density.f Function (GPL), density.gamma Function (GPL), density.logistic Function (GPL), density.normal Function (GPL), density.poisson Function (GPL), density.studentizedRange Function (GPL), density.t Function (GPL), density.uniform Function (GPL), density.weibull Function (GPL), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay
Chapter 2

Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.spread.sd Function (GPL), region.spread.se Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL)

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 p. 4 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

Figure 2-91
Example: Creating a box plot

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

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.rect Function (GPL), density.beta Function (GPL), density.chiSquare Function (GPL), density.exponential Function (GPL), density.f Function (GPL), density.gamma Function (GPL), density.logistic Function (GPL), density.normal Function (GPL), density.poisson Function (GPL), density.studentizedRange Function (GPL), density.t Function (GPL), density.uniform Function (GPL), density.weibull Function (GPL), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.spread.sd Function (GPL), region.spread.se Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL)

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 p. 4 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. For more information, see the topic dim Function on p. 109.

**<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**

**Figure 2-92**

Example: Histogram binning

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

**Figure 2-93**

Example: Histogram binning with specified bin sizes

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

**Figure 2-94**

Example: Binned scatterplot

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

**Statistic Functions**

See GPL Functions on p. 59.

**Valid Functions**

binCount Function (GPL), binStart Function (GPL), binWidth Function (GPL)
Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), density.beta Function (GPL), density.chiSquare Function (GPL), density.exponential Function (GPL), density.f Function (GPL), density.gamma Function (GPL), density.logistic Function (GPL), density.normal Function (GPL), density.poisson Function (GPL), density.studentizedRange Function (GPL), density.t Function (GPL), density.uniform Function (GPL), density.weibull Function (GPL), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.spread.sd Function (GPL), region.spread.se Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), summary.variance Function (GPL)

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.
Chapter 2

Examples

Figure 2-95
Example: Defining a specific number of bins

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

Figure 2-96
Example: Defining a specific number of bins for multiple binned dimensions

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

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.rect Function (GPL)

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

Figure 2-97
Example: Specifying the first bin on a continuous scale

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

Figure 2-98
Example: Specifying the first bin on a date scale

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

Figure 2-99
Example: Specifying the first bin for multiple binned dimensions

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

bin.dot Function (GPL), bin.hex Function (GPL), bin.rect Function (GPL)

**binWidth Function**

**Syntax**

\[ \text{binWidth}(\text{<numeric> ...}) \]

\textbf{<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 \textit{d} or \textit{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, \text{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, \text{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**

Figure 2-100

Example: Defining a specific bin width

\[
\text{ELEMENT: interval(position(summary.count(bin.rect(salary, binWidth(1000)))))}
\]

Figure 2-101

Example: Defining a specific bin width for multiple binned dimensions

\[
\text{ELEMENT: interval(position(bin.rect(salbegin*salary, dim(1,2), binWidth(1000,2000))))}
\]

Figure 2-102

Example: Defining a specific bin width for date data

\[
\text{ELEMENT: interval(position(summary.count(bin.rect(date, binWidth(30d)))))}
\]

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.rect Function (GPL)

**chiSquare Function**

**Syntax**

\[ \text{chiSquare}(\text{<degrees of freedom>}) \]

\textbf{<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**

\[
\text{SCALE: } \text{prob(dim}(2), \text{chiSquare}(5))
\]

** Applies To**

prob Scale (GPL)

**closed Function**

**Syntax**

\[
\text{closed()}
\]

**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**

Figure 2-103

*Example: Creating a closed line*

\[
\text{ELEMENT: line(position(x*y), closed())}
\]

** Applies To**

area Element (GPL), edge Element (GPL), line Element (GPL), path Element (GPL)

**cluster Function**

**Syntax**

\[
\text{cluster(<integer> ...)}
\]

\(<\text{integer}>\). 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

Figure 2-104
Example: Clustering on the first dimension in a 2-D coordinate system

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

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

SOURCE: \texttt{s = userSource(id("Employeedata"))}
DATA: \texttt{jobcat = col(source(s), name("jobcat"), unit.category())}
DATA: \texttt{gender = col(source(s), name("gender"), unit.category())}
DATA: \texttt{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(1), label("Gender"))
ELEMENT: interval(position(summary.mean(jobcat*salary*gender)), color(jobcat))

Figure 2-105
Clustered bar chart

SOURCE: \texttt{s = userSource(id("Employeedata"))}
DATA: \texttt{jobcat = col(source(s), name("jobcat"), unit.category())}
DATA: \texttt{gender = col(source(s), name("gender"), unit.category())}
DATA: \texttt{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*gender)))
Figure 2-106  
Faceted bar chart

**Figure 2-107**  
Example: Clustering on the first dimension in a 3-D coordinate system

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

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.

**Figure 2-108**  
Example: Clustering on the second dimension in a 3-D coordinate system

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

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 (GPL)

**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

Figure 2-109
Example: Specifying a continuous variable from a data source

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

Figure 2-110
Example: Specifying a categorical variable from a data source

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

Figure 2-111
Example: Specifying a date variable from a data source

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

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 PASW 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. For more information, see the topic format Function on p. 117.

Valid Functions

in Function (GPL), format Function (GPL), notIn Function (GPL)

Applies To

DATA Statement (GPL)

collapse Function

Syntax

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

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**

Figure 2-112

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

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

**Applies To**

TRANS Statement (GPL)

**color Function (For GPL Graphic Elements)**

*Note:* If you are modifying the color for a guide, refer to color Function (For GPL Guides) on p. 83.

**Syntax**

```plaintext
color(<algebra>)

or

color(color.<color constant>)

or

color(color."color value")

or

color(<statistic function>)
```

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. For more information, see the topic Color Constants in Appendix A on p. 344. 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**

**Figure 2-113**
Example: Specifying a color value with a constant

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

**Figure 2-114**
Example: Specifying a color value with RGB color components

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

**Figure 2-115**
Example: Specifying a color value for the bar border

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

**Figure 2-116**
Example: Using the values of a variable to control color

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

**Figure 2-117**
Example: Using a statistical function to control color

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

**Statistic Functions**

See GPL Functions on p. 59.

**Applies To**

area Element (GPL), edge Element (GPL), interval Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL), line Element (GPL)

**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 p. 82.
Chapter 2

Syntax

color(color.<color constant>)

or

color(color."color value")

<color constant>. A constant indicating a specific color, such as red. For more information, see the topic Color Constants in Appendix A on p. 344.

“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

Figure 2-118
Example: Specifying an axis color

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

Applies To

axis Guide Type (GPL), form.line Guide Type (GPL)

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 p. 85.

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**

Figure 2-119  
*Example: Using the values of a variable to control color brightness*

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

Figure 2-120  
*Example: Using a statistical function to control color brightness*

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

**Statistic Functions**

See GPL Functions on p. 59.

**Applies To**

area Element (GPL), edge Element (GPL), interval Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL), line Element (GPL)

**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 p. 84.

**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**

Figure 2-121  
*Example: Specifying a reference line brightness*

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

form.line Guide Type (GPL)

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 p. 87.

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

Figure 2-122
Example: Using the values of a variable to control color hue

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

Figure 2-123
Example: Using a statistical function to control color hue

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

Statistic Functions

See GPL Functions on p. 59.
Applies To
area Element (GPL), edge Element (GPL), interval Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL), line Element (GPL)

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 p. 86.

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
Figure 2-124
Example: Specifying a reference line hue
GUIDE: form.line(position(*,2000), color.hue(color.hue."0.5"))

Applies To
form.line Guide Type (GPL)

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 p. 88.

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**

Figure 2-125
Example: Using the values of a variable to control color saturation

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

Figure 2-126
Example: Using a statistical function to control color

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

**Statistic Functions**

See GPL Functions on p. 59.

**Applies To**

area Element (GPL), edge Element (GPL), interval Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL), line Element (GPL)

**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 p. 87.

**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**

Figure 2-127
Example: Specifying a reference line saturation

GUIDE: form.line(position(*,2000), color.saturation(color.saturation."0.5"))
**Applies To**

form.line Guide Type (GPL)

### csvSource Function

**Syntax**

```plaintext
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**

**Figure 2-128  Example: Reading a CSV file**

```plaintext
SOURCE: mydata = csvSource(file("/Data/Employee data.csv"))
```

**Valid Functions**

- missing.listwise Function (GPL), missing.pairwise Function (GPL), weight Function (GPL)

**Applies To**

- SOURCE Statement (GPL), savSource Function (GPL), sqlSource Function (GPL)

### dataMaximum Function

**Syntax**

```plaintext
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

Figure 2-129
Example: Specifying a maximum on the 2nd dimension (y axis)

SCALE: linear(dim(2), dataMaximum())

Applies To
linear Scale (GPL), log Scale (GPL), pow Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

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

Figure 2-130
Example: Specifying a minimum on the 2nd dimension (y axis)

SCALE: linear(dim(2), dataMinimum())

Applies To
linear Scale (GPL), log Scale (GPL), pow Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

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**

Figure 2-131  
*Example: Specifying the distance between major ticks*

```
GUIDE: axis(dim(1), delta(1000))
```

**Applies To**

axis Guide Type (GPL)

---

**density.beta Function**

**Syntax**

```
density.beta(<algebra>, shape1(<numeric>), shape2(<numeric>))
```

*or*

```
density.beta(<binning function>, shape1(<numeric>), shape2(<numeric>))
```

<algebra>. Graph algebra, such as x*y. Refer to *Brief Overview of GPL Algebra* on p. 4 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**.

**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**

Figure 2-132  
*Example: Adding a beta distribution curve*

```
ELEMENT: line(position(density.beta(x, shape1(2), shape2(5))))
```

---

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation
density.chiSquare Function

Syntax

density.chiSquare(<algebra>, degreesOfFreedom(<integer>))

or

density.chiSquare(<binning function>, degreesOfFreedom(<integer>))

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 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.

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**

**Figure 2-133**  
*Example: Adding a chi-square distribution curve*

\[
\text{ELEMENT: line(position(density.chiSquare(x, degreesoffreedom(5))))}
\]

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conf.i.count Function (GPL), region.conf.i.mean Function (GPL), region.conf.i.percent.count Function (GPL), region.conf.i.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)
**density.exponential Function**

**Syntax**

density.exponential(<algebra>, rate(<numeric>))

or

density.exponential(<binning function>, rate(<numeric>))

*<algebra>*. Graph algebra, such as \(x*y\). Refer to Brief Overview of GPL Algebra on p. 4 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.

**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**

Figure 2-134

Example: Adding a chi-square distribution curve

ELEMENT: line(position(density.exponential(x, rate(1.5))))

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL),
density.f Function

**Syntax**

```plaintext
density.f(<algebra>, degreesOfFreedom1(<integer>), degreesOfFreedom2(<integer>))
```

or

```plaintext
density.f(<binning function>, degreesOfFreedom1(<integer>), degreesOfFreedom2(<integer>))
```

**<algebra>**. Graph algebra, such as \(x*y\). Refer to *Brief Overview of GPL Algebra* on p. 4 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.

**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**

**Figure 2-135**

*Example: Adding an F distribution curve*

```plaintext
ELEMENT: line(position(density.f(x, degreesOfFreedom1(5), degreesOfFreedom2(2))))
```
Chapter 2

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confidence Function (GPL), region.confidence.mean Function (GPL), region.confidence.percent.count Function (GPL), region.confidence.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

density.gamma Function

**Syntax**

density.gamma(<algebra>, rate(<numeric>))

or

density.gamma(<binning function>, rate(<numeric>))

<algebra>. Graph algebra, such as \(x \times y\). Refer to Brief Overview of GPL Algebra on p. 4 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.
<numeric>. \texttt{rate} defines the parameter for the distribution. This takes a positive numeric value and is \texttt{required}.

\textbf{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.

\textbf{Examples}

Figure 2-136
\textit{Example: Adding a gamma distribution curve}

\texttt{ELEMENT: line(position(density.gamma(x, rate(2.5))))}

\textbf{Binning Functions}

\texttt{bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)}

\textbf{Applies To}

\texttt{bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.mean Function (GPL), summary.max Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se
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 \( x \times y \). Refer to Brief Overview of GPL Algebra on p. 4 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.

**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

Figure 2-137
Example: Adding the default kernel distribution

```text
ELEMENT: line(position(density.kernel.epanechnikov(x)))
```

Figure 2-138
Example: Adding a kernel distribution using a fixed window

```text
ELEMENT: line(position(density.kernel.epanechnikov(x, fixedWindow(0.05))))
```

Figure 2-139
Example: Adding a kernel distribution using k nearest neighbors

```text
ELEMENT: line(position(density.kernel.epanechnikov(x, nearestNeighbor(100))))
```

Figure 2-140
Example: Creating a 3-D graph showing kernel densities

```text
COORD: rect(dim(1,2,3))
ELEMENT: interval(position(density.kernel.epanechnikov.joint(x*y)))
```

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 (GPL), segments Function (GPL)

Applies To

- bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL),
density.logistic Function

Syntax

density.logistic(<algebra>, location(<numeric>), scaleDensity(<numeric>))

or

density.logistic(<binning function>, location(<numeric>), scaleDensity(<numeric>))

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 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.

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

Figure 2-141
Example: Adding a logistic distribution curve

ELEMENT: line(position(density.logistic(x, location(5), scaleDensity(2))))
Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delanay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confid.count Function (GPL), region.confid.mean Function (GPL), region.confid.percent.count Function (GPL), region.confid.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.confid.count Function (GPL), summary.confid.cumulative Function (GPL), summary.confid.percent.count Function (GPL), summary.confid.percent.cumulative Function (GPL), summary.confid.percent.sum Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.percent.count Function (GPL), summary.percent.percent.cumulative Function (GPL), summary.percent.percent.sum Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

density.normal Function

Syntax

density.normal(<algebra>, mean(<numeric>), standardDeviation(<numeric>))

or
density.normal(<binning function>, mean(<numeric>), standardDeviation(<numeric>))

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 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.

Description

Calculates the probability density of the normal distribution. This is often used to add a distribution curve.

Examples

Figure 2-142
Example: Adding a normal curve to a histogram

ELEMENT: interval(position(summary.count(bin.rect(x))))
ELEMENT: line(position(density.normal(x)))

Figure 2-143
Example: Creating a normal curve with specific parameters

ELEMENT: line(position(density.normal(x, mean(50000), standardDeviation(15000))))

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL)
density.poisson Function

Syntax

density.poisson(<algebra>, rate(<numeric>))

or

density.poisson(<binning function>, rate(<numeric>))

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional.

<numerical>. 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.

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

Figure 2-144
Example: Adding a Poisson distribution curve

ELEMENT: line(position(density.poisson(x, rate(5.5))))

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements)
density.studentizedRange Function

Syntax

density.studentizedRange(<algebra>, degreesOfFreedom(<integer>), k(<numeric>))

or

density.studentizedRange(<binning function>, degreesOfFreedom(<integer>), k(<numeric>))

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 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.

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

Figure 2-145
Example: Adding a Studentized range distribution curve

\[ \text{ELEMENT: line(position(density.studentizedRange(x, degreesOfFreedom(5), k(2.5))))} \]

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confid.count Function (GPL), region.confid.mean Function (GPL), region.confid.percent.count Function (GPL), region.confid.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.sd Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.sum.Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

density.t Function

Syntax

density.t(<algebra>, degreesOfFreedom(<integer>))

or
density.t(<binning function>, degreesOfFreedom(<integer>))

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 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.

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

Figure 2-146
Example: Adding a Student’s t distribution curve

ELEMENT: line(position(density.t(x, degreesOfFreedom(5))))

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.sc Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile
Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

density.uniform Function

Syntax

density.uniform(<algebra>, min(<numeric>), max(<numeric>))

or
density.uniform(<binning function>, min(<numeric>), max(<numeric>))

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 for an introduction to graph algebra.

<binning function>. A binning function. The binning function is optional.

<numerics>. 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.

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

Figure 2-147
Example: Adding a uniform distribution curve

ELEMENT: line(position(density.uniform(x)))

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function
density.weibull Function

Syntax

density.weibull(<algebra>, rate(<numeric>), scaleDensity(<numeric>))

or

density.weibull(<binning function>, rate(<numeric>), scaleDensity(<numeric>))

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 for an introduction to graph algebra. The algebra is optional.

<binning function>. A binning function. The binning function is optional. I

<integer>. rate and scaleDensity define the parameters for the distribution. These take numeric values and are required.

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

Figure 2-148
Example: Adding a Weibull distribution curve

```
ELEMENT: line(position(density.logistic(x, location(5), scaleDensity(2))))
```

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conf.count Function (GPL), region.conf.mean Function (GPL), region.conf.percent.count Function (GPL), region.conf.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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:

\[ a \times b \times c \times d \]

The variables in the algebra correspond to the following dimensions:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>dim(1)</td>
</tr>
<tr>
<td>b</td>
<td>dim(2)</td>
</tr>
<tr>
<td>c</td>
<td>dim(3)</td>
</tr>
<tr>
<td>d</td>
<td>dim(4)</td>
</tr>
</tbody>
</table>

Blended variables cannot be separated. The blend of the two variables corresponds to one dimension. Consider the following:

\[ a \times (b + c) \times d \]

The variables in the algebra correspond to the following dimensions:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>dim(1)</td>
</tr>
<tr>
<td>b+c</td>
<td>dim(2)</td>
</tr>
<tr>
<td>d</td>
<td>dim(3)</td>
</tr>
</tbody>
</table>

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, \( \text{dim}(1) \)), and the next variable gets the primary dimension number followed by a dot and a 1 (for example, \( \text{dim}(1.1) \)). Consider the following:

\[ a \times (b/c) \times d \]

The variables in the algebra correspond to the following dimensions:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>dim(1)</td>
</tr>
<tr>
<td>c</td>
<td>dim(2)</td>
</tr>
<tr>
<td>b</td>
<td>dim(2.1)</td>
</tr>
<tr>
<td>d</td>
<td>dim(3)</td>
</tr>
</tbody>
</table>
Examples

Figure 2-149
Example: Specifying a two-dimensional, rectangular coordinate system

COORD: rect(dim(1,2))

Figure 2-150
Example: Specifying an axis label for the second dimension

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

Applies To
parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), asn Scale (GPL), atanh Scale (GPL), cat Scale (GPL), linear Scale (GPL), log Scale (GPL), logit Scale (GPL), pow Scale (GPL), prob Scale (GPL), probit Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL), axis Guide Type (GPL), base.coordinate Function (GPL), bin.dot Function (GPL), bin.hex Function (GPL), bin.rect Function (GPL), reflect Function (GPL)

end Function

Syntax

end()

Description

Specifies the end of the GPL block that defines a particular graph or page.

