# Package 'dbplot'

February 7, 2020

**Version** 0.3.3

**Title** Simplifies Plotting Data Inside Databases

**Description** Leverages 'dplyr' to process the calculations of a plot inside a database.
This package provides helper functions that abstract the work at three levels:
outputs a 'ggplot', outputs the calculations, outputs the formula
needed to calculate bins.

**Depends** R (>= 3.1)

**Imports** dplyr (>= 0.7), rlang (>= 0.3), ggplot2, purrr, magrittr

**Suggests** dbplyr (>= 1.4.0), testthat, tidyr, covr

**License** GPL-3

**URL** `https://github.com/edgararuiz/dbplot`

**BugReports** `https://github.com/edgararuiz/dbplot/issues`

**RoxygenNote** 7.0.2

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Edgar Ruiz [aut, cre]

**Maintainer** Edgar Ruiz `<edgararuiz@gmail.com>`

**Repository** CRAN

**Date/Publication** 2020-02-07 01:10:09 UTC

## R topics documented:

---

`dbplot_bar` *Bar plot*

---

## Description

Uses very generic dplyr code to aggregate data and then ‘ggplot2‘ to create the plot. Because of this approach, the calculations automatically run inside the database if ‘data‘ has a database or sparklyr connection. The ‘class()‘ of such tables in R are: tbl_sql, tbl_dbi, tbl_spark

## Usage

```
dbplot_bar(data, x, ..., y = n())
```

## Arguments

| | |
|---|---|
| `data` | A table (tbl) |
| `x` | A discrete variable |
| `...` | A set of named or unamed aggregations |
| `y` | The aggregation formula. Defaults to count (n) |

## See Also

`dbplot_line` , `dbplot_histogram`, `dbplot_raster`

## Examples

```

library(ggplot2)
library(dplyr)

# Returns a plot of the row count per am
mtcars %>%
  dbplot_bar(am)

# Returns a plot of the average mpg per am
mtcars %>%
  dbplot_bar(am, mean(mpg))

# Returns the average and sum of mpg per am
mtcars %>%
  dbplot_bar(am, avg_mpg = mean(mpg), sum_mpg = sum(mpg))
```

## dbplot_boxplot *Boxplot*

### Description

Uses very generic dplyr code to aggregate data and then ‘ggplot2‘ to create the boxplot Because of this approach, the calculations automatically run inside the database if ‘data‘ has a database or sparklyr connection. The ‘class()‘ of such tables in R are: tbl_sql, tbl_dbi, tbl_spark

It currently only works with Spark and Hive connections.

### Usage

```
dbplot_boxplot(data, x, var, coef = 1.5)
```

### Arguments

| | |
|---|---|
| `data` | A table (tbl) |
| `x` | A discrete variable in which to group the boxplots |
| `var` | A continuous variable |
| `coef` | Length of the whiskers as multiple of IQR. Defaults to 1.5 |

### See Also

`dbplot_bar`, `dbplot_line` , `dbplot_raster`, `dbplot_histogram`

## dbplot_histogram *Histogram*

### Description

Uses very generic dplyr code to aggregate data and then ‘ggplot2‘ to create the histogram. Because of this approach, the calculations automatically run inside the database if ‘data‘ has a database or sparklyr connection. The ‘class()‘ of such tables in R are: tbl_sql, tbl_dbi, tbl_spark

### Usage

```
dbplot_histogram(data, x, bins = 30, binwidth = NULL)
```

### Arguments

| | |
|---|---|
| `data` | A table (tbl) |
| `x` | A continuous variable |
| `bins` | Number of bins. Defaults to 30. |
| `binwidth` | Single value that sets the side of the bins, it overrides bins |

## See Also

dbplot_bar, dbplot_line , dbplot_raster

## Examples

```
library(ggplot2)
library(dplyr)

# A ggplot histogram with 30 bins
mtcars %>%
  dbplot_histogram(mpg)

# A ggplot histogram with bins of size 10
mtcars %>%
  dbplot_histogram(mpg, binwidth = 10)
```

---

## dbplot_line *Bar plot*

---

### Description

Uses very generic dplyr code to aggregate data and then ‘ggplot2‘ to create a line plot. Because of this approach, the calculations automatically run inside the database if ‘data‘ has a database or sparklyr connection. The ‘class()‘ of such tables in R are: tbl_sql, tbl_dbi, tbl_spark

If multiple named aggregations are passed, ‘dbplot‘ will only use one SQL query to perform all of the operations. The purpose is to increase efficiency, and only make one "trip" to the database in order to obtains multiple, related, plots.

