Package: rsgl (via r-universe)

October 9, 2026

Title An Implementation of the 'SGL' Graphics Language

Version 0.1.0.9000

Description Generates plots from a database connection and a 'SGL' statement. 'SGL' is a graphics language designed to look and feel like 'SQL'. It is especially useful for those familiar with 'SQL' who want to specify plots in a similar manner. The 'SGL' language is described in Chapman (2025) <doi:10.48550/arXiv.2505.14690>.

License MIT + file LICENSE

URL https://github.com/sgl-projects/rsgl, https://sgl-projects.github.io/rsgl/, https://arxiv.org/abs/2505.14690

BugReports https://github.com/sgl-projects/rsgl/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0), vdiffr, lubridate, knitr, rmarkdown, patrick, tibble, withr

Config/testthat/edition 3

Imports DBI, dplyr, duckdb, ggplot2, purrr, Rcpp, rlang

LinkingTo Rcpp

VignetteBuilder knitr, rmarkdown

Depends R (>= 4.1)

LazyData true

Config/pak/sysreqs xz-utils

Repository https://sgl-projects.r-universe.dev

Date/Publication 2026-08-10 15:52:39 UTC

RemoteUrl https://github.com/sgl-projects/rsgl

RemoteRef HEAD

RemoteSha 46829965e47f0964b725c7becb9647512a3c4e93

Contents

avg . . . 2
bars . . . 3
bin . . . 4
boxes . . . 5
cars . . . 6
count . . . 6
dbGetPlot . . . 7
linear . . . 8
lines . . . 9
ln . . . 10
log . . . 11
points . . . 11
trees . . . 12
type_classifications . . . 13
violins . . . 14

Index **15**

---

avg *Average*

Description

The avg function returns the average of a column within each group.

Function Name

- avg

Arguments

- The name of a numerical column to average (required).

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)

dbGetPlot(con, "
  visualize
    origin as x,
    avg(horsepower) as y
  from cars
  group by
    origin
  using bars
```

```
")
```

---

bars *Bars*

Description

Documents the aliases, aesthetics, and qualifiers for the bar geom.

Aliases

- bar
- bars

Aesthetics

- x
- y
- theta
- r
- color

Qualifiers

- horizontal: orients the bars horizontally.
- unstacked: doesn't stack overlapping bars.
- vertical: orients the bars vertically.

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)
dbGetPlot(con, "
  visualize
    bin(miles_per_gallon) as x,
    count(*) as y
  from cars
  group by
    bin(miles_per_gallon)
  using bars
")

dbGetPlot(con, "
  visualize
    bin(miles_per_gallon) as y,
```

```
    count(*) as x
  from cars
  group by
    bin(miles_per_gallon)
  using horizontal bars
")
```

---

bin *Bin*

Description

The bin function creates equal-width bins for a column; original values are transformed into bin-center values.

Function Name

- bin

Arguments

- The name of a numerical column to bin (required).
- The number of bins (optional, default 30).

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)

dbGetPlot(con, "
  visualize
    bin(miles_per_gallon) as x,
    count(*) as y
  from cars
  group by
    bin(miles_per_gallon)
  using bars
")

dbGetPlot(con, "
  visualize
    bin(miles_per_gallon, 10) as x,
    count(*) as y
  from cars
  group by
    bin(miles_per_gallon, 10)
  using bars
")
```

boxes *Boxes*

Description

Documents the aliases, aesthetics, and qualifiers for the box geom.

Aliases

- box
- boxes

Aesthetics

- x
- y
- theta
- r
- color

Qualifiers

- horizontal: orients the boxes horizontally.
- vertical: orients the boxes vertically.

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)
dbGetPlot(con, "
  visualize
    origin as x,
    miles_per_gallon as y
  from cars
  using boxes
")

dbGetPlot(con, "
  visualize
    origin as y,
    miles_per_gallon as x
  from cars
  using horizontal boxes
")
```

cars *Cars dataset*

Description

A sample dataset used throughout rsgl documentation and examples containing attributes for a collection of cars. Loading rsgl masks datasets::cars; refer to the built-in version as `datasets::cars` if you need it.

Usage

```
cars
```

Format

A data frame with 406 rows and 5 variables:

car_id Integer identifier for the car.

horsepower Engine horsepower.

miles_per_gallon Fuel economy in miles per US gallon.

origin Country of origin (`USA`, `Europe`, `Japan`).

year Model year (1970–1982).

Source

Derived from the cars dataset in the vega-datasets collection (https://github.com/vega/vega-datasets/blob/main/data/cars.json). The `Miles_per_Gallon`, `Horsepower`, and `Origin` columns were kept and renamed, `year` is the year taken from the original `Year` date, and `car_id` is a row identifier.

count *Count*

Description

The `count` function returns the number of rows in each group.

Function Name

- `count`

Arguments

- `*` (required).