Examples

Figure 2-151
Example: Defining a particular graph

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

Figure 2-152
Example: Defining a page

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()

Applies To

GRAPH Statement (GPL), PAGE Statement (GPL)
Chapter 2

**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. For more information, see the topic eval Operators and Functions on p. 112. 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**

**Figure 2-153**

Example: Plotting the difference between two variables

TRANS: saldiff = eval(salary-salbegin)
ELEMENT: point(position(summary.mean(jobcat*saldiff)))

**Figure 2-154**

Example: Creating a graph from an equation

DATA: x = iter(-100,100,1)
TRANS: y = eval(x**2)
ELEMENT: line(position(x*y))

**Figure 2-155**

Example: Plotting percent greater than a value

TRANS: salGreaterThan = eval(salary>50000)
ELEMENT: interval(position(summary.percentTrue(jobcat*salGreaterThan)))

**Applies To**

TRANS Statement (GPL), collapse Function (GPL)

**Eval Operators and Functions**

Following are the operators and functions that you can use with the eval function. See eval Function on p. 112 for information about the eval function.

**Operators**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition or string concatenation</td>
<td>Using + with numbers adds the numbers. Using it with strings concatenates the strings.</td>
</tr>
</tbody>
</table>
### Operator Meaning Notes

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Subtraction</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td></td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td></td>
</tr>
<tr>
<td>()</td>
<td>Grouping</td>
<td>Grouped expressions are calculated before other expressions.</td>
</tr>
<tr>
<td>**</td>
<td>Exponentiation</td>
<td></td>
</tr>
<tr>
<td>==</td>
<td>Equal</td>
<td></td>
</tr>
<tr>
<td>!=</td>
<td>Not equal</td>
<td></td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
<td></td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Logical AND</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| ? :      | Conditional | These operators are shorthand for *then-else* when evaluating a Boolean operand. For example, 
\( x > 15 ? \text{"High" : "Low"} \) returns “High” if \( x > 15 \). Otherwise, the expression returns “Low”. |

### Mathematical Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Result</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs(n)</td>
<td>The absolute value of ( n )</td>
<td></td>
</tr>
<tr>
<td>acos(n)</td>
<td>The inverse cosine (arccosine) of ( n )</td>
<td></td>
</tr>
<tr>
<td>asin(n)</td>
<td>The inverse sine (arcsine) of ( n )</td>
<td></td>
</tr>
<tr>
<td>atan(n)</td>
<td>The inverse tangent (arctangent) of ( n )</td>
<td></td>
</tr>
<tr>
<td>atanh(n)</td>
<td>The hyperbolic inverse tangent (hyperbolic arctangent) of ( n )</td>
<td></td>
</tr>
<tr>
<td>ceil(n)</td>
<td>The smallest integer that is greater than ( n )</td>
<td>Round up</td>
</tr>
<tr>
<td>cos(n)</td>
<td>The cosine of ( n )</td>
<td></td>
</tr>
<tr>
<td>cosh(n)</td>
<td>The hyperbolic cosine of ( n )</td>
<td></td>
</tr>
<tr>
<td>exp(n)</td>
<td>( e ) raised to the power ( n ), where ( e ) is the base of the natural logarithms</td>
<td></td>
</tr>
<tr>
<td>floor(n)</td>
<td>The largest integer that is less than ( n )</td>
<td>Round down</td>
</tr>
<tr>
<td>gamma(n)</td>
<td>The complete Gamma function of ( n )</td>
<td></td>
</tr>
<tr>
<td>int(n)</td>
<td>The value of ( n ) truncated to an integer (toward 0)</td>
<td></td>
</tr>
<tr>
<td>lgamma(n)</td>
<td>The logarithm of the complete Gamma function of ( n )</td>
<td></td>
</tr>
<tr>
<td>log(n)</td>
<td>The natural (base-e) logarithm of ( n )</td>
<td></td>
</tr>
</tbody>
</table>
### Chapter 2

<table>
<thead>
<tr>
<th>Function</th>
<th>Result</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\log_2(n)$</td>
<td>The base-2 logarithm of $n$</td>
<td></td>
</tr>
<tr>
<td>$\log_{10}(n)$</td>
<td>The base-10 logarithm of $n$</td>
<td></td>
</tr>
<tr>
<td>$\text{mod}(n, \text{modulus})$</td>
<td>The remainder when $n$ is divided by $\text{modulus}$</td>
<td></td>
</tr>
<tr>
<td>$\text{pow}(n, \text{power})$</td>
<td>The value of $n$ raised to the power of $\text{power}$</td>
<td></td>
</tr>
<tr>
<td>$\text{round}(n)$</td>
<td>The integer that results from rounding the absolute value of $n$ and then reattaching the sign. Numbers ending in 0.5 exactly are rounded away from 0. For example, $\text{round}(-4.5)$ rounds to -5.</td>
<td></td>
</tr>
<tr>
<td>$\sin(n)$</td>
<td>The sine of $n$</td>
<td></td>
</tr>
<tr>
<td>$\sinh(n)$</td>
<td>The hyperbolic sine of $n$</td>
<td></td>
</tr>
<tr>
<td>$\sqrt{n}$</td>
<td>The positive square root of $n$</td>
<td></td>
</tr>
<tr>
<td>$\tan(n)$</td>
<td>The tangent of $n$</td>
<td></td>
</tr>
<tr>
<td>$\tanh(n)$</td>
<td>The hyperbolic tangent of $n$</td>
<td></td>
</tr>
</tbody>
</table>

### String Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Result</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{concatenate}(\text{string1, string2})$</td>
<td>A string that is the concatenation of $\text{string1}$ and $\text{string2}$</td>
<td></td>
</tr>
<tr>
<td>$\text{datetostring}(\text{date})$</td>
<td>The string that results when $\text{date}$ is converted to a string</td>
<td></td>
</tr>
<tr>
<td>$\text{indexof}(\text{haystack, needle}, [\text{divisor}])$</td>
<td>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, $\text{indexof}(x, \text{&quot;abcd&quot;})$ will return the value of the starting position of the complete string &quot;abcd&quot; in the string variable x; $\text{indexof}(x, \text{&quot;abcd&quot;}, 1)$ will return the value of the position of the first occurrence of any of the values in the string; and $\text{indexof}(x, \text{&quot;abcd&quot;}, 2)$ 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.</td>
<td></td>
</tr>
<tr>
<td>$\text{length}(\text{string})$</td>
<td>A number indicating the length of $\text{string}$</td>
<td></td>
</tr>
</tbody>
</table>
### Function Result Notes

<table>
<thead>
<tr>
<th>Function</th>
<th>Result</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>lowercase(string)</td>
<td>string with uppercase letters changed to lowercase and other characters unchanged</td>
<td></td>
</tr>
<tr>
<td>ltrim(string[, char])</td>
<td>string with any leading instances of char removed. If char is not specified, leading blanks are removed. Char must resolve to a single character.</td>
<td></td>
</tr>
<tr>
<td>midstring(string, start, end)</td>
<td>The substring beginning at position start of string and ending at end</td>
<td></td>
</tr>
<tr>
<td>numbertostring(n)</td>
<td>The string that results when n is converted to a string</td>
<td></td>
</tr>
<tr>
<td>replace(target, old, new)</td>
<td>In target, instances of old are replaced with new. All arguments are strings.</td>
<td></td>
</tr>
<tr>
<td>rtrim(string[, char])</td>
<td>string with any trailing instances of char removed. If char is not specified, trailing blanks are removed. Char must resolve to a single character.</td>
<td></td>
</tr>
<tr>
<td>stringtodate(string)</td>
<td>The value of the string expression string as a date</td>
<td></td>
</tr>
<tr>
<td>stringtonumber(string)</td>
<td>The value of the string expression string as a number</td>
<td></td>
</tr>
<tr>
<td>substring(string, start, length)</td>
<td>The substring beginning at position start of string and running for length length</td>
<td></td>
</tr>
<tr>
<td>trim(string)</td>
<td>string with any leading and trailing blanks removed</td>
<td></td>
</tr>
<tr>
<td>uppercase(string)</td>
<td>string with lowercase letters changed to uppercase and other characters unchanged</td>
<td></td>
</tr>
</tbody>
</table>

### Date and Time Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Result</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>date()</td>
<td>The current date</td>
<td></td>
</tr>
<tr>
<td>time()</td>
<td>The current time</td>
<td></td>
</tr>
</tbody>
</table>

### Constants

<table>
<thead>
<tr>
<th>Constant</th>
<th>Meaning</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>pi</td>
<td>pi</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>Euler’s number or the base of the natural logarithm</td>
<td></td>
</tr>
</tbody>
</table>
**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**

Figure 2-156

*Example: Exclude a category*

```
SCALE: cat(dim(1), exclude("No Response"))
```

Figure 2-157

*Example: Excluding multiple categories*

```
SCALE: cat(dim(1), exclude("No Response", "Didn't Ask"))
```

** Applies To**

cat Scale (GPL)

**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**

Figure 2-158

*Example: Specifying a different power exponent*

```
SCALE: power(dim(2), exponent(3))
```
**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**

Figure 2-159
Example: Specifying an exponential distribution for the probability scale

SCALE: prob(dim(2), exponential(1.5))

**Applies To**

prob Scale (GPL)

---

**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 (GPL)

---

**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 2-2**  
**Abbreviations for date formats**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Month</td>
</tr>
<tr>
<td>d</td>
<td>Day</td>
</tr>
<tr>
<td>y</td>
<td>Year</td>
</tr>
<tr>
<td>m</td>
<td>Minute</td>
</tr>
<tr>
<td>s</td>
<td>Second</td>
</tr>
</tbody>
</table>

*Note:* In PASW Statistics, dates from a `userSource` are passed to GPL as numeric values. Therefore, this function does not have any affect on a `userSource`.

**Examples**

**Figure 2-160**  
*Example: Indicating a date format of the form 1/31/2006*

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

**Applies To**

`col Function (GPL)`

**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**

**Figure 2-161**  
*Example: Displaying dates on an axis*

GUIDE: axis(dim(1), format.date())

**Applies To**

`axis Guide Type (GPL)`
**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**

Figure 2-162
*Example: Displaying dates and times on an axis*

GUIDE: axis(dim(1), format.dateTime())

**Applies To**

axis Guide Type (GPL)

**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**

Figure 2-163
*Example: Displaying times on an axis*

GUIDE: axis(dim(1), format.time())

**Applies To**

axis Guide Type (GPL)

**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**

Figure 2-164
*Example: Creating a directed acyclic graph*

```javascript
ELEMENT: edge(position(layout.dag(node(id), from(fromVar), to(toVar))))
```

**Applies To**

layout.circle Function (GPL), layout.dag Function (GPL), layout.data Function (GPL), layout.grid Function (GPL), layout.network Function (GPL), layout.random Function (GPL), layout.tree Function (GPL)

---

**gamma Function**

**Syntax**

```javascript
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**

Figure 2-165
*Example: Specifying a gamma distribution for the probability scale*

```javascript
SCALE: prob(dim(2), gamma(2.5))
```

**Applies To**

prob Scale (GPL)

---

**gap Function**

**Syntax**

```javascript
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**

Figure 2-166
*Example: Forcing adjacent axes to touch*

GUIDE: axis(dim(3), gap(0px))

**Applies To**

axis Guide Type (GPL)

---

**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**

Figure 2-167
*Example: Displaying grid lines*

GUIDE: axis(dim(2), gridlines())

**Applies To**

axis Guide Type (GPL)

---

**in Function**

**Syntax**

\[
\text{in(min(<value>), max(<value>))}
\]

**<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 \text{min} and \text{max} parameters. This function is valid only for continuous variables.
Chapter 2

Examples

Figure 2-168  
Example: Including only a subset of continuous values

DATA: gender = col(source(mydata), name("salary"),
in(min(0), max(50000)))

Applies To

col Function (GPL)

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

Figure 2-169  
Example: Include a category

SCALE: cat(dim(1), include("No Response"))

Figure 2-170  
Example: Including multiple categories

SCALE: cat(dim(1), include("No Response", "Didn't Ask"))

Figure 2-171  
Example: Include a value

SCALE: linear(dim(2), include(0))

Figure 2-172  
Example: Including multiple values

SCALE: linear(dim(2), include(0, 100))

Applies To

cat Scale (GPL), linear Scale (GPL), log Scale (GPL), pow Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)
**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**

Figure 2-173

*Example: Create an index variable and label by the variable*

TRANS: casenum = index()
ELEMENT: point(position(x*y), label(casenum))

**Applies To**

TRANS Statement (GPL), collapse Function (GPL)

---

**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**

Figure 2-174

*Example: Creating a graph from an equation*

DATA: x = iter(-100,100,1)
TRANS: y = eval(x**2)
ELEMENT: line(position(x*y))

**Applies To**

DATA Statement (GPL)
Chapter 2

**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**

Figure 2-175

Example: Specifying an interpolation line

ELEMENT: line(position(smooth.step(educ*salary)), jump())

**Applies To**

area Element (GPL), line Element (GPL)

**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 p. 125.

**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

Figure 2-176  
Example: Labeling by another variable

ELEMENT: point(position(salbegin*salary), label(gender))

Figure 2-177  
Example: Labeling by the result of a statistic

ELEMENT: point(position(summary.count(jobcat)), label(summary.count()))

Figure 2-178  
Example: Labeling by the result of a statistic

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

Figure 2-179  
Example: Creating a multi-line label

ELEMENT: interval(position(summary.mean(jobcat*salary)), label("Count:"), label(summary.count()))

Statistic Functions

See GPL Functions on p. 59.

Valid Functions

showAll Function (GPL)

Applies To

area Element (GPL), edge Element (GPL), interval Element (GPL), line Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL)

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 p. 124.

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.
Chapter 2

**Examples**

Figure 2-180  
*Example: Specifying an axis title*

```
GUIDE: axis(dim(1), label('Job Category'))
```

Figure 2-181  
*Example: Specifying a legend title*

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

Figure 2-182  
*Example: Specifying a graph title*

```
GUIDE: text.title(label('Sales By Region'))
```

**Applies To**

axis Guide Type (GPL), form.line Guide Type (GPL), legend Guide Type (GPL), text.footnote Guide Type (GPL), text.subfootnote Guide Type (GPL), text.subsubfootnote Guide Type (GPL), text.subtitle Guide Type (GPL), text.subsubtitle Guide Type (GPL), text.title Guide Type (GPL)

**layout.circle Function**

**Syntax**

```
layout.circle(<function>)
```

or

```
layout.circle(<algebra>, <function>)
```

**<algebra>**. Graph algebra, such as \(x \times y\). Refer to *Brief Overview of GPL Algebra* on p. 4 for an introduction to graph algebra. The algebra for network graphs is \(1 \times 1\) 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 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

Figure 2-183  
Example: Creating a circular network diagram

\[
\text{ELEMENT: edge(position(layout.circle(node(id), from(fromVar), to(toVar))))} \\
\text{ELEMENT: point(position(layout.circle(node(id), from(fromVar), to(toVar))), label(id))}
\]

Valid Functions

from Function (GPL), node Function (GPL), to Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect 
Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For 
GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation 
Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function 
(GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function 
(GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), 
link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function 
(GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL 
Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), 
region.confi.percent.count Function (GPL), region.confi.percent.count.cumulative Function 
(GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function 
(GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function 
(GPL), summary.count.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), 
summary.max Function (GPL), summary.mean Function (GPL), summary.median Function 
(GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative 
Function (GPL), summary.percent.sum Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), 
summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

layout.dag Function

Syntax

layout.dag(<function>))

or

layout.dag(<algebra>, <function>)
**Graph algebra.** Graph algebra, such as $x*y$. Refer to Brief Overview of GPL Algebra on p. 4 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**

**Figure 2-184**

**Example: Creating a directed acyclic graph**

```plaintext
ELEMENT: edge(position(layout.dag(node(id), from(fromVar), to(toVar))))
ELEMENT: point(position(layout.dag(node(id), from(fromVar), to(toVar))), label(id))
```

**Valid Functions**

from Function (GPL), node Function (GPL), to Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conficount Function (GPL), region.confi.mean Function (GPL), region.confipercent Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cummulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function
(GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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 p. 4 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**

Figure 2-185

*Example: Creating a network diagram*

```
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))))
```
Valid Functions
from Function (GPL), node Function (GPL), to Function (GPL)

Applies To
bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.kurtosis Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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**

Figure 2-186

*Example: Creating a grid network diagram*

```gls
element: edge(position(layout.grid(node(id), from(fromVar), to(toVar))))
element: point(position(layout.grid(node(id), from(fromVar), to(toVar))), label(id))
```

**Valid Functions**

from Function (GPL), node Function (GPL), to Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percent.sum.cumulative Function
layout.network Function

Syntax

layout.network(<function>)

or

layout.network(<algebra>, <function>)

<algebra>. Graph algebra, such as \(x \times y\). Refer to Brief Overview of GPL Algebra on p. 4 for an introduction to graph algebra. The algebra for network graphs is \(1 \times 1\) 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

Figure 2-187
Example: Creating a network diagram

ELEMENT: edge(position(layout.network(node(id), from(fromVar), to(toVar))))
ELEMENT: point(position(layout.network(node(id), from(fromVar), to(toVar))), label(id))

Valid Functions

from Function (GPL), node Function (GPL), to Function (GPL)
### Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delamay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

### layout.random Function

**Syntax**

```
layout.random(<function>)
```

*or*

```
layout.random(<algebra>, <function>)
```

**<algebra>**. Graph algebra, such as \(x \times y\). Refer to Brief Overview of GPL Algebra on p. 4 for an introduction to graph algebra. The algebra for network graphs is \(1 \times 1\) 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

Figure 2-188
Example: Creating a random network diagram

```plaintext
ELEMENT: edge(position(layout.random(node(id), from(fromVar), to(toVar))))
ELEMENT: point(position(layout.random(node(id), from(fromVar), to(toVar))), label(id))
```

Valid Functions

from Function (GPL), node Function (GPL), to Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.percent Function (GPL), region.confi.percent.count.cumulative Function (GPL), region.confi.percent.cumulative Function (GPL), region.confi.percent.sum Function (GPL), region.confi.percent.sum.cumulative Function (GPL), region.confi.proportion.count Function (GPL), region.confi.proportion Function (GPL), region.confi.proportion.count.cumulative Function (GPL), region.confi.proportion.cumulative Function (GPL), region.confi.proportion.sum Function (GPL), region.confi.proportion.sum.cumulative Function (GPL), region.range Function (GPL), region.sd Function (GPL), region.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.rank Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se
Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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 p. 4 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**