### Usage

```
dbplot_line(data, x, ..., y = n())
```

### Arguments

| | |
|---|---|
| data | A table (tbl) |
| x | A discrete variable |
| ... | A set of named or unamed aggregations |
| y | The aggregation formula. Defaults to count (n) |

### See Also

dbplot_bar, dbplot_histogram, dbplot_raster

## Examples

```
library(ggplot2)
library(dplyr)

# Returns a plot of the row count per cyl
mtcars %>%
  dbplot_line(cyl)

# Returns a plot of the average mpg per cyl
mtcars %>%
  dbplot_line(cyl, mean(mpg))

# Returns the average and sum of mpg per am
mtcars %>%
  dbplot_line(am, avg_mpg = mean(mpg), sum_mpg = sum(mpg))
```

---

## `dbplot_raster` *Raster plot*

---

### Description

To visualize two continuous variables, we typically resort to a Scatter plot. However, this may not be practical when visualizing millions or billions of dots representing the intersections of the two variables. A Raster plot may be a better option, because it concentrates the intersections into squares that are easier to parse visually.

Uses very generic dplyr code to aggregate data and ggplot2 to create a raster plot. Because of this approach, the calculations automatically run inside the database if ‘data‘ has a database or sparklyr connection. The ‘class()‘ of such tables in R are: tbl_sql, tbl_dbi, tbl_spark

### Usage

```
dbplot_raster(data, x, y, fill = n(), resolution = 100, complete = FALSE)
```

### Arguments

| | |
|---|---|
| `data` | A table (tbl) |
| `x` | A continuous variable |
| `y` | A continuous variable |
| `fill` | The aggregation formula. Defaults to count (n) |
| `resolution` | The number of bins created by variable. The highest the number, the more records can be potentially imported from the sourd |
| `complete` | Uses tidyr::complete to include empty bins. Inserts value of 0. |

## Details

There are two considerations when using a Raster plot with a database. Both considerations are related to the size of the results downloaded from the database:

- The number of bins requested: The higher the bins value is, the more data is downloaded from the database.

- How concentrated the data is: This refers to how many intersections return a value. The more intersections without a value, the less data is downloaded from the database.

## See Also

dbplot_bar, dbplot_line , dbplot_histogram

## Examples

```
library(ggplot2)
library(dplyr)

# Returns a 100x100 raster plot of record count of intersections of eruptions and waiting
faithful %>%
  dbplot_raster(eruptions, waiting)

# Returns a 50x50 raster plot of eruption averages of intersections of eruptions and waiting
faithful %>%
  dbplot_raster(eruptions, waiting, fill = mean(eruptions), resolution = 50)
```

---

db_bin *Bin formula*

---

## Description

Uses the rlang package to build the formula needed to create the bins of a numeric variable in an unevaluated fashion. This way, the formula can be then passed inside a dplyr verb.

## Usage

```
db_bin(var, bins = 30, binwidth = NULL)
```

## Arguments

| | |
|---|---|
| var | Variable name or formula |
| bins | Number of bins. Defaults to 30. |
| binwidth | Single value that sets the side of the bins, it overrides bins |

## Examples

```
library(dplyr)

# Important: Always name the field and
# prefix the function with `!!`` (See Details)

# Uses the default 30 bins
mtcars %>%
  group_by(x = !!db_bin(mpg)) %>%
  tally()

# Uses binwidth which overrides bins
mtcars %>%
  group_by(x = !!db_bin(mpg, binwidth = 10)) %>%
  tally()
```

---

`db_compute_bins` *Calculates a histogram bins*

---

## Description

Uses very generic dplyr code to create histogram bins. Because of this approach, the calculations automatically run inside the database if ‘data‘ has a database or sparklyr connection. The ‘class()‘ of such tables in R are: tbl_sql, tbl_dbi, tbl_spark