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)

dbGetPlot(con, "
  visualize
    origin as x,
    count(*) as y
  from cars
  group by
    origin
  using bars
")
```

`dbGetPlot` *Generate a plot from a SGL statement*

Description

`dbGetPlot` takes a database connection and a SGL statement and returns the corresponding plot.

Usage

```
dbGetPlot(con, sgl_stmt)
```

Arguments

`con`	A database connection (as returned by `DBI::dbConnect()`)
`sgl_stmt`	A SGL statement (string)

Value

The plot defined by the SGL statement (a `sgl_plot` object)

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)
p <- dbGetPlot(con, "
  visualize
    horsepower as x,
    miles_per_gallon as y
  from cars
  using points
")
print(p)
```

`linear` *Linear*

Description

The `linear` scale linearly maps data values to a visual property. It is the default scale for numerical aesthetic mappings.

Function Name

- `linear`

Arguments

- The name of an aesthetic to scale (required).

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)

# explicit linear scales
dbGetPlot(con, "
  visualize
    horsepower as x,
    miles_per_gallon as y
  from cars
  using points
  scale by
    linear(x),
    linear(y)
")

# default linear scales
dbGetPlot(con, "
  visualize
    horsepower as x,
    miles_per_gallon as y
  from cars
  using points
")
```

lines *Lines*

Description

Documents the aliases, aesthetics, and qualifiers for the line geom.

Aliases

- line
- lines

Aesthetics

- x
- y
- theta
- r
- color

Qualifiers

- horizontal: orients the line horizontally; points are connected in order of increasing x/theta values.
- regression: fits a linear regression line to the data.
- vertical: orients the line vertically; points are connected in order of increasing y/r values.

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "trees", trees)
dbGetPlot(con, "
  visualize
    age as x,
    circumference as y
  from trees
  collect by
    tree_id
  using lines
")

dbWriteTable(con, "cars", cars)
dbGetPlot(con, "
  visualize
    horsepower as x,
    miles_per_gallon as y
```

```
  from cars
  using (
     points
     layer
     regression line
  )
   scale by
     log(x),
     log(y)
")
```

---

ln *Natural Log*

Description

The ln scale maps data values to a visual property through a natural logarithm transformation.

Function Name

- ln

Arguments

- The name of an aesthetic to scale (required).

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)

dbGetPlot(con, "
  visualize
    horsepower as x,
    miles_per_gallon as y
  from cars
  using points
  scale by
    ln(x),
    ln(y)
")
```

log *Log*

Description

The log scale maps data values to a visual property through a base-10 logarithm transformation.

Function Name

- log

Arguments

- The name of an aesthetic to scale (required).

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)

dbGetPlot(con, "
  visualize
    horsepower as x,
    miles_per_gallon as y
  from cars
  using points
  scale by
    log(x),
    log(y)
")
```

points *Points*

Description

Documents the aliases, aesthetics, and qualifiers for the point geom.

Aliases

- point
- points

Aesthetics

- `x`
- `y`
- `theta`
- `r`
- `color`
- `size`

Qualifiers

- `jittered`: adds a small amount of random noise to each point's position.

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)
dbGetPlot(con, "
  visualize
    horsepower as x,
    miles_per_gallon as y
  from cars
  using points
")

set.seed(0)
dbGetPlot(con, "
  visualize
    origin as x,
    miles_per_gallon as y
  from cars
  using jittered points
")
```

`trees` *Trees dataset*

Description

A dataset used throughout rsgl documentation and examples containing repeated measurements of tree growth over time. Loading rsgl masks datasets::trees; refer to the built-in version as `datasets::trees` if you need it.

Usage

```
trees
```

Format

A data frame with 35 rows and 3 variables:

tree_id Integer identifier for the tree.

age Age of the tree at the time of measurement.

circumference Trunk circumference at the time of measurement.

Source

Derived from datasets::Orange (R Core Team), with the `Tree` column renamed to `tree_id`.

`type_classifications` *Get SGL type classifications for columns in a table*

Description

`type_classifications` takes a database connection and a table name and returns the SGL type classifications (numerical, categorical, or temporal) of the table's columns.

Usage

```
type_classifications(con, table_name)
```

Arguments

`con`	A database connection (as returned by `DBI::dbConnect()`)
`table_name`	The name of a table

Value

A dataframe listing the SGL type classification of each column.

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "iris", iris)
type_classifications(con, "iris")
```

violins *Violins*

Description

Documents the aliases, aesthetics, and qualifiers for the violin geom.

Aliases

- violin
- violins

Aesthetics

- x
- y
- theta
- r
- color

Qualifiers

- horizontal: orients the violins horizontally.
- vertical: orients the violins vertically.

Examples

```
library(duckdb)
con <- dbConnect(duckdb())
dbWriteTable(con, "cars", cars)
dbGetPlot(con, "
  visualize
    origin as x,
    miles_per_gallon as y
  from cars
  using violins
")

dbGetPlot(con, "
  visualize
    origin as y,
    miles_per_gallon as x
  from cars
  using horizontal violins
")
```

Index

∗ **datasets**
    cars, 6
    trees, 12

avg, 2

bars, 3
bin, 4
boxes, 5

cars, 6
count, 6

datasets::cars, *6*
datasets::Orange, *13*
datasets::trees, *12*
dbGetPlot, 7
DBI::dbConnect(), *7*, *13*

linear, 8
lines, 9
ln, 10
log, 11

points, 11

trees, 12
type_classifications, 13

violins, 14