Figure 2-189

**Example: Creating a tree**

```
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))
```

**Valid Functions**

from Function (GPL), node Function (GPL), root Function (GPL), to Function (GPL)
Chapter 2

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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 p. 4 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

Figure 2-190
Example: Creating an alpha shape graph

ELEMENT: edge(position(link.alpha(x*y, radius(50))))

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function
link.complete Function

Syntax

\[
\text{link.complete}(\text{<algebra>})
\]

or

\[
\text{link.complete}(\text{<binning function>})
\]

or

\[
\text{link.complete}(\text{<statistic function>})
\]

<algebra>. Graph algebra, such as \(x*y\). Refer to Brief Overview of GPL Algebra on p. 4 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 \text{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

Figure 2-191
Example: Creating a complete graph

\[
\text{ELEMENT: edge(position(link.complete(x*y)))}
\]

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)
**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.means Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**link.delaunay Function**

**Syntax**

link.delaunay(<algebra>)

or

link.delaunay(<binning function>)

or

link.delaunay(<statistic function>)

<algebra>. Graph algebra, such as \(x\cdot y\). Refer to Brief Overview of GPL Algebra on p. 4 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**

Figure 2-192  
Example: Creating a Delaunay triangulation

ELEMENT: edge(position(link.delaunay(x*y)))

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.sc.kurtosis Function (GPL), summary.se.skewness Function (GPL),
summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**link.distance Function**

**Syntax**

\[
\text{link.distance}(<\text{algebra}>, \text{radius}(<\text{numeric}>))
\]

\[\text{or}\]

\[
\text{link.distance}(<\text{binning function}>, \text{radius}(<\text{numeric}>))
\]

\[\text{or}\]

\[
\text{link.distance}(<\text{statistic function}>, \text{radius}(<\text{numeric}>))
\]

*<algebra>*. Graph algebra, such as \(x*y\). Refer to **Brief Overview of GPL Algebra** on p. 4 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**

Figure 2-193

Example: Creating a distance graph

**ELEMENT**: edge(position(link.distance(x*y), radius(5000)))

**Statistic Functions**

See **GPL Functions** on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation
**link.gabriel Function**

**Syntax**

```r
link.gabriel(<algebra>)
```

or

```r
link.gabriel(<binning function>)
```

or

```r
link.gabriel(<statistic function>)
```

**<algebra>**. Graph algebra, such as \(x \times y\). Refer to Brief Overview of GPL Algebra on p. 4 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 \(\text{link.gabriel}\).
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

Figure 2-194
Example: Creating a Gabriel graph

\[
\text{ELEMENT: edge(position(link.gabriel(x*y)))}
\]

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.sc.kurtosis Function (GPL), summary.se.skewness Function (GPL),
**link.hull Function**

**Syntax**

\[
\text{link.hull(<algebra>)} \\
\text{or} \\
\text{link.hull(<binning function>)} \\
\text{or} \\
\text{link.hull(<statistic function>)}
\]

*<algebra>*. Graph algebra, such as \(x \cdot y\). Refer to *Brief Overview of GPL Algebra* on p. 4 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**

Figure 2-195

*Example: Creating a convex hull*

\[
\text{ELEMENT: edge(position(link.hull(x*y)))}
\]

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation
Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confidence count Function (GPL), region.confidence mean Function (GPL), region.confidence.percent.count Function (GPL), region.confidence.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.count.cumulative True Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.median Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent.count Function (GPL), summary.percent.mean Function (GPL), summary.percent.median Function (GPL), summary.percent.percent.count Cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative True Function (GPL), summary.proportion.first Function (GPL), summary.proportion.mean Function (GPL), summary.proportion.median Function (GPL), summary.proportion.percent.count True Function (GPL), summary.proportion.percent.mean Function (GPL), summary.proportion.percent.median Function (GPL), summary.proportion.percent.percent.count True Function (GPL), summary.proportion.percent.percent.count Cumulative Function (GPL), summary.proportion.percent.percent.percent.count True Function (GPL), summary.proportion.percent.percent.percent.count Cumulative Function (GPL), summary.proportion.percent.sum Function (GPL), summary.proportion.percent.sum.cumulative Function (GPL), summary.proportion.percent.sum.cumulative True Function (GPL), summary.proportion.percent.sum.true Function (GPL), summary.proportion.percent.true Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**link.influence Function**

**Syntax**

```
link.influence(<algebra>)
```

or

```
link.influence(<binning function>)
```

or

```
link.influence(<statistic function>)
```

<algebra>. Graph algebra, such as \( x \times y \). Refer to Brief Overview of GPL Algebra on p. 4 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**

Figure 2-196

Example: Creating a sphere of influence graph

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

**Statistic Functions**

See **GPL Functions** on p. 59.

**Binning Functions**

- bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

- bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL),
link.join Function

Syntax

\[
\text{link.join}(<\text{algebra}>)
\]

\[
\text{or}
\]

\[
\text{link.join}(<\text{binning function}>)
\]

\[
\text{or}
\]

\[
\text{link.join}(<\text{statistic function}>)
\]

<algebra>. Graph algebra, such as \(x \cdot y\). Refer to Brief Overview of GPL Algebra on p. 4 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

Figure 2-197
Example: Creating a bridge plot

\[
\text{ELEMENT: edge(position(link.join(x1*y1 + x2*y2)), label(a))}
\]

\[
\text{ELEMENT: point(position(x1*y1 + x2*y2), label("Before" + "After"))}
\]

This example assumes data that is in a format like the following:

Table 2-3
Example data

<table>
<thead>
<tr>
<th>#</th>
<th>x1</th>
<th>y1</th>
<th>x2</th>
<th>y2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bill</td>
<td>45</td>
<td>50</td>
<td>58</td>
<td>67</td>
</tr>
<tr>
<td>Alice</td>
<td>32</td>
<td>40</td>
<td>33</td>
<td>40</td>
</tr>
<tr>
<td>Bob</td>
<td>22</td>
<td>31</td>
<td>26</td>
<td>35</td>
</tr>
<tr>
<td>Audrey</td>
<td>55</td>
<td>59</td>
<td>52</td>
<td>64</td>
</tr>
</tbody>
</table>

Figure 2-198
Example: Drawing vectors from the origin

\[
\text{TRANS: zero = eval(0)}
\]

\[
\text{ELEMENT: edge(position(link.join(zero*zero + x*y)), shape(shape.arrow))}
\]
Statistic Functions
See GPL Functions on p. 59.

Binning Functions
bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To
bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confidence.count Function (GPL), region.confidence.mean Function (GPL), region.confidence.percent.count Function (GPL), region.confidence.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.first.count Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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**

Figure 2-199

*Example: Creating a minimal spanning tree*

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

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative
Chapter 2

Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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

Figure 2-200

Example: Creating a nearest neighbor graph

ELEMENT: edge(position(link.neighbor(x*y, neighborCount(3))))

Statistic Functions

See GPL Functions on p. 59.
**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delannay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conf.count Function (GPL), region.conf.mean Function (GPL), region.conf.percent.count Function (GPL), region.conf.percent.proportion.count Function (GPL), region.conf.percent.range Function (GPL), region.conf.percent.sd Function (GPL), region.conf.percent.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**link.relativeNeighborhood Function**

**Syntax**

```plaintext
link.relativeNeighborhood(<algebra>)
```

or

```plaintext
link.relativeNeighborhood(<binning function>)
```

or

```plaintext
link.relativeNeighborhood(<statistic function>)
```
<algebra>. Graph algebra, such as \(x \times y\). Refer to Brief Overview of GPL Algebra on p. 4 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

Figure 2-201
Example: Creating a relative neighborhood graph

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

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conficount Function (GPL), region.confimean Function (GPL), region.confipercent.count Function (GPL), region.confiproportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative
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 p. 4 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**

Figure 2-202

*Example: Creating a sequence graph*

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

**Statistic Functions**

See GPL Functions on p. 59.
Chapter 2

**Binning Functions**

- bin.dot Function (GPL)
- bin.hex Function (GPL)
- bin.quantile.letter Function (GPL)
- bin.rect Function (GPL)

**Applies To**

- bin.dot Function (GPL)
- bin.hex Function (GPL)
- bin.quantile.letter Function (GPL)
- bin.rect Function (GPL)
- color Function (For GPL Graphic Elements)
- color.brightness Function (For GPL Graphic Elements)
- color.hue Function (For GPL Graphic Elements)
- color.saturation Function (For GPL Graphic Elements)
- link.alpha Function (GPL)
- link.complete Function (GPL)
- link delaunay Function (GPL)
- link.distance Function (GPL)
- link.gabriel Function (GPL)
- link.hull Function (GPL)
- link.influence Function (GPL)
- link.join Function (GPL)
- link.mst Function (GPL)
- link.neighbor Function (GPL)
- link.relativeNeighborhood Function (GPL)
- region.confi.count Function (GPL)
- region.confi.mean Function (GPL)
- region.confi.percent.count Function (GPL)
- region.confi.proportion.count Function (GPL)
- region.spread.range Function (GPL)
- region.spread.sd Function (GPL)
- region.spread.se Function (GPL)
- size Function (For GPL Graphic Elements)
- split Function (GPL)
- summary.count Function (GPL)
- summary.count.cumulative Function (GPL)
- summary.countTrue Function (GPL)
- summary.first Function (GPL)
- summary.kurtosis Function (GPL)
- summary.last Function (GPL)
- summary.max Function (GPL)
- summary.median Function (GPL)
- summary.min Function (GPL)
- summary.mode Function (GPL)
- summary.percent Function (GPL)
- summary.percent.count Function (GPL)
- summary.percent.cumulative Function (GPL)
- summary.percent.sum Function (GPL)
- summary.percent.cumulative Function (GPL)
- summary.percent.sum.cumulative Function (GPL)
- summary.percentile Function (GPL)
- summary.percentTrue Function (GPL)
- summary.proportion Function (GPL)
- summary.proportion.count Function (GPL)
- summary.proportion.cumulative Function (GPL)
- summary.proportion.sum Function (GPL)
- summary.proportion.sum.cumulative Function (GPL)
- summary.proportionTrue Function (GPL)
- summary.range Function (GPL)
- summary.sd Function (GPL)
- summary.se Function (GPL)
- summary.se.kurtosis Function (GPL)
- summary.se.skewness Function (GPL)
- summary.sum Function (GPL)
- summary.cumulative Function (GPL)
- summary.variance Function (GPL)
- transparency Function (For GPL Graphic Elements)

**link.tsp Function**

**Syntax**

```plaintext
link.tsp(<algebra>)
```

*or*

```plaintext
link.tsp(<binning function>)
```

*or*

```plaintext
link.tsp(<statistic function>)
```
<algebra>. Graph algebra, such as \(x \times y\). Refer to Brief Overview of GPL Algebra on p. 4 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**

Figure 2-203
*Example: Drawing the solution to the travelling salesman problem*

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

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), position Function (For GPL Graphic Elements), region.conf.int.count Function (GPL), region.conf.int.mean Function (GPL), region.conf.int.percent.count Function (GPL), region.conf.int.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile
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

Figure 2-204
Example: Specifying a logistic distribution for the probability scale

SCALE: prob(dim(2), logistic(5, 2))

Applies To

prob Scale (GPL)

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**

Figure 2-205  
*Example: Mapping a category to a color*

```
SCALE: cat(aesthetic(aesthetic.color), map(\text{"IL", color.blue}))
```

Figure 2-206  
*Example: Mapping multiple categories to colors*

```
SCALE: cat(aesthetic(aesthetic.color), map(\text{"IL", color.blue}, \text{"CA", color.green}))
```

**Applies To**

cat Scale (GPL)

---

**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**

Figure 2-207  
*Example: Adding a kernel distribution*

```
ELEMENT: line(position(density.kernel.epanechnikov(x, marron())))
```

**Applies To**

density.kernel Function (GPL)

---

**max Function**

**Syntax**

```
max(<numeric>)
```

<numeric>. A numeric value indicating the maximum scale value.
**Description**

Specifies a maximum value for a scale.

**Examples**

Figure 2-208
*Example: Specifying a maximum on the 2nd dimension (y axis)*

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

**Applies To**

linear Scale (GPL), log Scale (GPL), pow Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

---

**min Function**

**Syntax**

[min(<numeric>)]

**<numeric>**. A numeric value indicating the minimum scale value.

**Description**

Specifies a minimum value for a scale.

**Examples**

Figure 2-209
*Example: Specifying a minimum on the 2nd dimension (y axis)*

SCALE: linear(dim(2), min(0))

**Applies To**

linear Scale (GPL), log Scale (GPL), pow Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

---

**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

Figure 2-210
Example: Mirroring dimensions in a population pyramid

COORD: transpose(mirror(rect(dim(1, 2)))))

Coordinate Types and Transformations

parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), project Function (GPL), reflect Function (GPL), transpose Function (GPL), wrap Function (GPL)

Applies To

COORD Statement (GPL), parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), project Function (GPL)

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

Figure 2-211
Example: Specifying gaps for missing values

ELEMENT: line(position(x*y), missing.gap())

Applies To

area Element (GPL), edge Element (GPL), line Element (GPL), path Element (GPL)

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

Figure 2-212
Example: Interpolating through missing values

ELEMENT: line(position(x*y), missing.interpolate())

Applies To
area Element (GPL), edge Element (GPL), line Element (GPL), path Element (GPL)

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

Figure 2-213
Example: Excluding missing values listwise

SOURCE: s = csvSource(file("mydata.csv"), missing.listwise())

Applies To
csvSource Function (GPL), sqlSource Function (GPL), userSource Function (GPL)

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

Figure 2-214
Example: Excluding missing values pairwise

SOURCE: s = csvSource(file("mydata.csv"), missing.pairwise())
Applies To
csvSource Function (GPL), sqlSource Function (GPL), userSource Function (GPL)

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

Figure 2-215 Example: Specifying wings for missing values

ELEMENT: line(position(x*y), missing.wings())

Applies To
area Element (GPL), edge Element (GPL), line Element (GPL), path Element (GPL)

multiple Function

Syntax

multiple(<numeric>)

<numeric>. A positive numeric value.

Description

Specifies a multiplier for statistic functions.

Examples

Figure 2-216 Example: Specifying 2 standard deviations

ELEMENT: interval(position(region.spread.sd(x*y, multiple(2))), shape(shape.ibeam))

Applies To
region.spread.sd Function (GPL), region.spread.se Function (GPL)
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

Figure 2-217
Example: Creating a linear fit line through the origin

ELEMENT: line(position(smooth.linear(salbegin\*salary, noConstant())))

Applies To

region.conf.smooth Function (GPL), smooth.cubic Function (GPL), smooth.linear Function (GPL), smooth.quadratic Function (GPL)

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

Figure 2-218
Example: Creating a directed acyclic graph

ELEMENT: edge(position(layout.dag(node(id), from(fromVar), to(toVar))))

Applies To

layout.circle Function (GPL), layout.dag Function (GPL), layout.data Function (GPL), layout.grid Function (GPL), layout.network Function (GPL), layout.random Function (GPL), layout.tree Function (GPL)
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

Figure 2-219
Example: Excluding a category from a variable

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

Applies To

col Function (GPL)

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

Figure 2-220
Example: Specifying a normal distribution for the probability scale

SCALE: prob(dim(2), normal(50000, 15000))

Applies To

prob Scale (GPL)


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

Figure 2-221
Example: Moving the y-axis to the opposite side

GUIDE: axis(dim(2), label("Count"), opposite())

Figure 2-222
Example: Adding a derived axis

GUIDE: axis(dim(2), label("Cumulative Count"))
GUIDE: axis(dim(2), label("Cumulative Percent"), opposite(), unit.percent())

Applies To

axis Guide Type (GPL)

origin Function (For GPL Graphs)

Note: If you are modifying the origin for a scale, refer to origin Function (For GPL Scales) on p. 165.

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

Figure 2-223
Example: Positioning a graph with absolute units

GRAPH: start(origin(2in, 4in))

Figure 2-224
Example: Positioning a graph with percentages

GRAPH: start(origin(10%, 100%))

Applies To

begin Function (For GPL Graphs)

**origin Function (For GPL Scales)**

*Note:* If you are modifying the origin for a graph, refer to origin Function (For GPL Graphs) on p. 164.

**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**

Figure 2-225
Example: Specifying the origin

SCALE: linear(dim(2), origin(100))

**Applies To**

linear Scale (GPL), log Scale (GPL), pow Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

**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**

Figure 2-226
*Example: Specifying a poisson distribution for the probability scale*

`SCALE: prob(dim(2), poisson(5.5))`

**Applies To**

prob Scale (GPL)

**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 p. 167.

**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 p. 4 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**

Figure 2-227
*Example: Scatterplot*

`ELEMENT: point(position(x*y))`
Figure 2.228  
Example: Bar chart of means  

\[
\text{ELEMENT: interval(position(summary.mean(x*y)))}
\]

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

area Element (GPL), edge Element (GPL), interval Element (GPL), line Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL)

**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 p. 166.

**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**

Figure 2.229  
Example: Horizontal reference line  

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

Figure 2.230  
Example: Vertical reference line  

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

**Applies To**

form.line Guide Type (GPL)
Chapter 2

**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**

Figure 2-231  
Example: Drawing straight lines

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

**Applies To**

area Element (GPL), edge Element (GPL), line Element (GPL), path Element (GPL), polygon Element (GPL)

**project Function**

**Syntax**

```
project.<projection>()
```

`<projection>`: A valid projection name.

**Description**

Transforms the coordinate system using a map projection.

**Examples**

Figure 2-232  
Example: Creating a map

```
SOURCE: maps = mapSource(file("World.smz"), layer("World"))
DATA: lon*lat = mapVariables(source(maps))
COORD: project.mercator()
ELEMENT: polygon(position(lon*lat))
```

**Valid Projection Names**

- lambert
- mercator
- transverseMercator
- winkelTripel
**proportion Function**

**Syntax**

\[
\text{proportion(<numeric>)}
\]

<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**

**Figure 2-233**
Example: Creating a loess fit line with specific smoother window

\[
\text{ELEMENT: line(position(smooth.loess(salbegin\text{*}salary, proportion(0.9))}})
\]

**Applies To**

smooth.cubic Function (GPL), smooth.linear Function (GPL), smooth.loess Function (GPL), smooth.mean Function (GPL), smooth.median Function (GPL), smooth.quadratic Function (GPL)

**reflect Function**

**Syntax**

\[
\text{reflect(dim(<numeric>), <coord>)}
\]

<numeric>. A numeric value indicating the dimension across which the graph is reflected. For more information, see the topic dim Function on p. 109.

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

**Description**

Reflects the coordinate system across the specified dimension.