## Usage

```
db_compute_bins(data, x, bins = 30, binwidth = NULL)
```

## Arguments

| | |
|---|---|
| `data` | A table (tbl) |
| `x` | A continuous variable |
| `bins` | Number of bins. Defaults to 30. |
| `binwidth` | Single value that sets the side of the bins, it overrides bins |

## See Also

`db_bin`,

## Examples

```
# Returns record count for 30 bins in mpg
mtcars %>%
  db_compute_bins(mpg)
```

```
# Returns record count for bins of size 10
mtcars %>%
  db_compute_bins(mpg, binwidth = 10)
```

## db_compute_boxplot *Returns a dataframe with boxplot calculations*

### Description

Uses very generic dplyr code to create boxplot calculations. Because of this approach, the calculations automatically run inside the database if ‘data‘ has a database or sparklyr connection. The ‘class()‘ of such tables in R are: tbl_sql, tbl_dbi, tbl_spark

It currently only works with Spark, Hive, and SQL Server connections.

Note that this function supports input tbl that already contains grouping variables. This can be useful when creating faceted boxplots.

### Usage

```
db_compute_boxplot(data, x, var, coef = 1.5)
```

### Arguments

| | |
|---|---|
| `data` | A table (tbl), can already contain additional grouping vars specified |
| `x` | A discrete variable in which to group the boxplots |
| `var` | A continuous variable |
| `coef` | Length of the whiskers as multiple of IQR. Defaults to 1.5 |

### Examples

```
mtcars %>%
  db_compute_boxplot(am, mpg)
```

## db_compute_count *Aggregates over a discrete field*

### Description

Uses very generic dplyr code to aggregate data. Because of this approach, the calculations automatically run inside the database if ‘data‘ has a database or sparklyr connection. The ‘class()‘ of such tables in R are: tbl_sql, tbl_dbi, tbl_sql

### Usage

```
db_compute_count(data, x, ..., y = n())
```

## Arguments

| | |
|---|---|
| `data` | A table (tbl) |
| `x` | A discrete variable |
| `...` | A set of named or unamed aggregations |
| `y` | The aggregation formula. Defaults to count (n) |

## Examples

```
# Returns the row count per am
mtcars %>%
  db_compute_count(am)

# Returns the average mpg per am
mtcars %>%
  db_compute_count(am, mean(mpg))

# Returns the average and sum of mpg per am
mtcars %>%
  db_compute_count(am, mean(mpg), sum(mpg))
```

---

## `db_compute_raster` *Aggregates intersections of two variables*

---

## Description

To visualize two continuous variables, we typically resort to a Scatter plot. However, this may not be practical when visualizing millions or billions of dots representing the intersections of the two variables. A Raster plot may be a better option, because it concentrates the intersections into squares that are easier to parse visually.

Uses very generic dplyr code to aggregate data. Because of this approach, the calculations automatically run inside the database if ‘data‘ has a database or sparklyr connection. The ‘class()‘ of such tables in R are: tbl_sql, tbl_dbi, tbl_sql

## Usage

```
db_compute_raster(data, x, y, fill = n(), resolution = 100, complete = FALSE)

db_compute_raster2(data, x, y, fill = n(), resolution = 100, complete = FALSE)
```

## Arguments

| | |
|---|---|
| `data` | A table (tbl) |
| `x` | A continuous variable |
| `y` | A continuous variable |
| `fill` | The aggregation formula. Defaults to count (n) |

| | |
|---|---|
| `resolution` | The number of bins created by variable. The highest the number, the more records can be potentially imported from the source |
| `complete` | Uses tidyr::complete to include empty bins. Inserts value of 0. |

## Details

There are two considerations when using a Raster plot with a database. Both considerations are related to the size of the results downloaded from the database:

- The number of bins requested: The higher the bins value is, the more data is downloaded from the database.

- How concentrated the data is: This refers to how many intersections return a value. The more intersections without a value, the less data is downloaded from the database.

## Examples

```
# Returns a 100x100 grid of record count of intersections of eruptions and waiting
faithful %>%
  db_compute_raster(eruptions, waiting)

# Returns a 50x50 grid of eruption averages of intersections of eruptions and waiting
faithful %>%
  db_compute_raster(eruptions, waiting, fill = mean(eruptions), resolution = 50)
```

# Index