**Examples**

**Figure 2-234**
Example: Creating an icicle plot

\[
\text{COORD: rect(dim(1,2), reflect(dim(2))})
\text{ELEMENT: interval(position(x\text{*}y))}
\]
**Coordinate Types and Transformations**

parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), mirror Function (GPL), project Function (GPL), transpose Function (GPL), wrap Function (GPL)

**Applies To**

COORD Statement (GPL), parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), project Function (GPL)

**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 p. 4 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**

Figure 2-235

Example: Creating error bars

ELEMENT: interval(position(region.confi.count(jobcat)), shape(shape.ibeam))

**Statistic Functions**

See GPL Functions on p. 59.

**Valid Functions**

alpha Function (GPL)
**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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 p. 4 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

Figure 2-236
Example: Creating error bars

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

Statistic Functions

See GPL Functions on p. 59.

Valid Functions

alpha Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.percent.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL),
**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 \( x \times y \). 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 p. 4 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**

Figure 2-237

*Example: Creating error bars*

```
ELEMENT: interval(position(region.confi.percent.count(jobcat)), shape(shape.ibeam))
```

**Statistic Functions**

See *GPL Functions* on p. 59.

**Valid Functions**

alpha Function (GPL)

**Base Functions**

base.aesthetic Function (GPL), base.all Function (GPL), base.coordinate Function (GPL)
Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 \( x \times y \). 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 p. 4 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().
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**

Figure 2-238
Example: Creating error bars

```plaintext
ELEMENT: interval(position(region.confi.proportion.count(jobcat)), shape(shape.ibeam))
```

**Statistic Functions**

See GPL Functions on p. 59.

**Valid Functions**

- `alpha Function (GPL)`

**Base Functions**

- `base.aesthetic Function (GPL)`, `base.all Function (GPL)`, `base.coordinate Function (GPL)`

**Applies To**

- `bin.dot Function (GPL)`, `bin.hex Function (GPL)`, `bin.quantile.letter Function (GPL)`, `bin.rect Function (GPL)`, `color Function (For GPL Graphic Elements)`, `color.brightness Function (For GPL Graphic Elements)`, `color.hue Function (For GPL Graphic Elements)`, `color.saturation Function (For GPL Graphic Elements)`, `link.alpha Function (GPL)`, `link.complete Function (GPL)`, `link delaunay Function (GPL)`, `link.distance Function (GPL)`, `link.gabriel Function (GPL)`, `link.hull Function (GPL)`, `link.influence Function (GPL)`, `link.join Function (GPL)`, `link.mst Function (GPL)`, `link.neighbor Function (GPL)`, `link.relativeNeighborhood Function (GPL)`, `link.sequence Function (GPL)`, `link.tsp Function (GPL)`, `position Function (For GPL Graphic Elements)`, `region.confi.count Function (GPL)`, `region.confi.mean Function (GPL)`, `region.confi.percent.count Function (GPL)`, `region.confi.percent.count.cumulative Function (GPL)`, `region.confi.percent.count.cumulative Function (GPL)`, `region.confi.percent.count.cumulative Function (GPL)`, `region.confi.percent.count.cumulative Function (GPL)`, `region.confi.percent.count.cumulative Function (GPL)`
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 p. 4 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

Figure 2-239
Example: Showing the confidence interval around a fit line

ELEMENT: line(position(region.confi.smooth.linear(salbegin*salary)))

Valid Functions

alpha Function (GPL), noConstant Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function
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 p. 4 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.
Chapter 2

Examples

Figure 2-240
Example: Range bar showing minimum and maximum of one variable

ELEMENT: interval(position(region.spread.range(jobcat*salary)))

Figure 2-241
Example: Range bar showing minimum of one variable to maximum of another

ELEMENT: interval(position(region.spread.range(jobcat*(salbegin+salary))))

Statistic Functions

See GPL Functions on p. 59.

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

region.spread.sd Function

Syntax

region.spread.sd(<algebra>, <function>)
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 p. 4 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

Figure 2-242
Example: Creating an error bar

ELEMENT: interval(position(region.spread.sd(jobcat*salary)))

Figure 2-243
Example: Creating an error bar for 2 standard deviations

ELEMENT: interval(position(region.spread.sd(jobcat*salary, multiple(2))))

Statistic Functions

See GPL Functions on p. 59.

Valid Functions

multiple Function (GPL)

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)
Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.percent.count.cumulative Function (GPL), region.confi.percent.count.cumulative Function (GPL), region.confi.percent.mean Function (GPL), region.confi.percent.sum Function (GPL), region.confi.percent.sum.cumulative Function (GPL), region.confi.percentTrue Function (GPL), region.confi.percentTrue Function (GPL), region.spread.range Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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**

- **Figure 2-244**  
  *Example: Creating an error bar*  
  ```
  ELEMENT: interval(position(region.spread.se(jobcat*salary)))
  ```

- **Figure 2-245**  
  *Example: Creating an error bar for 2 standard errors*  
  ```
  ELEMENT: interval(position(region.spread.se(jobcat*salary, multiple(2))))
  ```

**Statistic Functions**

See GPL Functions on p. 59.

**Valid Functions**

- multiple Function (GPL)

**Binning Functions**

- bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

- bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confidence.count Function (GPL), region.confidence.mean Function (GPL), region.confidence.percent.count Function (GPL), region.confidence.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL),
summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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**

**Figure 2-246**

Example: Using reverse alpha-numeric sorting

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

**Figure 2-247**

Example: Reversing a linear scale

SCALE: linear(dim(2), reverse())

**Figure 2-248**

Example: Reversing the direction of an aesthetic scale

SCALE: linear(aesthetic(aesthetic.color.brightness), reverse())

**Figure 2-249**

Example: Reversing the direction of a coordinate system

COORD: polar.theta(reverse())
Applies To
polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), cat Scale (GPL), linear Scale (GPL), log Scale (GPL), pow Scale (GPL), safeLog Scale (GPL), safePower Scale (GPL), time Scale (GPL)

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
Figure 2-250
Example: Specifying a root node

ELEMENT: edge(position(layout.tree(node(id), from(fromVar), to(toVar), root("A"))))

Applies To
layout.tree Function (GPL)

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
Figure 2-251
Example: Creating a scatterplot with equal scales

COORD: rect(dim(1,2), sameRatio())
ELEMENT: point(position(salbegin*salary))

Applies To
rect Coordinate Type (GPL)
Chapter 2

**savSource Function**

**Syntax**

```
savSource(file("file path"))
```

*Note:* This function works with the PASW Statistics data file driver. You must have the driver installed and configured before using the function. You can download the driver and its documentation from [http://www.spss.com/drivers](http://www.spss.com/drivers). You can also use the `userSource` function to read PASW Statistics data directly. For more information, see the topic `userSource Function` on p. 263.

*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. You can also use forward slashes.

**Description**

Reads the contents of a SAV data file for PASW Statistics. This function is used to assign the contents of the file to a data source.

**Examples**

Figure 2-252

*Example: Reading a SAV file*

```
SOURCE: mydata = savSource(file("/Data/Employee data.sav"))
```

**Applies To**

SOURCE Statement (GPL), csvSource Function (GPL), sqlSource Function (GPL)

**scale Function (For GPL Axes)**

*Note:* If you are specifying a size for a graph, refer to `scale Function (For GPL Graphs)` on p. 185. 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 p. 185. If you are specifying a size for a page, refer to `scale Function (For GPL Pages)` on p. 186.

**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
Figure 2-253
Example: Associating an axis with a named scale
SCALE: y2= linear(dim(2))
GUIDE: axis(scale(y2), label("Count"))

Applies To
axis Guide Type (GPL)

scale Function (For GPL Graphs)

Note: If you are specifying a scale associated with an axis, refer to scale Function (For GPL Axes) on p. 184. 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 p. 185. If you are specifying a size for a page, refer to scale Function (For GPL Pages) on p. 186.

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
Figure 2-254
Example: Sizing a graph with absolute units
GRAPH: begin(scale(2in, 4in))

Figure 2-255
Example: Sizing a graph with percentages
GRAPH: begin(scale(80%, 100%))

Applies To
begin Function (For GPL Graphs)

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 p. 184. If you are specifying a size for a graph, refer to scale Function (For GPL Graphs) on p. 185. If you are specifying a size for a page, refer to scale Function (For GPL Pages) on p. 186.
Chapter 2

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

Figure 2-256
Example: Associating a graphic element with a named scale

SCALE: y2= linear(dim(2))
ELEMENT: line(scale(y2), position(summary.count(x)))

Applies To

form.line Guide Type (GPL), area Element (GPL), edge Element (GPL), interval Element (GPL), line Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL)

scale Function (For GPL Pages)

Note: If you are specifying a scale associated with an axis, refer to scale Function (For GPL Axes) on p. 184. 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 p. 185. If you are specifying a size for a graph, refer to scale Function (For GPL Graphs) on p. 185.

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

Figure 2-257
Example: Sizing a visualization

PAGE: begin(scale(500px, 400px))
**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**

Figure 2-258
Example: Adding a kernel distribution

ELEMENT: line(position(density.kernel.epanechnikov(x, segments(150))))

**Applies To**

density.kernel Function (GPL)

**shape Function (For GPL Graphic Elements)**

*Note:* If you are modifying the shape for a guide, refer to shape Function (For GPL Guides) on p. 188.

**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. For more information, see the topic Shape Constants in Appendix A on p. 345.

**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**

Figure 2-259
*Example: Specifying a shape value*

ELEMENT: point(position(salbegin*salary), shape.interior(shape.square))

Figure 2-260
*Example: Using the values of a variable to control shape*

ELEMENT: point(position(salbegin*salary), shape.interior(jobcat))

Figure 2-261
*Example: Using the values of a variable to control line dashing*

ELEMENT: line(position(salbegin*salary), shape.interior(jobcat))

Figure 2-262
*Example: Using the values of a variable to control border dashing*

ELEMENT: interval(position(gender*salary*jobcat), shape.exterior(gender))

**Applies To**

edge Element (GPL), interval Element (GPL), line Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL)

**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 p. 187.

**Syntax**

shape(shape.<shape constant>)

**<shape constant>**. A constant indicating a specific shape, such as shape.dash. For more information, see the topic Shape Constants in Appendix A on p. 345.

**Description**

Controls the dashing of reference lines.
**Examples**

Figure 2-263  
Example: Specifying a dashed reference line  

GUIDE: form.line(position(*,2000), shape(shape.dash))

**Applies To**

form.line Guide Type (GPL)

---

**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**

Figure 2-264  
Example: Displaying all labels  

ELEMENT: point(position(x*y), label(z, showAll()))

**Applies To**

label Function (For GPL Graphic Elements)

---

**size Function (For GPL Graphic Elements)**

*Note:* If you are modifying the size for a guide, refer to size Function (For GPL Guides) on p. 190.

**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. For more information, see the topic Size Constants in Appendix A on p. 345.

<statistic function>. A statistic function.

Description

Specifies the size of the individual graphic elements.

Examples

Figure 2-265
Example: Using a variable to control size

ELEMENT: point(position(x*y), size(z))

Figure 2-266
Example: Specifying a percentage for size

ELEMENT: interval(position(x*y), size(size."60%"))

Figure 2-267
Example: Specifying a value for size

ELEMENT: interval(position(x*y), size(size."6px"))

Figure 2-268
Example: Specifying a constant for size

ELEMENT: interval(position(x*y), size(size.large))

Statistic Functions

See GPL Functions on p. 59.

Applies To

edge Element (GPL), interval Element (GPL), line Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL)

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 p. 189.

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. For more information, see the topic Size Constants in Appendix A on p. 345.

**Description**

Controls the thickness of reference lines.

**Examples**

Figure 2-269

*Example: Specifying a reference line with a thickness of 15 pixels*

GUIDE: form.line(position(*,1000), size(size."15px"))

**Applies To**

form.line Guide Type (GPL)

---

**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 p. 4 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**

Figure 2-270

*Example: Creating a cubic fit line*

ELEMENT: line(position(smooth.cubic(salbegin*salary)))
Chapter 2

**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 (GPL), noConstant Function (GPL)

**Applies To**

- bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conf.count Function (GPL), region.conf.mean Function (GPL), region.conf.percent.count Function (GPL), region.conf.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.sc.kurtosis Function (GPL), summary.sc.skewness Function (GPL),
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 p. 4 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

Figure 2-271
Example: Creating a linear fit line

ELEMENT: line(position(smooth.linear(salbegin*salary)))

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 (GPL), noConstant Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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**

Figure 2-272
*Example: Creating a loess fit line*

ELEMENT: line(position(smooth.loess(salbegin*salary)))

Figure 2-273
*Example: Creating a loess fit line with specific kernel*

ELEMENT: line(position(smooth.loess.uniform(salbegin*salary)))

**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 (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL),
region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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

Figure 2-274
Example: Creating a line at the mean of the y axis

ELEMENT: line(position(smooth.mean.uniform(salbegin*salary)))
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 (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conf.count Function (GPL), region.conf.mean Function (GPL), region.conf.percent.count Function (GPL), region.conf.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.sc.kurtosis Function (GPL), summary.se.skewness Function (GPL),
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 p. 4 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

Figure 2-275
Example: Creating a line at the median of the y axis

ELEMENT: line(position(smooth.median.uniform(salbegin*salary)))

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 (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confidence.count Function (GPL), region.confidence.mean Function (GPL), region.confidence.percent.count Function (GPL), region.confidence.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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

Figure 2-276  
Example: Creating a quadratic fit line

ELEMENT: line(position(smooth.quadratic(salbegin*salary)))

Kernel Functions

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniform</td>
<td>All data receive equal weights.</td>
</tr>
<tr>
<td>epanechnikov</td>
<td>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.</td>
</tr>
<tr>
<td>biweight</td>
<td>Data far from the current point receive more weight than the triweight kernel allows but less weight than the Epanechnikov kernel permits.</td>
</tr>
<tr>
<td>tricube</td>
<td>Data close to the current point receive higher weights than both the Epanechnikov and biweight kernels allow.</td>
</tr>
<tr>
<td>triweight</td>
<td>Data close to the current point receive higher weights than any other kernel allows. Extreme cases get very little weight.</td>
</tr>
<tr>
<td>gaussian</td>
<td>Weights follow a normal distribution, resulting in higher weighting of extreme cases than the Epanechnikov, biweight, tricube, and triweight kernels.</td>
</tr>
<tr>
<td>cauchy</td>
<td>Extreme values receive more weight than the other kernels, with the exception of the uniform kernel, allow.</td>
</tr>
</tbody>
</table>

Valid Functions

proportion Function (GPL), noConstant Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL),
summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**smooth.spline Function**

**Syntax**

```
smooth.spline(<algebra>)
```

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 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**

Figure 2-277

Example: Creating a cubic spline curve

```
ELEMENT: line(position(smooth.spline(salbegin*salary)))
```

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.percent.cumulative Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function
smooth.step Function

Syntax

smooth.step(<algebra>)

or

smooth.step.<position>(<algebra>)

<algebra>. Graph algebra, such as \( x \times y \). Refer to Brief Overview of GPL Algebra on p. 4 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

Figure 2-278
Example: Creating a step interpolation line

ELEMENT: line(position(smooth.step.center(salbegin*salary)))

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function
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 PASW 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

Figure 2-279
Example: Sorting the categories

SCALE: cat(dim(1), sort.data())

Applies To

cat Scale (GPL)
**sort.natural Function**

**Syntax**

sort.natural()

**Description**

Sorts the categorical values in alphanumerical order.

**Examples**

Figure 2-280

Example: Sorting the categories

SCALE: cat(dim(1), values("Male", "Female"), sort.natural())

**Applies To**

cat Scale (GPL)

---

**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**

Figure 2-281

Example: Sorting the categories

SCALE: cat(dim(1), sort.statistic(summary.mean(salary)))

**Statistic Functions**

See GPL Functions on p. 59.

**Applies To**

cat Scale (GPL)
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.

Examples

Figure 2-282  
Example: Sorting the categories explicitly

SCALE: cat(dim(1), sort.values("Male"))

In this example, Male will appear first. It is not necessary to specify that Female will appear next.

Applies To

cat Scale (GPL)

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

Figure 2-283  
Example: Creating groups of lines

ELEMENT: line(position(salbegin*salary), split(gender))
**Statistic Functions**

See [GPL Functions](#) on p. 59.

**Applies To**

area Element (GPL), edge Element (GPL), interval Element (GPL), line Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL)

### sqlSource Function

**Syntax**

```r
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**

Figure 2-284

Example: Reading from a database

```r
SOURCE: mydata = sqlSource(url("jdbc:microsoft:sqlserver://localhost:1433"), user("fred"),
password("secret"), query("select * from employeeData"))
```

**Valid Functions**

missing.listwise Function (GPL), missing.pairwise Function (GPL), weight Function (GPL)

**Applies To**

SOURCE Statement (GPL), csvSource Function (GPL), savSource Function (GPL)

### start Function

**Syntax**

```r
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**

Figure 2-285  
*Example: Specifying the first major tick*

GUIDE: axis(dim(1), start(1000))

**Applies To**

axis Guide Type (GPL)

---

**startAngle Function**

**Syntax**

```
sstartAngle(<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**

Figure 2-286  
*Example: Specifying the first slice at 9:00*

COORD: polar.theta(startAngle(90))

Figure 2-287  
*Example: Specifying the first slice at 3:00 (90 degrees)*

COORD: polar.theta(startAngle(-90))

**Applies To**

polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL)

---

**studentizedRange Function**

**Syntax**

```
ststudentizedRange(<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

Figure 2-288
Example: Specifying a Studentized range distribution for the probability scale

SCALE: prob(dim(2), studentizedRange(5, 2.5))

Applies To

prob Scale (GPL)

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 p. 4 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

Figure 2-289
Example: Specifying a bar chart of counts

ELEMENT: interval(position(summary.count(jobcat)))
Figure 2-290
Example: Counting non-missing cases for a continuous variable

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

Figure 2-291
Example: Specifying a histogram

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

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)
Chapter 2

**summary.count.cumulative Function**

**Syntax**

```plaintext
summary.count.cumulative(<algebra>)
```

or

```plaintext
summary.count.cumulative(<binning function>)
```

or

```plaintext
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 p. 4 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**

**Figure 2-292**
**Example: Specifying a bar chart of cumulative counts**

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

**Figure 2-293**
**Example: Specifying a cumulative histogram**

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

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

`bin.dot Function (GPL)`, `bin.hex Function (GPL)`, `bin.quantile.letter Function (GPL)`, `bin.rect Function (GPL)`
Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confidence.count Function (GPL), region.confidence.mean Function (GPL), region.confidence.percent.count Function (GPL), region.confidence.percent.count.cumulative Function (GPL), region.confluence.count Function (GPL), region.confluence.mean Function (GPL), region.confluence.percent.count Function (GPL), region.confluence.percent.count.cumulative Function (GPL), region.confluence.percent.sum Function (GPL), region.confluence.percent.sum.cumulative Function (GPL), region.confluence.percentTrue Function (GPL), region.confluence.proportion.count Function (GPL), region.confluence.proportion.count.cumulative Function (GPL), region.confluence.proportion.cumulative Function (GPL), region.confluence.proportion.sum Function (GPL), region.confluence.proportion.sum.cumulative Function (GPL), region.confluence.proportionTrue Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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 p. 4.

**Examples**

**Figure 2-294**  
*Example: Plotting count greater than a value*

TRANS: salGreaterThan = eval(salary>50000)  
ELEMENT: interval(position(summary.countTrue(jobcat*salGreaterThan)))

**Figure 2-295**  
*Example: Plotting count less than, equal to, and 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)))

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confidence.count Function (GPL), region.confidence.mean Function (GPL), region.confidence.percent.count Function (GPL), region.confidence.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative
Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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 p. 4 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 p. 4.

*Examples*

Figure 2-296  
*Example: Calculating the first salary value for each jobcat category*

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

*Statistic Functions*

See GPL Functions on p. 59.

*Binning Functions*

`bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)`
Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeneighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conficount Function (GPL), region.confimean Function (GPL), region.confipercent.count Function (GPL), region.confiproportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.sc Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percenttrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportiontrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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 p. 4.

**Examples**

Figure 2-297  
*Example: Calculating the kurtosis of salary for each jobcat category*

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

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

- `bin.dot Function (GPL)`
- `bin.hex Function (GPL)`
- `bin.quantile.letter Function (GPL)`
- `bin.rect Function (GPL)`

**Applies To**

- `bin.dot Function (GPL)`
- `bin.hex Function (GPL)`
- `bin.quantile.letter Function (GPL)`
- `bin.rect Function (GPL)`
- `color Function (For GPL Graphic Elements)`
- `color.brightness Function (For GPL Graphic Elements)`
- `color.hue Function (For GPL Graphic Elements)`
- `color.saturation Function (For GPL Graphic Elements)`
- `link.alpha Function (GPL)`
- `link.complete Function (GPL)`
- `link.delaunay Function (GPL)`
- `link.distance Function (GPL)`
- `link.gabriel Function (GPL)`
- `link.hull Function (GPL)`
- `link.influence Function (GPL)`
- `link.join Function (GPL)`
- `link.mst Function (GPL)`
- `link.neighbor Function (GPL)`
- `link.relativeNeighborhood Function (GPL)`
- `link.sequence Function (GPL)`
- `link.tsp Function (GPL)`
- `position Function (For GPL Graphic Elements)`
- `region.confi.count Function (GPL)`
- `region.confifull.count Function (GPL)`
- `region.confimean Function (GPL)`
- `region.confiprop.count Function (GPL)`
- `region.spread.range Function (GPL)`
- `region.spread.sd Function (GPL)`
- `region.spread.se Function (GPL)`
- `size Function (For GPL Graphic Elements)`
- `split Function (GPL)`
- `summary.count Function (GPL)`
- `summary.count.cumulative Function (GPL)`
- `summary.countTrue Function (GPL)`
- `summary.first Function (GPL)`
- `summary.last Function (GPL)`
- `summary.max Function (GPL)`
- `summary.mean Function (GPL)`
- `summary.median Function (GPL)`
- `summary.min Function (GPL)`
- `summary.mode Function (GPL)`
- `summary.percent Function (GPL)`
- `summary.percent.count Function (GPL)`
- `summary.percent.cumulative Function (GPL)`
- `summary.percent.lambda Function (GPL)`
- `summary.percentTable Function (GPL)`
- `summary.percentTrue Function (GPL)`
- `summary.proportion Function (GPL)`
- `summary.proportion.count Function (GPL)`
- `summary.proportion.cumulative Function (GPL)`
- `summary.proportion.lambda Function (GPL)`
- `summary.proportionTable Function (GPL)`
- `summary.proportionTrue Function (GPL)`
- `summary.range Function (GPL)`
- `summary.sd Function (GPL)`
- `summary.se Function (GPL)`
- `summary.se.kurtosis Function (GPL)`
- `summary.se.skewness Function (GPL)`
- `summary.sum Function (GPL)`
- `summary.sum.cumulative Function (GPL)`
- `summary.sumTable Function (GPL)`
**summary.last Function**

**Syntax**

\[
\text{summary.last(<algebra>)}
\]

\[
\text{or}
\]

\[
\text{summary.last(<binning function>)}
\]

\[
\text{or}
\]

\[
\text{summary.last(<statistic function>)}
\]

**<algebra>**. Graph algebra, such as \(x*y\). Refer to Brief Overview of GPL Algebra on p. 4 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 \(\text{summary.max}\).

**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 p. 4.

**Examples**

Figure 2-298

*Example: Calculating the last salary value for each jobcat category*

\[
\text{ELEMENT: interval(position(summary.last(jobcat*salary)))}
\]

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function
(GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confli.count Function (GPL), region.confli.mean Function (GPL), region.confli.percent.count Function (GPL), region.confli.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.cumulative.count Function (GPL), summary.percent.cumulative.count.cumulative Function (GPL), summary.percent.cumulative.percent Function (GPL), summary.percent.cumulative.sum Function (GPL), summary.percent.cumulative.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative.count Function (GPL), summary.proportion.cumulative.count.cumulative Function (GPL), summary.proportion.cumulative.percent Function (GPL), summary.proportion.cumulative.sum Function (GPL), summary.proportion.cumulative.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**summary.max Function**

**Syntax**

```plaintext
summary.max(<algebra>)
```

*or*

```plaintext
summary.max(<binning function>)
```

*or*

```plaintext
summary.max(<statistic function>)
```

*<algebra>*. Graph algebra, such as \( x \cdot y \). Refer to Brief Overview of GPL Algebra on p. 4 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`. 
Chapter 2

**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 p. 4.

**Examples**

Figure 2-299

*Example: Calculating the maximum salary for each jobcat category*

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

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

`bin.dot` Function (GPL), `bin.hex` Function (GPL), `bin.quantile.letter` Function (GPL), `bin.rect` Function (GPL)

**Applies To**

`bin.dot` Function (GPL), `bin.hex` Function (GPL), `bin.quantile.letter` Function (GPL), `bin.rect` Function (GPL), `color` Function (For GPL Graphic Elements), `color.brightness` Function (For GPL Graphic Elements), `color.hue` Function (For GPL Graphic Elements), `color.saturation` Function (For GPL Graphic Elements), `link.alpha` Function (GPL), `link.complete` Function (GPL), `link.delaunay` Function (GPL), `link.distance` Function (GPL), `link.gabriel` Function (GPL), `link.hull` Function (GPL), `link.influence` Function (GPL), `link.join` Function (GPL), `link.mst` Function (GPL), `link.neighbor` Function (GPL), `link.relativeNeighborhood` Function (GPL), `link.sequence` Function (GPL), `link.tsp` Function (GPL), `position` Function (For GPL Graphic Elements), `region.confi.count` Function (GPL), `region.confi.mean` Function (GPL), `region.confi.percent.count` Function (GPL), `region.confi.proportion.count` Function (GPL), `region.spread.range` Function (GPL), `region.spread.sd` Function (GPL), `region.spread.se` Function (GPL), `size` Function (For GPL Graphic Elements), `split` Function (GPL), `summary.count` Function (GPL), `summary.cumulative` Function (GPL), `summary.first` Function (GPL), `summary.kurtosis` Function (GPL), `summary.last` Function (GPL), `summary.mean` Function (GPL), `summary.median` Function (GPL), `summary.min` Function (GPL), `summary.mode` Function (GPL), `summary.percent` Function (GPL), `summary.percent.cumulative` Function (GPL), `summary.percent.sum` Function (GPL), `summary.percentTrue` Function (GPL), `summary.proportion` Function (GPL), `summary.proportion.count` Function (GPL), `summary.proportion.cumulative` Function (GPL), `summary.proportion.sum` Function (GPL), `summary.sd` Function (GPL), `summary.se` Function (GPL), `summary.se.kurtosis` Function (GPL), `summary.se.skewness` Function (GPL), `summary.sum` Function (GPL), `summary.sum.cumulative` Function (GPL)
Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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 p. 4 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 p. 4.

**Examples**

Figure 2-300

*Example: Calculating the mean salary for each jobcat category*

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

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function
**summary.median Function**

**Syntax**

```c
summary.median(<algebra>)
```

**or**

```c
summary.median(<binning function>)
```

**or**

```c
summary.median(<statistic function>)
```

**<algebra>**. Graph algebra, such as `x*y`. Refer to Brief Overview of GPL Algebra on p. 4 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 p. 4.

Examples

Figure 2-301

Example: Calculating the median salary for each jobcat category

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

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confidence.count Function (GPL), region.confidence.mean Function (GPL), region.confidence.percent.count Function (GPL), region.confidence.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), ...
Chapter 2

Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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 p. 4 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 p. 4.

**Examples**

Figure 2-302

*Example: Calculating the minimum salary for each jobcat category*

ELEMENT: interval(position(summary.min(jobcat*salary)))

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function
**summary.mode Function**

**Syntax**

```r
summary.mode(<algebra>)
```

_or_

```r
summary.mode(<binning function>)
```

_or_

```r
summary.mode(<statistic function>)
```

*<algebra>*. Graph algebra, such as x*y. Refer to *Brief Overview of GPL Algebra* on p. 4 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 p. 4.

Examples

Figure 2-303
Example: Calculating the mode of educationalLevel for each jobcat category

ELEMENT: interval(position(summary.mode(jobcat*educationalLevel)))

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confidence.count Function (GPL), region.confidence.mean Function (GPL), region.confidence.percent.count Function (GPL), region.confidence.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL)
Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**summary.percent Function**

**Description**

`summary.percent` is an alias for `summary.percent.sum`. For more information, see the topic `summary.percent.sum Function` on p. 229.

**Examples**

Figure 2-304

Example: Calculating percentages of a summed variable

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

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conf.interval Function (GPL), region.conf.mean Function (GPL), region.conf.percent.count Function (GPL), region.conf.proportion.cumulative Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.mean Function (GPL), summary.percent.median Function (GPL), summary.percent.min Function (GPL), summary.percent.max Function (GPL), summary.percent.range Function (GPL), summary.percent.sd Function (GPL), summary.percent.se Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.first Function (GPL), summary.proportion.mean Function (GPL), summary.proportion.median Function (GPL), summary.proportion.min Function (GPL), summary.proportion.max Function (GPL), summary.proportion.range Function (GPL), summary.proportion.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.median Function (GPL), summary.se.mean Function (GPL), summary.se.max Function (GPL), summary.se.min Function (GPL), summary.second Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)
### 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 p. 4 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**

Figure 2-305  
Example: Calculating percentages of counts

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

Figure 2-306  
Example: Graphing a histogram of percentages

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

**Statistic Functions**

See GPL Functions on p. 59.

**Base Functions**

base.aesthetic Function (GPL), base.all Function (GPL), base.coordinate Function (GPL)

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)
A **Applies To**

- bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conficonfcount Function (GPL), region.confimean Function (GPL), region.confipercent.count Function (GPL), region.confiproportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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 \( x \cdot y \). In the second case, the percentage is calculated for cases with non-missing \( y \)-variable values. Refer to Brief Overview of GPL Algebra on p. 4 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**

**Figure 2-307**
*Example: Calculating cumulative percentages of counts*

```r
ELEMENT: interval(position(summary.percent.count.cumulative(jobcat)))
```

**Figure 2-308**
*Example: Calculating a cumulative histogram of percentages*

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

**Statistic Functions**

See GPL Functions on p. 59.

**Base Functions**

- `base.aesthetic Function (GPL)`, `base.all Function (GPL)`, `base.coordinate Function (GPL)`

**Binning Functions**

- `bin.dot Function (GPL)`, `bin.hex Function (GPL)`, `bin.quantile.letter Function (GPL)`, `bin.rect Function (GPL)`

**Applies To**

- `bin.dot Function (GPL)`, `bin.hex Function (GPL)`, `bin.quantile.letter Function (GPL)`, `bin.rect Function (GPL)`, `color Function (For GPL Graphic Elements)`, `color.brightness Function (For GPL Graphic Elements)`, `color.hue Function (For GPL Graphic Elements)`, `color.saturation Function (For GPL Graphic Elements)`, `link.alpha Function (GPL)`, `link.complete Function (GPL)`, `link.delaunay Function (GPL)`, `link.distance Function (GPL)`, `link.gabriel Function (GPL)`, `link.hull Function (GPL)`, `link.influence Function (GPL)`, `link.join Function (GPL)`, `link.mst Function (GPL)`, `link.neighbor Function (GPL)`, `link.relativeNeighborhood Function (GPL)`, `link.sequence Function (GPL)`, `link.tsp Function (GPL)`, `position Function (For GPL Graphic Elements)`, `region.confi.count Function (GPL)`, `region.confi.mean Function (GPL)`, `region.confi.percent.count Function (GPL)`, `region.confi.proportion.count Function (GPL)`, `region.spread.range Function (GPL)`, `region.spread.sd Function (GPL)`, `region.spread.se Function (GPL)`, `size Function (For GPL Graphic Elements)`, `split Function (GPL)`, `summary.count Function (GPL)`
(GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL),
summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL),
summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL),
summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL),
summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL),
summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL),
summary.proportion Function (GPL), summary.proportion.count Function (GPL),
summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL),
summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL),
summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL),
summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL),
summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

### summary.percent.cumulative Function

**Description**

`summary.percent.cumulative` is an alias for `summary.percent.sum.cumulative`. For more information, see the topic `summary.percent.sum.cumulative` Function on p. 231.

**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**

**Figure 2-309**

*Example: Calculating cumulative percentages of a summed variable*

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

### 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 p. 4 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. For more information, see the topic `summary.percent.count Function` on p. 226.

**Examples**

Figure 2-310
Example: Calculating percentages of a summed variable

```plaintext
ELEMENT: interval(position(summary.percent.sum(jobcat*salary)))
```

**Statistic Functions**

See GPL Functions on p. 59.

**Base Functions**

`base.aesthetic Function (GPL)`, `base.all Function (GPL)`, `base.coordinate Function (GPL)`

**Binning Functions**

`bin.dot Function (GPL)`, `bin.hex Function (GPL)`, `bin.quantile.letter Function (GPL)`, `bin.rect Function (GPL)`

**Applies To**

`bin.dot Function (GPL)`, `bin.hex Function (GPL)`, `bin.quantile.letter Function (GPL)`, `bin.rect Function (GPL)`, `color Function (For GPL Graphic Elements)`, `color.brightness Function (For GPL Graphic Elements)`, `color.hue Function (For GPL Graphic Elements)`, `color.saturation Function (For GPL Graphic Elements)`, `link.alpha Function (GPL)`, `link.complete Function (GPL)`, `link.delaunay Function (GPL)`, `link.distance Function (GPL)`, `link.gabriel Function (GPL)`, `link.hull Function (GPL)`, `link.influence Function (GPL)`, `link.join Function (GPL)`, `link.mst Function (GPL)`, `link.neighbor Function (GPL)`, `link.relativeNeighborhood Function (GPL)`, `link.sequence Function (GPL)`, `link.tsp Function (GPL)`, `position Function (For GPL Graphic Elements)`, `region.confidence.count Function (GPL)`, `region.confidence.mean Function (GPL)`, `region.confidence.percent.count Function (GPL)`, `region.confidence.proportion.count Function (GPL)`, `region.spread.range Function (GPL)`, `region.spread.sd Function (GPL)`, `region.spread.se Function (GPL)`, `size Function (For GPL Graphic Elements)`, `split Function (GPL)`, `summary.count Function (GPL)`, `summary.count.cumulative Function (GPL)`, `summary.count.True Function (GPL)`, `summary.first Function (GPL)`, `summary.kurtosis Function (GPL)`, `summary.last Function (GPL)`,
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 p. 4 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. For more information, see the topic summary.percent.count.cumulative Function on p. 227.

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.
Chapter 2

Examples

Figure 2-31
Example: Calculating cumulative percentages of a summed variable

```r
ELEMENT: interval(position(summary.percent.sum.cumulative(jobcat*salary)))
```

Statistic Functions

See GPL Functions on p. 59.

Base Functions

base.aesthetic Function (GPL), base.all Function (GPL), base.coordinate Function (GPL)

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)
**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 \( x \times y \). Refer to *Brief Overview of GPL Algebra* on p. 4 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 p. 4.

**Examples**

Figure 2-312

*Example: Calculating the 25th percentile of salary for each jobcat category*

```
ELEMENT: interval(position(summary.percentile(jobcat*salary, alpha(0.25))))
```

**Statistic Functions**

See *GPL Functions* on p. 59.

**Valid Functions**

alpha Function (GPL)

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)
Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confim.count Function (GPL), region.confim.mean Function (GPL), region.confim.percent.count Function (GPL), region.confim.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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 p. 4.

**Examples**

Figure 2-313  
*Example: Plotting percentage greater than a value*

TRANS: salGreaterThan = eval(salary>50000)  
ELEMENT: interval(position(summary.percentTrue(jobcat*salGreaterThan)))

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), position Function (For GPL Graphic Elements), region.conf.internal Function (GPL), region.conf.mean Function (GPL), region.conf.percent.count Function (GPL), region.conf.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.percent Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percentile Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL),
summary.proportion Function

**Description**

`summary.proportion` is an alias for `summary.proportion.sum`. For more information, see the topic `summary.proportion.sum` Function on p. 240.

**Examples**

Figure 2-314  
*Example: Calculating proportions of a summed variable*

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

**Applies To**

- `bin.dot` Function (GPL)
- `bin.hex` Function (GPL)
- `bin.quantile.letter` Function (GPL)
- `bin.rect` Function (GPL)
- `color` Function (For GPL Graphic Elements)
- `color.brightness` Function (For GPL Graphic Elements)
- `color.hue` Function (For GPL Graphic Elements)
- `color.saturation` Function (For GPL Graphic Elements)
- `link.alpha` Function (GPL)
- `link.complete` Function (GPL)
- `link.delaunay` Function (GPL)
- `link.distance` Function (GPL)
- `link.gabriel` Function (GPL)
- `link.hull` Function (GPL)
- `link.influence` Function (GPL)
- `link.join` Function (GPL)
- `link.mst` Function (GPL)
- `link.neighbor` Function (GPL)
- `link.relativeNeighborhood` Function (GPL)
- `link.sequence` Function (GPL)
- `link.tsp` Function (GPL)
- `position` Function (For GPL Graphic Elements)
- `region.confini.count` Function (GPL)
- `region.confini.mean` Function (GPL)
- `region.confini.percent.count` Function (GPL)
- `region.confini.proportion.count` Function (GPL)
- `region.spread.range` Function (GPL)
- `region.spread.sd` Function (GPL)
- `region.spread.se` Function (GPL)
- `size` Function (For GPL Graphic Elements)
- `split` Function (GPL)
- `summary.count` Function (GPL)
- `summary.count.cumulative` Function (GPL)
- `summary.countTrue` Function (GPL)
- `summary.first` Function (GPL)
- `summary.kurtosis` Function (GPL)
- `summary.last` Function (GPL)
- `summary.max` Function (GPL)
- `summary.mean` Function (GPL)
- `summary.median` Function (GPL)
- `summary.min` Function (GPL)
- `summary.mode` Function (GPL)
- `summary.percent` Function (GPL)
- `summary.percent.count` Function (GPL)
- `summary.percent.count.cumulative` Function (GPL)
- `summary.percent.cumulative` Function (GPL)
- `summary.percent.sum` Function (GPL)
- `summary.percent.sum.cumulative` Function (GPL)
- `summary.percentTrue` Function (GPL)
- `summary.proportion` Function (GPL)
- `summary.proportion.count` Function (GPL)
- `summary.proportion.count.cumulative` Function (GPL)
- `summary.proportion.cumulative` Function (GPL)
- `summary.proportion.sum` Function (GPL)
- `summary.proportion.sum.cumulative` Function (GPL)
- `summary.range` Function (GPL)
- `summary.sd` Function (GPL)
- `summary.se` Function (GPL)
- `summary.se.kurtosis` Function (GPL)
- `summary.se.skewness` Function (GPL)
- `summary.sum` Function (GPL)
- `summary.sum.cumulative` Function (GPL)
- `summary.variance` Function (GPL)
- `transparency` Function (For GPL Graphic Elements)
**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 \(x \times y\). In the second case, the proportion is calculated for cases with non-missing \(y\)-variable values. Refer to Brief Overview of GPL Algebra on p. 4 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**

Figure 2-315

Example: Calculating proportions of counts

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

**Statistic Functions**

See GPL Functions on p. 59.

**Base Functions**

`base.aesthetic Function (GPL)`, `base.all Function (GPL)`, `base.coordinate Function (GPL)`

**Binning Functions**

`bin.dot Function (GPL)`, `bin.hex Function (GPL)`, `bin.quantile.letter Function (GPL)`, `bin.rect Function (GPL)`
Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delamay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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**

Figure 2-316

*Example: Calculating cumulative proportions of counts*

ELEMENT: interval(position(summary.proportion.count.cumulative(jobcat)))

**Statistic Functions**

See GPL Functions on p. 59.

**Base Functions**

base.aesthetic Function (GPL), base.all Function (GPL), base.coordinate Function (GPL)

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.sc Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function
Chapter 2

(GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

summary.proportion.cumulative Function

Description

summary.proportion.cumulative is an alias for summary.proportion.sum.cumulative. For more information, see the topic summary.proportion.sum.cumulative Function on p. 242.

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

Figure 2-317
Example: Calculating cumulative proportions of a summed variable

ELEMENT: interval(position(summary.proportion.cumulative(jobcat*salary)))

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 p. 4 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. For more information, see the topic summary.proportion.count Function on p. 237.

**Examples**

Figure 2-318
Example: Calculating proportions of a summed variable

ELEMENT: interval(position(summary.proportion.sum(jobcat*salary)))

**Statistic Functions**

See GPL Functions on p. 59.

**Base Functions**

base.aesthetic Function (GPL), base.all Function (GPL), base.coordinate Function (GPL)

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confci.count Function (GPL), region.confci.mean Function (GPL), region.confci.percent.count Function (GPL), region.confci.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL)
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 p. 4 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. For more information, see the topic summary.proportion.count.cumulative Function on p. 238.

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

Figure 2-319
Example: Calculating cumulative proportions of a summed variable

ELEMENT: interval(position(summary.proportion.sum.cumulative(jobcat*salary)))

Statistic Functions

See GPL Functions on p. 59.

Base Functions

base.aesthetic Function (GPL), base.all Function (GPL), base.coordinate Function (GPL)

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)
**summary.proportionTrue Function**

**Syntax**

\[
\text{summary.proportionTrue(}<\text{algebra}>\text{)}
\]

or

\[
\text{summary.proportionTrue(}<\text{binning function}>\text{)}
\]

or

\[
\text{summary.proportionTrue(}<\text{statistic function}>\text{)}
\]

<algebra>. Graph algebra, such as \(x*y\). Refer to Brief Overview of GPL Algebra on p. 4 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 \text{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 \text{summary.percentTrue} except it reports a value between 0 and 1 instead of 0 and 100. If \text{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 p. 4.

**Examples**

Figure 2-320

Example: Plotting proportion greater than a value

\[
\text{TRANS: salGreaterThan = eval(salary>50000)}
\]

\[
\text{ELEMENT: interval(position(summary.proportionTrue(jobcat*salGreaterThan)))}
\]

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

\text{bin.dot Function (GPL)}, \text{bin.hex Function (GPL)}, \text{bin.quantile.letter Function (GPL)}, \text{bin.rect Function (GPL)}

**Applies To**

\text{bin.dot Function (GPL)}, \text{bin.hex Function (GPL)}, \text{bin.quantile.letter Function (GPL)}, \text{bin.rect Function (GPL)}, \text{color Function (For GPL Graphic Elements)}, \text{color.brightness Function (For GPL Graphic Elements)}, \text{color.hue Function (For GPL Graphic Elements)}, \text{color.saturation Function (For GPL Graphic Elements)}, \text{link.alpha Function (GPL)}, \text{link.complete Function}
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 p. 4 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.
Chapter 2

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 p. 4.

This function returns only one value. If you want to display the interval between the minimum and maximum values, use `region.spread.range`. For more information, see the topic `region.spread.range Function` on p. 177.

Examples

Figure 2-321

Example: Calculating the range of salary for each jobcat category

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

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

`bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)`

Applies To

`bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.deluay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confim.count Function (GPL), region.confim.mean Function (GPL), region.confim.percent.count Function (GPL), region.confim.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.proportion Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL)`
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 p. 4 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 p. 4.

Examples

Figure 2-322
Example: Calculating the standard deviation of salary for each jobcat category

ELEMENT: interval(position(summary.sd(jobcat*salary)))

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)
Chapter 2

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

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 p. 4 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 p. 4.

Examples

Figure 2-323
Example: Calculating the standard error of salary for each jobcat category

ELEMENT: interval(position(summary.se(jobcat*salary)))

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum Function (GPL),
summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**summary.se.kurtosis Function**

**Syntax**

```r
summary.se.kurtosis(<algebra>)
```

**or**

```r
summary.se.kurtosis(<binning function>)
```

**or**

```r
summary.se.kurtosis(<statistic function>)
```

**<algebra>**. Graph algebra, such as `x*y`. Refer to Brief Overview of GPL Algebra on p. 4 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 p. 4.

**Examples**

Figure 2-324

*Example: Calculating the standard error of the kurtosis of salary for each jobcat category*

```r
ELEMENT: interval(position(summary.se.kurtosis(jobcat*salary)))
```

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation
Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.median Function (GPL), summary.mean Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.skewness Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**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 p. 4 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 p. 4.

Examples

Figure 2-325
Example: Calculating the standard error of the skewness of salary for each jobcat category

ELEMENT: interval(position(summary.se.skewness(jobcat*salary)))

Statistic Functions

See GPL Functions on p. 59.

Binning Functions

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.conf.count Function (GPL), region.conf.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.proportion.count Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.sum Function (GPL), summary.sum Function (GPL),
summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**summary.sum Function**

**Syntax**

```r
summary.sum(<algebra>)
```

or

```r
summary.sum(<binning function>)
```

or

```r
summary.sum(<statistic function>)
```

<algebra>. Graph algebra, such as x*y. Refer to Brief Overview of GPL Algebra on p. 4 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 p. 4.

**Examples**

Figure 2-326

*Example: Calculating the sum of salary for each jobcat category*

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

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function
(GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum.cumulative Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements)

**summary.sum.cumulative Function**

**Syntax**

```r
summary.sum.cumulative(<algebra>)
```

*or*

```r
summary.sum.cumulative(<binning function>)
```

*or*

```r
summary.sum.cumulative(<statistic function>)
```

**<algebra>**. Graph algebra, such as $x*y$. Refer to Brief Overview of GPL Algebra on p. 4 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 p. 4.

*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**

Figure 2-327  
*Example: Calculating the cumulative sum*

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

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)

**Applies To**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.range Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.mean Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentilе Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL)
Chapter 2

(summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.variance Function (GPL), transparency Function (For GPL Graphic Elements))

**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 p. 4 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 p. 4.

**Examples**

Figure 2-328

*Example: Calculating the variance of salary for each jobcat category*

ELEMENT: interval(position(summary.variance(jobcat*salary)))

**Statistic Functions**

See GPL Functions on p. 59.

**Binning Functions**

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL)
Applies To

bin.dot Function (GPL), bin.hex Function (GPL), bin.quantile.letter Function (GPL), bin.rect Function (GPL), color Function (For GPL Graphic Elements), color.brightness Function (For GPL Graphic Elements), color.hue Function (For GPL Graphic Elements), color.saturation Function (For GPL Graphic Elements), link.alpha Function (GPL), link.complete Function (GPL), link.delaunay Function (GPL), link.distance Function (GPL), link.gabriel Function (GPL), link.hull Function (GPL), link.influence Function (GPL), link.join Function (GPL), link.mst Function (GPL), link.neighbor Function (GPL), link.relativeNeighborhood Function (GPL), link.sequence Function (GPL), link.tsp Function (GPL), position Function (For GPL Graphic Elements), region.confi.count Function (GPL), region.confi.mean Function (GPL), region.confi.percent.count Function (GPL), region.confi.proportion.count Function (GPL), region.spread.neighborhood Function (GPL), region.spread.sd Function (GPL), region.spread.se Function (GPL), size Function (For GPL Graphic Elements), split Function (GPL), summary.count Function (GPL), summary.count.cumulative Function (GPL), summary.countTrue Function (GPL), summary.first Function (GPL), summary.kurtosis Function (GPL), summary.last Function (GPL), summary.max Function (GPL), summary.median Function (GPL), summary.min Function (GPL), summary.mode Function (GPL), summary.percent count Function (GPL), summary.percent.count Function (GPL), summary.percent.count.cumulative Function (GPL), summary.percent.cumulative Function (GPL), summary.percent.sum Function (GPL), summary.percent.sum.cumulative Function (GPL), summary.percentile Function (GPL), summary.percentTrue Function (GPL), summary.proportion Function (GPL), summary.proportion.count Function (GPL), summary.proportion.count.cumulative Function (GPL), summary.proportion.cumulative Function (GPL), summary.proportion.sum Function (GPL), summary.proportion.sum.cumulative Function (GPL), summary.proportionTrue Function (GPL), summary.range Function (GPL), summary.sd Function (GPL), summary.se.kurtosis Function (GPL), summary.se.skewness Function (GPL), summary.sum Function (GPL), summary.sum.cumulative Function (GPL), transparency Function (For GPL Graphic Elements)

t Function

Syntax

t(<degree of freedom>)

<degrees of freedom>. Numeric value indicating the degrees of freedom. This value must be greater than 0.

Description

Specifies a Student’s t distribution for the probability scale.

Examples

Figure 2-329
Example: Specifying a Student’s t distribution for the probability scale

SCALE: prob(dim(2), t(5))
Applies To

prob Scale (GPL)

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. For more information, see the topic Pattern Constants in Appendix A on p. 346.

<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

Figure 2-330
Example: Specifying a pattern

ELEMENT: line(position(x*y), texture.pattern(texture.pattern.checkered))

Figure 2-331
Example: Using the values of a variable to control pattern

ELEMENT: point(position(x*y), texture.pattern(z))

Statistic Functions

See GPL Functions on p. 59.

Applies To

area Element (GPL), interval Element (GPL), point Element (GPL), schema Element (GPL)
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

Figure 2-332
Example: Hiding tick marks

GUIDE: axis(dim(2), ticks(null()))

Applies To

axis Guide Type (GPL)

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

Figure 2-333
Example: Creating a directed acyclic graph

ELEMENT: edge(position(layout.dag(node(id), from(fromVar), to(toVar))))

Applies To

layout.circle Function (GPL), layout.dag Function (GPL), layout.data Function (GPL), layout.grid Function (GPL), layout.network Function (GPL), layout.random Function (GPL), layout.tree Function (GPL)
transparency Function (For GPL Graphic Elements)

Note: If you are modifying the transparency for a guide, refer to transparency Function (For GPL Guides) on p. 261.

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

Figure 2-334
Example: Using a variable to control transparency

ELEMENT: point(position(x*y), transparency(z))

Figure 2-335
Example: Specifying a value for transparency

ELEMENT: interval(position(x*y), transparency(transparency."0.6"))

Figure 2-336
Example: Specifying a transparency for the fill

ELEMENT: interval(position(x*y),
transparency.interior(transparency."0.8"))

Statistic Functions

See GPL Functions on p. 59.
**Applies To**
area Element (GPL), edge Element (GPL), interval Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL), line Element (GPL)

**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 p. 260.

**Syntax**

```plaintext
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**

Figure 2-337
*Example: Specifying a transparency for a reference line*

GUIDE: form.line(position(*, 2000), transparency(transparency."0.5"))

**Applies To**
form.line Guide Type (GPL)

**transpose Function**

**Syntax**

```plaintext
transpose(<coord>)
```

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

**Description**

Transposes the coordinate system.

**Examples**

Figure 2-338
*Example: Transposing a 2-D rectangular coordinate system*

COORD: transpose()
Figure 2-339
*Example: Transposing a clustered coordinate system*
COORD: transpose(rect(dim(1,2), cluster(3)))

**Coordinate Types and Transformations**
parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), mirror Function (GPL), project Function (GPL), reflect Function (GPL), wrap Function (GPL)

**Applies To**
COORD Statement (GPL), parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), project Function (GPL)

**uniform Function**

**Syntax**
uniform(<minimum>, <maximum>)

**Description**
Specifies a uniform distribution for the probability scale.

**Examples**
Figure 2-340
*Example: Specifying a uniform distribution for the probability scale*
SCALE: prob(dim(2), uniform(5000, 20000))

**Applies To**
prob Scale (GPL)

**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

Figure 2-341  Example: Adding a percent axis

GUIDE: axis(dim(2), label("Percent"), unit.percent())

Applies To

axis Guide Type (GPL)

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 PASW 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 SPSS Inc. application passes to GPL.

Examples

Figure 2-342  Example: Reading a userSource

SOURCE: mydata = userSource(id("graphdataset"))

Valid Functions

missing.listwise Function (GPL), missing.pairwise Function (GPL), weight Function (GPL)

Applies To

SOURCE Statement (GPL), csvSource Function (GPL), savSource Function (GPL), sqlSource Function (GPL)

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**

Figure 2-343
Example: Specifying the categories

SCALE: cat(dim(1), values("Male", "Female"))

**Applies To**

cat Scale (GPL)

**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**

Figure 2-344
Example: Hiding categories

SCALE: cat(aesthetic(aesthetic.visible), map("m", visible.false))
ELEMENT: line(position(salbegin*salary), visible(gender))

**Applies To**

area Element (GPL), edge Element (GPL), interval Element (GPL), line Element (GPL), path Element (GPL), point Element (GPL), polygon Element (GPL), schema Element (GPL)

**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**

Figure 2-345  
*Example: Specifying a Weibull distribution for the probability scale*

SCALE: prob(dim(2), weibull(5, 2))

**Applies To**

prob Scale (GPL)

---

**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**

Figure 2-346  
*Example: Specifying a weighted variable*

SOURCE: mydata = csvSource(file("/Data/Edge data.csv"), weight(weightedVar))
DATA: weightedVar = col(source(mydata), name="weights")

**Applies To**

csvSource Function (GPL), sqlSource Function (GPL), userSource Function (GPL)

---

**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

Figure 2-347
Example: Wrapping facets

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

Coordinate Types and Transformations

parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), mirror Function (GPL), project Function (GPL), reflect Function (GPL), transpose Function (GPL)

Applies To

COORD Statement (GPL), parallel Coordinate Type (GPL), polar Coordinate Type (GPL), polar.theta Coordinate Type (GPL), rect Coordinate Type (GPL), project Function (GPL)
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. For more information, see the topic Using the Examples in Your Application on p. 267.

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 PASW 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.

First, you need the right data source. The examples use four different userSources (Cars, Employeedata, stocks2004, and World95), which correspond to PASW Statistics SAV files located in the directory identified above.

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.
Chapter 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:

```plaintext
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 PASW Viz Designer**

- Create or open a visualization.
- If the ViZml/GPL palette is not displayed, from the menus choose:
  View
  Palettes
  ViZml/GPL
- Click the GPL tab.
- Enter the GPL into the palette. You can also copy and paste examples from the online help.
- 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:

```plaintext
SOURCE: s = csvSource(file("C:/Program Files/SPSSInc/Viz 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)))
```

- After entering and modifying the GPL, click the execute button.

**Figure 3-1**

*Execute button*

**Summary Bar Chart Examples**

This section provides examples of different types of summary bar charts.
**Simple Bar Chart**

Figure 3-2
GPL for simple bar chart

```plaintext
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 3-3
Simple bar chart

**Simple Bar Chart of Counts**

Figure 3-4
GPL for simple bar chart of counts

```plaintext
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 3-5
Simple bar chart of counts

Figure 3-6
GPL for 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)))

Simple Horizontal Bar Chart
**Simple Bar Chart With Error Bars**

Figure 3-8
GPL for 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."0.4in"), color(color.black))
Simple Bar Chart with Bar for All Categories

Figure 3-10
GPL for 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)))

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.
Stacked Bar Chart

**Figure 3-12**
GPL for stacked bar chart

```plaintext
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 3-13
Stacked bar chart

Clustered Bar Chart

Figure 3-14
GPL for 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))
Following is another option for creating a graph that appears clustered. It uses the `dodge collision modifier`. (See `dodge Collision Modifier` on p. 56.) 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”).

**Figure 3-16**
GPL for **dodged bar chart**

```plaintext
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))
```
Clustered and Stacked Bar Chart

Figure 3-18
GPL for 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"))
   label("Minority Classification"))
ELEMENT: interval.stack(position(summary.sum(gender*salary*jobcat)),
   color.exterior(color.black),
   color.interior(gender), texture.pattern.interior(minority))
**Bar Chart Using an Evaluation Function**

Figure 3-20
GPL using an evaluation function to calculate mean percentage increase

```
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))
```
Chapter 3

Figure 3-21
GPL using an evaluation function to calculate mean percentage increase

![Chart showing mean percentage increase for different employment categories and genders.]

Figure 3-22
GPL using an evaluation function to calculate percent less than a value

```plaintext
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)))
```
**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.

**Figure 3-24**
GPL for bar chart with mapped aesthetics

```r
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("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))
```
**Faceted (Paneled) Bar Chart**

Although the following examples create bar charts, faceting is common to all graphs.

Figure 3-26

GPL for faceted bar chart

```
SOURCE: s = userSource(id("Employee data"))
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 3-27  
Faceted bar chart

Figure 3-28  
GPL for 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: 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 3-29  
Faceted bar chart with nested categories
Figure 3-30
GPL for multi-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: 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 3-31
Multi-faceted bar chart

3-D Bar Chart

Figure 3-32
GPL for 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 3-33
3-D 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"))
```
Histogram Examples

This section provides examples of different types of histograms.

Histogram

Figure 3-36
GPL for 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))))
Histogram with Distribution Curve

Figure 3-38
GPL for 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.normal(salary)))
Chapter 3

Figure 3-39
Histogram with normal curve

Figure 3-40
GPL for histogram with kernel density 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 3-41
Histogram with kernel density curve

Figure 3-42
GPL for 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))))

Percentage Histogram
Figure 3-43
Percentage histogram

Frequency Polygon

Figure 3-44
GPL for 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))))
Stacked Histogram

Figure 3-46
GPL for 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))
Faceted (Paneled) Histogram

Figure 3-48
GPL for faceted 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 3-49
Faceted histogram

Figure 3-50
GPL for population pyramid

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 3-51
Population pyramid

Cumulative Histogram

Figure 3-52
GPL for 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))))
**3-D Histogram**

Figure 3-54

*GPL for 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)))))))
```
High-Low Chart Examples

This section provides examples of different types of high-low charts.

Simple Range Bar for One Variable

Figure 3-56
GPL for 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 3-57**  
Simple range bar for one variable

**Simple Range Bar for Two Variables**

**Figure 3-58**  
GPL for simple range bar for two variables

```glsl
SOURCE: s = userSource(id("Employedata"))
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(1), label("Job Category"))
ELEMENT: interval(position(region.spread.range(jobcat*(salbegin+salary))))
```
Chapter 3

Figure 3-59
Simple range bar for two variables

Figure 3-60
GPL for high-low-close chart

High-Low-Close Chart

SOURCE: s = userSource(id("stocks2004"))
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: text.title(label("NASDAQ Composite"))
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"))
**Scatter/Dot Examples**

This section provides examples of different types of scatterplots and dot plots.

**Simple 1-D Scatterplot**

Figure 3-62
GPL for simple 1-D scatterplot

SOURCE: \( s = \text{userSource(id("Employeedata"))} \)
DATA: \( \text{salary} = \text{col(source(s), name("salary"))} \)
COORD: rect(dim(1))
GUIDE: axis(dim(1), label("Salary"))
ELEMENT: point(position(salary))
Simple 2-D Scatterplot

Figure 3-64
GPL for 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))
```
Simple 2-D Scatterplot with Fit Line

Figure 3-66
GPL for simple 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)))
Grouped Scatterplot

Figure 3-68
GPL for 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 3-69**
*Grouped scatterplot*

**Grouped Scatterplot with Convex Hull**

**Figure 3-70**
*GPL for grouped scatterplot with convex hull*

```plaintext
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))
```
**Scatterplot Matrix (SPLOM)**

**Figure 3-72**
GPL for scatterplot matrix

SOURCE: s = userSource(id("Cars"))
DATA: weight=col(source(s), name("weight"))
DATA: mpg=col(source(s), name("mpg"))
DATA: horse=col(source(s), name("horse"))
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((mpg/"MPG"+weight/"Weight"+horse/"HP")* (mpg/"MPG"+weight/"Weight"+horse/"HP")))
**Figure 3-73**
*Scatterplot Matrix*

**Bubble Plot**

*Figure 3-74*
*GPL for bubble plot*

```r
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))
```
**Bubble plot**

Figure 3-75

**G各类**

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())))

**Binned Scatterplot**

Figure 3-76

GPL for 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())))
**Binned Scatterplot with Polygons**

Figure 3-78
GPL for binned scatterplot with polygons

```r
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 3-79
Binned scatterplot with polygons

Scatterplot with Border Histograms

Figure 3-80
GPL for 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 3-81**

*Scatterplot with border histograms*

![Scatterplot with border histograms](image)

**Scatterplot with Border Boxplots**

**Figure 3-82**

*GPL for scatterplot with border boxplots*

```plaintext
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 3-83
Scatterplot with border boxplots

Figure 3-84
GPL for 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)))

Dot Plot
Figure 3-85
*Dot plot*

![Dot plot image]

Figure 3-86
*GPL for grouped dot plot*

```plaintext
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 3-87
Grouped dot plot

2-D Dot Plot

Figure 3-88
GPL for 2-D dot plot

SOURCE: s = userSource(id("Cars"))
DATA: origin = col(source(s), name("origin"), unit.category())
DATA: mpg = col(source(s), name("mpg"))
GUIDE: axis(dim(1), label("Miles Per Gallon"), delta(5.0))
GUIDE: axis(dim(2), label("Country of Origin"))
ELEMENT: point.dodge.symmetric(position(bin.dot(mpg*origin)))
Figure 3-89
2-D dot plot

Figure 3-90
GPL for alternate 2-D dot plot

SOURCE: s = userSource(id("Cars"))
DATA: origin = col(source(s), name("origin"), unit.category())
DATA: mpg = col(source(s), name("mpg"))
GUIDE: axis(dim(1), label("Miles Per Gallon"), delta(5.0))
GUIDE: axis(dim(2), label("Country of Origin"))
ELEMENT: point.dodge.symmetric(position(bin.dot(origin*mpg, dim(2))))
Figure 3-91
Alternate 2-D dot plot

Jittered Categorical Scatterplot

Figure 3-92
GPL for jittered categorical scatterplot

SOURCE: s = userSource(id("Employedata"))
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))
**Line Chart Examples**

This section provides examples of different types of line charts.

**Simple Line Chart**

**Figure 3-94**
_GPL for simple line chart_

```r
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))
```
**Simple Line Chart with Points**

*Figure 3-96
GPL for 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 3-97
Simple line chart with points

![Simple line chart with points]

**Line Chart of Date Data**

Figure 3-98
GPL for line chart of date data

```plaintext
SOURCE: s = userSource(id("stocks2004"))
DATA: Date = col(source(s), name("Date"), unit.time(), format("MM/dd/yy"))
DATA: Close = col(source(s), name("Close"))
GUIDE: text.title(label("NASDAQ Composite"))
GUIDE: axis.title(label("Beginning Salary"))
GUIDE: axis.title(label("Current Salary"))
SCALE: time(dim(1), dataMaximum())
ELEMENT: line(position(Date*Close))
```
Figure 3-99
Line chart of date data

![Line chart of date data](image)

**Line Chart With Step Interpolation**

Figure 3-100
GPL for line chart with step interpolation

```plaintext
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)))
```
**Fit Line**

Figure 3-102
GPL for linear fit line overlaid on scatterplot

```gpl
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.linear(salbegin*salary)))
```
Figure 3-103
Linear fit line overlaid on scatterplot

![Linear fit line overlaid on scatterplot](image)

**Line Chart from Equation**

Figure 3-104
GPL for 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 3-105
Line chart from equation

Figure 3-106
GPL for line chart from equation in polar coordinates

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 3-107
Line chart from equation in polar coordinates
Chapter 3

**Line Chart with Separate Scales**

**Figure 3-108**
GPL for line chart with separate scales

```plaintext
SOURCE: s = userSource(id("stocks2004"))
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"))
GUIDE: text.title(label("NASDAQ Composite"))
SCALE: time(dim(1), dataMaximum())
ELEMENT: line(position(date*(close/"Close"+volume/"Volume")))
```

**Figure 3-109**
Line chart with separate scales

![NASDAQ Composite](image)

**Line Chart in Parallel Coordinates**

The following example creates a line chart in parallel coordinates. If we didn’t use the result of the `index` function to split the line, every case would be connected, so there would be only one line in the resulting graph. The line would appear to zigzag back and forth between the last axis and the first one. The explicit transparency setting is not required. It is used so that it is easier to read the axis labels.
GPL Examples

Figure 3-10
GPL for line chart in parallel coordinates

```r
SOURCE: s = userSource(id("World95"))
DATA: lifeexpm=col(source(s), name("lifeexpm"))
DATA: lifeexpf=col(source(s), name("lifeexpf"))
DATA: babymort=col(source(s), name("babymort"))
DATA: literacy=col(source(s), name("literacy"))
DATA: region=col(source(s), name("region"), unit.category())
TRANS: caseid = index()
COORD: parallel()
GUIDE: axis(dim(1), label("Life Expectancy (Males)"))
GUIDE: axis(dim(2), label("Life Expectancy (Females)"))
GUIDE: axis(dim(3), label("Literacy"))
GUIDE: axis(dim(4), label("Infant Mortality"))
GUIDE: legend(aesthetic(aesthetic.color), label("Region"))
ELEMENT: line(position(lifeexpm*lifeexpf*literacy*babymort),
             split(caseid), color(region),
             transparency(transparency."0.5"))
```

Figure 3-11
Line chart in parallel coordinates

Here is the same chart plotted in polar coordinates. We also reversed the scale for dimension 4 to emphasize the relationship among the variables. As you can see, there are distinct bands of color representing the different regions. Also note the `preserveStraightLines` and `closed` functions, which improve the look of the graph in polar coordinates.
Figure 3-112  
GPL for line chart in polar parallel coordinates

```
SOURCE: s = userSource(id("World95"))
DATA: lifeexpm=col(source(s), name("lifeexpm"))
DATA: lifeexpf=col(source(s), name("lifeexpf"))
DATA: babymort=col(source(s), name("babymort"))
DATA: literacy=col(source(s), name("literacy"))
DATA: region=col(source(s), name("region"), unit.category())
TRANS: caseid = index()
COORD: polar(parallel())
SCALE: linear(dim(4), reverse())
GUIDE: axis(dim(1), label("Life Expectancy (Males)"))
GUIDE: axis(dim(2), label("Life Expectancy (Females)"))
GUIDE: axis(dim(3), label("Literacy"))
GUIDE: axis(dim(4), label("Infant Mortality"))
GUIDE: legend(aesthetic(aesthetic.color), label("Region"))
ELEMENT: line(position(lifeexpm*lifeexpf*literacy*babymort),
                  split(caseid), color(region),
                  transparency(transparency."0.5"),
                  preserveStraightLines(), closed())
```

Figure 3-113  
Line chart in polar parallel coordinates

```
Region
- Africa
- East Europe
- Latin America
- Middle East
- OECD
- Pacific Asia
```

Pie Chart Examples

This section provides examples of different types of pie charts.

Pie Chart

Figure 3-114  
GPL for 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 3-115
Pie chart

![Pie chart](image)

### Figure 3-116
GPL for pie chart with labels

```plaintext
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 3-117
Pie chart with labels

![Pie chart](image)
Chapter 3

Paneled Pie Chart

Figure 3-18
GPL for 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 3-19
Paneled pie chart

Stacked Pie Chart

Figure 3-120
GPL for 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%'))
Boxplot Examples

This section provides examples of different types of box plots.

1-D Boxplot

Figure 3-122
GPL for 1-D box plot

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 3-123
1-D boxplot

![Boxplot](image)

Boxplot

Figure 3-124
GPL for boxplot

```glsl
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 3-125
Boxplot

Figure 3-126
GPL for boxplot with labeled outliers

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))
Clustered Boxplot

Figure 3-128
GPL for 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(3), label("Job Category"))
ELEMENT: schema(position(bin.quantile.letter(gender*salary*jobcat)), color(gender))
Figure 3-129
Clustered boxplot

Figure 3-130
GPL for boxplot with overlaid dot plot

SOURCE: s = userSource(id("World95"))
DATA: region = col(source(s), name("region"), unit.category())
DATA: birth = col(source(s), name("birth_rt"))
GUIDE: axis(dim(2), label("Birth Rate"))
GUIDE: axis(dim(1), label("Region"))
SCALE: linear(dim(2), include(0))
ELEMENT: schema(position(bin.quantile.letter(region*birth)))
ELEMENT: point.dodge.symmetric(position(bin.dot(region*birt, dim(2))),
color(color.red))
Multi-Graph Examples

This section provides examples of multiple graphs in the same page display.

Scatterplot with Border Histograms

Figure 3-132
GPL for 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%, 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 3-133
Scatterplot with border histograms

Figure 3-134
GPL for 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()
Chapter 3

Figure 3-135
Scatterplot with border boxplots

Figure 3-136
Stocks Line Chart with Volume Bar Chart

```plaintext
SOURCE: s = userSource(id("stocks2004"))
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()
```
**Dual Axis Graph**

Figure 3-138
GPL for dual axis graph

```plaintext
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 3-139
Dual axis graph

Histogram with Dot Plot

Figure 3-140
GPL for 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(2), ticks(null()))
ELEMENT: point.dodge.asymmetric(position(bin.dot(salary)))
GRAPH: end()
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.

Figure 3-142
GPL for collapsing small categories

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))
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.

Figure 3-144
GPL for mapping aesthetics

```r
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("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))
```
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. For more information, see the topic Line Chart with Separate Scales on p. 320.

```
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")))
```
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.

**Figure 3-148**

*GPL for grouping by separate variables*

```r
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"))
```
**Clustering Separate Variables**

This example demonstrates how you can cluster separate variables.

**Figure 3-150**
GPL for clustering separate variables

```plaintext
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 3-151
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.

Figure 3-152
GPL for binning 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())))
Categorical Heat Map

Figure 3-153
Binning over categorical values

```
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)))
```
Chapter 3

Figure 3-155
Categorical heat map

Figure 3-156
GPL for alternate 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 3-157
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.

**Figure 3-158**
GPL for creating categories with the eval function

```
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 3-159**
Creating categories with the eval function
## Appendices

### GPL Constants

#### Color Constants

<table>
<thead>
<tr>
<th>Color</th>
<th>Color</th>
<th>Color</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>aliceblue</td>
<td>aqua</td>
<td>azure</td>
<td>bisque</td>
</tr>
<tr>
<td>black</td>
<td>blanchedalmond</td>
<td>blue</td>
<td>blueviolet</td>
</tr>
<tr>
<td>brown</td>
<td>burlywood</td>
<td>cadetblue</td>
<td>chartreuse</td>
</tr>
<tr>
<td>chocolate</td>
<td>coral</td>
<td>cornflowerblue</td>
<td>cornsilk</td>
</tr>
<tr>
<td>crimson</td>
<td>cyan</td>
<td>darkblue</td>
<td>darkcyan</td>
</tr>
<tr>
<td>darkgoldenrod</td>
<td>darkgray</td>
<td>darkgreen</td>
<td>darkgrey</td>
</tr>
<tr>
<td>darkkhaki</td>
<td>darkmagenta</td>
<td>darkolivegreen</td>
<td>darkorange</td>
</tr>
<tr>
<td>darkorchid</td>
<td>darkred</td>
<td>darksalmon</td>
<td>darkseagreen</td>
</tr>
<tr>
<td>darkslateblue</td>
<td>darkslategray</td>
<td>darkslategrey</td>
<td>darkturquoise</td>
</tr>
<tr>
<td>darkviolet</td>
<td>deeppink</td>
<td>deepskyblue</td>
<td>dimgray</td>
</tr>
<tr>
<td>dimgray</td>
<td>dodgerblue</td>
<td>firebrick</td>
<td>floralwhite</td>
</tr>
<tr>
<td>forestgreen</td>
<td>fuchsia</td>
<td>gainsboro</td>
<td>ghostwhite</td>
</tr>
<tr>
<td>gold</td>
<td>goldenrod</td>
<td>gray</td>
<td>grey</td>
</tr>
<tr>
<td>green</td>
<td>greenyellow</td>
<td>honeydew</td>
<td>hotpink</td>
</tr>
<tr>
<td>indianred</td>
<td>indigo</td>
<td>ivory</td>
<td>khaki</td>
</tr>
<tr>
<td>lavender</td>
<td>lavenderblush</td>
<td>lawngreen</td>
<td>lemonchiffon</td>
</tr>
<tr>
<td>lightblue</td>
<td>lightcoral</td>
<td>lightcyan</td>
<td>lightgoldenrodyellow</td>
</tr>
<tr>
<td>lightgray</td>
<td>lightgreen</td>
<td>lightgrey</td>
<td>lightpink</td>
</tr>
<tr>
<td>lightsalmon</td>
<td>lightseagreen</td>
<td>lightskyblue</td>
<td>lightslategrey</td>
</tr>
<tr>
<td>lightslategrey</td>
<td>lightsteelblue</td>
<td>lightyellow</td>
<td>lime</td>
</tr>
<tr>
<td>limegreen</td>
<td>linen</td>
<td>magenta</td>
<td>maroon</td>
</tr>
<tr>
<td>mediumaquamarine</td>
<td>mediumblue</td>
<td>mediumorchid</td>
<td>mediumpurple</td>
</tr>
<tr>
<td>mediumseagreen</td>
<td>mediumslateblue</td>
<td>mediumspringgreen</td>
<td>mediumturquoise</td>
</tr>
<tr>
<td>mediumvioletred</td>
<td>midnightblue</td>
<td>mintcream</td>
<td>mistyrose</td>
</tr>
<tr>
<td>moccasin</td>
<td>navajowhite</td>
<td>navy</td>
<td>oldlace</td>
</tr>
<tr>
<td>olive</td>
<td>olivedrab</td>
<td>orange</td>
<td>orangered</td>
</tr>
<tr>
<td>orchid</td>
<td>palegoldenrod</td>
<td>palesgreen</td>
<td>paleturquoise</td>
</tr>
<tr>
<td>palevioletred</td>
<td>papayawhip</td>
<td>peachpuff</td>
<td>peru</td>
</tr>
<tr>
<td>pink</td>
<td>plum</td>
<td>powderblue</td>
<td>purple</td>
</tr>
<tr>
<td>red</td>
<td>rosybrown</td>
<td>royalblue</td>
<td>saddlebrown</td>
</tr>
<tr>
<td>salmon</td>
<td>sandybrown</td>
<td>seagreen</td>
<td>seashell</td>
</tr>
<tr>
<td>sienna</td>
<td>silver</td>
<td>skyblue</td>
<td>slateblue</td>
</tr>
<tr>
<td>slategray</td>
<td>slategrey</td>
<td>snow</td>
<td>springgreen</td>
</tr>
<tr>
<td>steelblue</td>
<td>tan</td>
<td>teal</td>
<td>thistle</td>
</tr>
</tbody>
</table>
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.

Table A-1
shape.interior constants for interval elements

<table>
<thead>
<tr>
<th>ibeam</th>
<th>line</th>
<th>square</th>
</tr>
</thead>
</table>

Table A-2
shape.interior constants for edge and point elements

<table>
<thead>
<tr>
<th>arrow</th>
<th>bowtie</th>
<th>circle</th>
<th>cross</th>
</tr>
</thead>
<tbody>
<tr>
<td>decagon</td>
<td>elbow</td>
<td>elbowArrow</td>
<td>female</td>
</tr>
<tr>
<td>flower</td>
<td>flower3</td>
<td>flower4</td>
<td>flower5</td>
</tr>
<tr>
<td>flower6</td>
<td>flower7</td>
<td>flower8</td>
<td>flower9</td>
</tr>
<tr>
<td>flower10</td>
<td>heptagon</td>
<td>hexagon</td>
<td>hollowBowtie</td>
</tr>
<tr>
<td>hollowCircle</td>
<td>hollowDecagon</td>
<td>hollowHeptagon</td>
<td>hollowHexagon</td>
</tr>
<tr>
<td>hollowNonagon</td>
<td>hollowOctagon</td>
<td>hollowPentagon</td>
<td>hollowPolygon</td>
</tr>
<tr>
<td>hollowRoundRectangle</td>
<td>hollowSquare</td>
<td>hollowStar</td>
<td>hollowStar3</td>
</tr>
<tr>
<td>hollowStar4</td>
<td>hollowStar5</td>
<td>hollowStar6</td>
<td>hollowStar7</td>
</tr>
<tr>
<td>hollowStar8</td>
<td>hollowStar9</td>
<td>hollowStar10</td>
<td>hollowTriangle</td>
</tr>
<tr>
<td>ibeam</td>
<td>line</td>
<td>male</td>
<td>nonagon</td>
</tr>
<tr>
<td>octagon</td>
<td>pentagon</td>
<td>plus</td>
<td>polygon</td>
</tr>
<tr>
<td>roundRectangle</td>
<td>square</td>
<td>star</td>
<td>star3</td>
</tr>
<tr>
<td>star4</td>
<td>star5</td>
<td>star6</td>
<td>star7</td>
</tr>
<tr>
<td>star8</td>
<td>star9</td>
<td>star10</td>
<td>triangle</td>
</tr>
</tbody>
</table>

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.

Table A-3
shape.interior constants for edge, line, and path elements

<table>
<thead>
<tr>
<th>dash</th>
<th>dash_1_dot</th>
<th>dash_2_dots</th>
<th>dash_3_dots</th>
</tr>
</thead>
<tbody>
<tr>
<td>dash_2x</td>
<td>dash_3x</td>
<td>dash_4_dots</td>
<td>dash_dash2x</td>
</tr>
<tr>
<td>half_dash</td>
<td>solid</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Size Constants

<table>
<thead>
<tr>
<th>tiny</th>
<th>small</th>
<th>medium</th>
<th>large</th>
</tr>
</thead>
<tbody>
<tr>
<td>huge</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Pattern Constants

<table>
<thead>
<tr>
<th>checkered</th>
<th>grid</th>
<th>grid2</th>
<th>grid3</th>
</tr>
</thead>
<tbody>
<tr>
<td>grid4</td>
<td>grid5</td>
<td>mesh</td>
<td>mesh2</td>
</tr>
<tr>
<td>mesh3</td>
<td>mesh4</td>
<td>mesh5</td>
<td>pluses</td>
</tr>
<tr>
<td>pluses2</td>
<td>solid</td>
<td>stripes</td>
<td>stripes2</td>
</tr>
</tbody>
</table>
Index

aestheticMaximum function (GPL), 61
aestheticMinimum function (GPL), 62
aestheticMissing function (GPL), 63
algebra (GPL), 4
  rules, 4
alpha function (GPL), 64
analysis variable (GPL), 5
area element (GPL), 48
asn scale (GPL), 31
atanh scale (GPL), 32
axis guide (GPL), 42
bar element (GPL), 49
base function (GPL), 64
base.aesthetic function (GPL), 65
base.all function (GPL), 65
base.coordinate function (GPL), 66
begin function (GPL), 67–68
beta function (GPL), 68
bin.dot function (GPL), 69
bin.hex function (GPL), 70
bin.quantile.letter function (GPL), 72
bin.rect function (GPL), 73
binStart function (GPL), 76
binWidth function (GPL), 77
blend operator (GPL), 4
boxplot (GPL), 55

cat (categorical) scale (GPL), 33
chiSquare function (GPL), 77
closed function (GPL), 78
cluster function (GPL), 78
col function (GPL), 80
collapse function (GPL), 81
collision modifiers (GPL)
  difference, 55
dodge, 56
dodge.asymmetric, 57
dodge.symmetric, 57
  jitter, 58
  stack, 58
color function (GPL), 82–83
color.brightness function (GPL), 84–85
color.hue function (GPL), 86–87
color.saturation function (GPL), 87–88
COMMENT statement (GPL), 22
constants (GPL)
  color, 344
  pattern, 346
  shape, 345
  size, 345
  user, 6
COORD statement (GPL), 25
coordinate types (GPL)
  parallel, 26
  polar, 27
  polar.theta, 28
  project, 168
  rect (rectangular), 29
cordinates (GPL), 7
cross operator (GPL), 4
csvSource function (GPL), 89

dataMaximum function (GPL), 24
dataMinimum function (GPL), 89
delta function (GPL), 90
density.beta function (GPL), 91
density.chiSquare function (GPL), 92
density.exponential function (GPL), 94
density.f function (GPL), 95
density.gamma function (GPL), 96
density.kernel function (GPL), 98
density.logistic function (GPL), 100
density.normal function (GPL), 101
density.poisson function (GPL), 103
density.studentizedRange function (GPL), 104
density.t function (GPL), 105
density.uniform function (GPL), 107
density.weibull function (GPL), 108
difference collision modifier (GPL), 55
dim function (GPL), 109
dodge collision modifier (GPL), 56
dodge.asymmetric collision modifier (GPL), 57
dodge.symmetric collision modifier (GPL), 57

difference collision modifier (GPL), 55
edge element (GPL), 50
ELEMENT statement (GPL), 47
element types (GPL)
  area, 48
  bar, 49
  edge, 50
  interval, 50
  line, 51
  path, 52
  point, 53
  polygon, 54
  schema, 55
dend function (GPL), 111
eval function (GPL), 112
exclude function (GPL), 116

348
exponent function (GPL), 116
exponential function (GPL), 117
f function (GPL), 117
footnote (GPL), 44–45
form.line guide (GPL), 43
format function (GPL), 117
format.date function (GPL), 118
format.dateTime function (GPL), 119
format.time function (GPL), 119
from function (GPL), 119
functions (GPL)
aestheticMaximum, 61
aestheticMinimum, 62
aestheticMissing, 63
alpha, 64
base, 64
base.aesthetic, 65
base.all, 65
base.coordinate, 66
begin, 67–68
beta, 68
bin.dot, 69
bin.hex, 70
bin.quantile.letter, 72
bin.rect, 73
binCount, 75
binStart, 76
binWidth, 77
chiSquare, 77
closed, 78
cluster, 78
col, 80
collapse, 81
color, 82–83
color.brightness, 84–85
color.hue, 86–87
color.saturation, 87–88
csvSource, 89
dataMaximum, 89
dataMinimum, 90
delta, 90
density.beta, 91
density.chiSquare, 92
density.exponential, 94
density.f, 95
density.gamma, 96
density.kernel, 98
density.logistic, 100
density.normal, 101
density.poisson, 103
density.studentizedRange, 104
density.t, 105
density.uniform, 107
density.weibull, 108
dim, 109
end, 111
eval, 112
exclude, 116
exponent, 116
exponential, 117
f, 117
format, 117
format.date, 118
format.dateTime, 119
format.time, 119
from, 119
gamma, 120
gap, 120
gridlines, 121
in, 121
include, 122
index, 123
iter, 123
jump, 124
label, 124–125
layout.circle, 126
layout.dag, 127
layout.data, 129
layout.grid, 130
layout.network, 132
layout.random, 133
layout.tree, 135
link.alpha, 136
link.complete, 138
link.delaunay, 139
link.distance, 141
link.gabriel, 142
link.hull, 144
link.influence, 145
link.join, 147
link.mst, 148
link.neighbor, 150
link.relativeNeighborhood, 151
link.sequence, 153
link.tsp, 154
logistic, 156
map, 156
marron, 157
max, 157
min, 158
mirror, 158
missing.gap, 159
missing.interpulate, 159
missing.listwise, 160
missing.pairwise, 160
missing.wings, 161
multiple, 161
noConstant, 162
node, 162
normal, 163
notIn, 163
opposite, 164
origin, 164–165
Index

poisson, 165
position, 166–167
preserveStraightLines, 168
proportion, 169
reflect, 169
region.confint.count, 170
region.confint.mean, 171
region.confint.percent.count, 173
region.confint.proportion.count, 174
region.confint.smooth, 176
region.spread.range, 177
region.spread.sd, 178
region.spread.se, 180
reverse, 182
root, 183
savSource, 184
scale, 184–186
segments, 187
shape, 187–188
showAll, 189
size, 189–190
smooth.cubic, 191
smooth.linear, 193
smooth.loess, 194
smooth.mean, 196
smooth.median, 198
smooth.quadratic, 199
smooth.spline, 201
smooth.step, 202
sort.data, 203
sort.natural, 204
sort.statistic, 204
sort.values, 205
split, 205
sqlSource, 206
start, 206
startAngle, 207
studentizedRange, 207
summary.count, 208
summary.count.cumulative, 210
summary.countTrue, 211
summary.first, 213
summary.kurtosis, 214
summary.last, 216
summary.max, 217
summary.mean, 219
summary.median, 220
summary.min, 222
summary.mode, 223
summary.percent, 225
summary.percent.count, 226
summary.percent.count.cumulative, 227
summary.percent.cumulative, 229
summary.percent.sum, 229
summary.percent.sum.cumulative, 231
summary.percentile, 233
summary.percentTrue, 234
summary.proportion, 236
summary.proportion.count, 237
summary.proportion.count.cumulative, 238
summary.proportion.cumulative, 240
summary.proportion.sum, 240
summary.proportion.sum.cumulative, 242
summary.proportionTrue, 244
summary.range, 245
summary.sd, 247
summary.se, 248
summary.se.kurtosis, 250
summary.se.skewness, 251
summary.sum, 253
summary.sum.cumulative, 254
summary.variance, 256
t, 257
texture.pattern, 258
tick, 259
to, 259
transparency, 260–261
transpose, 261
uniform, 262
unit.percent, 262
userSource, 263
values, 263
visible, 264
weibull, 264
weight, 265
wrap, 265
gamma function (GPL), 120
gap function (GPL), 120
GPL
algebra, 4
algebra and coordinate interaction, 7
algebra rules, 4
analysis variable, 5
blend operator, 4
classification, 16
color constants, 344
color, 344
constants, pattern, 346
constants, shape, 345
constants, size, 345
cross operator, 4
data sources for examples, 267
example data sources, 267
faceting, 15
introduction, 1
nest operator, 4
operator precedence, 5
operators, 4
paneling, 15
pattern constants, 346
shape constants, 345
size constants, 345
stacking, 13
syntax rules, 3
unity variable, 6
user constants, 6
using aesthetics, 18
GRAPH statement (GPL), 23
graphic element types (GPL)
area, 48
bar, 49
data, 50
division, 50
line, 51
path, 52
point, 53
polygon, 54
schema, 55
graphic elements (GPL), 47
gridlines function (GPL), 121
GUIDE statement (GPL), 41
guide types (GPL)
axis, 42
form.line, 43
legend, 44
line, 43
text.footnote, 44
textGlyph, 45
textGlyph, 45
textGlyph, 46
textGlyph, 46
textGlyph, 47
in function (GPL), 121
include function (GPL), 122
index function (GPL), 123
interval element (GPL), 50
iter function (GPL), 123

jitter collision modifier (GPL), 58
jump function (GPL), 124
label function (GPL), 124–125
layout.circle function (GPL), 126
layout.dag function (GPL), 127
layout.data function (GPL), 129
layout.grid function (GPL), 130
layout.network function (GPL), 132
layout.random function (GPL), 133
layout.tree function (GPL), 135
legend guide (GPL), 44
line element (GPL), 51
line guide (GPL), 43
linear scale (GPL), 34
link.alpha function (GPL), 136
link.complete function (GPL), 138
link.delaunay function (GPL), 139
link.distance function (GPL), 141
link.gabriel function (GPL), 142
link.hull function (GPL), 144
link.influence function (GPL), 145
link.join function (GPL), 147
link.mst function (GPL), 148
link.neighbor function (GPL), 150
link.relativeNeighborhood function (GPL), 151
link.sequence function (GPL), 153
link.tsp function (GPL), 154
log scale (GPL), 34
logistic function (GPL), 156
logit scale (GPL), 35

map function (GPL), 156
marron function (GPL), 157
max function (GPL), 157
min function (GPL), 158
mirror function (GPL), 158
missing.gap function (GPL), 159
missing.interpolate function (GPL), 159
missing.listwise function (GPL), 160
missing.pairwise function (GPL), 160
missing.wings function (GPL), 161
multiple function (GPL), 161
nest operator (GPL), 4
noConstant function (GPL), 162
node function (GPL), 162
normal function (GPL), 163
notIn function (GPL), 163

operator precedence (GPL), 5
operators (GPL), 4
opposite function (GPL), 164
origin function (GPL), 164–165

PAGE statement (GPL), 22
parallel coordinates (GPL), 26
path element (GPL), 52
point element (GPL), 53
poisson function (GPL), 165
polar coordinates (GPL), 27
polar.theta coordinates (GPL), 28
polygon element (GPL), 54
position function (GPL), 166–167
pow scale (GPL), 36
preserveStraightLines function (GPL), 168
prob scale (GPL), 37
probit scale (GPL), 38
project coordinates (GPL), 168
proportion function (GPL), 169
rect (rectangular) coordinates (GPL), 29
reference line (GPL), 43
reflect function (GPL), 169
region.conf.int function (GPL), 170
region.conf.mean function (GPL), 171
region.conf.percent.count function (GPL), 173
region.conf.proportion.count function (GPL), 174
region.conf.smooth function (GPL), 176
region.spread.range function (GPL), 177
region.spread.sd function (GPL), 178
region.spread.se function (GPL), 180
reverse function (GPL), 182
root function (GPL), 183

safeLog scale (GPL), 38
safePower scale (GPL), 39
savSource function (GPL), 184
scale function (GPL), 184–186
SCALE statement (GPL), 30
scale types (GPL), 31
asn, 31
atanh, 32
cat (categorical), 33
linear, 34
log, 34
logit, 35
pow, 36
prob, 37
probit, 38
safeLog, 38
safePower, 39
time, 40
schema element (GPL), 55
segments function (GPL), 187
shape function (GPL), 187–188
showAll function (GPL), 189
size function (GPL), 189–190
smooth.cubic function (GPL), 191
smooth.linear function (GPL), 193
smooth.loess function (GPL), 194
smooth.mean function (GPL), 196
smooth.median function (GPL), 198
smooth.quadratic function (GPL), 199
smooth.spline function (GPL), 201
smooth.step function (GPL), 202
sort.data function (GPL), 203
sort.natural function (GPL), 204
sort.statistic function (GPL), 204
sort.values function (GPL), 205
SOURCE statement (GPL), 24
split function (GPL), 205
sqlSource function (GPL), 206
stack collision modifier (GPL), 58
start function (GPL), 206
startAngle function (GPL), 207
statements (GPL), 21
COMMENT, 22
COORD, 25
DATA, 24
ELEMENT, 47
GRAPH, 23

GUIDE, 41
PAGE, 22
SCALE, 30
SOURCE, 24
TRANS, 25
studentizedRange function (GPL), 207
subfootnote guide (GPL), 45
subsubfootnote guide (GPL), 45
subtitle guide (GPL), 46
subtitle guide (GPL), 46
summary.count function (GPL), 208
summary.count.cumulative function (GPL), 210
summary.countTrue function (GPL), 211
summary.first function (GPL), 213
summary.kurtosis function (GPL), 214
summary.last function (GPL), 216
summary.max function (GPL), 217
summary.mean function (GPL), 219
summary.median function (GPL), 220
summary.min function (GPL), 222
summary.mode function (GPL), 223
summary.percent function (GPL), 225
summary.percent.count function (GPL), 226
summary.percent.cumulative function (GPL), 227
summary.percent.sum function (GPL), 229
summary.percent.sum.cumulative function (GPL), 231
summary.percentTrue function (GPL), 234
summary.proportion function (GPL), 236
summary.proportion.count function (GPL), 237
summary.proportion.cumulative function (GPL), 238
summary.proportion.sum function (GPL), 240
summary.proportion.sum.cumulative function (GPL), 242
summary.proportionTrue function (GPL), 244
summary.range function (GPL), 245
summary.sd function (GPL), 247
summary.se function (GPL), 248
summary.se.kurtosis function (GPL), 250
summary.se.skewness function (GPL), 251
summary.sum function (GPL), 253–254
summary.variance function (GPL), 256
syntax rules
GPL, 3
t function (GPL), 257
text.footnote guide (GPL), 44
text.subfootnote guide (GPL), 45
text.subsubfootnote guide (GPL), 45
text.subtitle guide (GPL), 46
text.title guide (GPL), 47
texture.pattern function (GPL), 258
ticks function (GPL), 259
time scale (GPL), 40
title 2 (GPL), 46
title 3 (GPL), 46
title guide (GPL), 47
to function (GPL), 259
TRANS statement (GPL), 25
transparency function (GPL), 260–261
transpose function (GPL), 261
uniform function (GPL), 262
unit.percent function (GPL), 262
unity variable (GPL), 6
user constants (GPL), 6
userSource function (GPL), 263
values function (GPL), 263
visible function (GPL), 264
weibull function (GPL), 264
weight function (GPL), 265
wrap function (GPL), 265