

# Package: plotGrouper (via r-universe)

June 3, 2026

**Title** Shiny app GUI wrapper for ggplot with built-in statistical analysis

**Version** 1.30.0

**Description** A shiny app-based GUI wrapper for ggplot with built-in statistical analysis. Import data from file and use dropdown menus and checkboxes to specify the plotting variables, graph type, and look of your plots. Once created, plots can be saved independently or stored in a report that can be saved as a pdf. If new data are added to the file, the report can be refreshed to include new data. Statistical tests can be selected and added to the graphs. Analysis of flow cytometry data is especially integrated with plotGrouper. Count data can be transformed to return the absolute number of cells in a sample (this feature requires inclusion of the number of beads per sample and information about any dilution performed).

**Depends** R (>= 3.5)

**Imports** ggplot2 (>= 3.0.0), dplyr (>= 0.7.6), tidyr (>= 0.2.0), tibble (>= 1.4.2), stringr (>= 1.3.1), readr (>= 1.1.1), readxl (>= 1.1.0), scales (>= 1.0.0), stats, grid, gridExtra (>= 2.3), egg (>= 0.4.0), gtable (>= 0.2.0), ggpubr (>= 0.1.8), shiny (>= 1.1.0), shinythemes (>= 1.1.1), colourpicker (>= 1.0), magrittr (>= 1.5), Hmisc (>= 4.1.1), rlang (>= 0.2.2)

**Suggests** knitr, htmltools, BiocStyle, rmarkdown, testthat

**VignetteBuilder** knitr

**biocViews** ImmunoOncology, FlowCytometry, GraphAndNetwork, StatisticalMethod, DataImport, GUI, MultipleComparison

**URL** <https://jdgagnon.github.io/plotGrouper/>

**BugReports** <https://github.com/jdgagnon/plotGrouper/issues>

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.0

**Config/pak/sysreqs** cmake make libicu-dev libuv1-dev libx11-dev zlib1g-dev

**Repository** https://bioc-release.r-universe.dev

**Date/Publication** 2026-04-28 12:48:35 UTC

**RemoteUrl** https://github.com/bioc/plotGrouper

**RemoteRef** RELEASE\_3\_23

**RemoteSha** 1186b731b65a80703a8a27acd5fa1290be17b7d3

## Contents

<code>%&gt;%</code> . . . . .	2
<code>gplot</code> . . . . .	3
<code>organizeData</code> . . . . .	5
<code>plotGrouper</code> . . . . .	6
<code>readData</code> . . . . .	7
<code>readData_example</code> . . . . .	7
<b>Index</b>	<b>9</b>

---

`%>%`

*Pipe graphics*

---

## Description

Like `dplyr`, `ggvis` also uses the pipe function, `%>%` to turn function composition into a series of imperative statements.

## Arguments

`lhs`, `rhs`      A visualisation and a function to apply to it

## Examples

```
# Instead of
dplyr::mutate(dplyr::filter(iris, Species == "versicolor"),
"Sample" = paste0(Species, dplyr::row_number()))
# You can write
dplyr::filter(iris, Species == "versicolor") %>%
dplyr::mutate("Sample" = paste0(Species, "_", dplyr::row_number()))
```

gplot

*A function to create a grouped plot and return a table grob.***Description**

This function allows you to create a grouped plot and return a table grob. It takes a tidy dataset containing sample replicate values for at least one variable, a column organizing each replicate into the proper comparison group, and a column that groups the variables to be plotted. Additional arguments allow for the re-ordering of the variables and the comparisons being plotted, selection of the type of graph to display (e.g., bar graph, boxplot, violin plot, points, statistical summary, etc...), as well as other aesthetics of the plot.

**Usage**

```
gplot(dataset = NULL, comparison = NULL, group.by = NULL,
       levs = TRUE, val = "value", geom = c("bar", "errorbar", "point",
       "stat", "seg"), p = "p.signif", ref.group = NULL,
       p.adjust.method = "holm", comparisons = NULL, method = "t.test",
       paired = FALSE, errortype = "mean_sdl", y.lim = NULL,
       y.lab = NULL, trans.y = "identity", x.lim = c(NA, NA),
       expand.y = c(0, 0), x.lab = NULL, trans.x = "identity",
       sci = FALSE, angle.x = FALSE, levs.comps = TRUE,
       group.labs = NULL, stats = FALSE, split = TRUE, split_str = NULL,
       trim = "none", leg.pos = "top", stroke = 0.25, font_size = 9,
       size = 1, width = 0.8, dodge = 0.8, plotWidth = 30,
       plotHeight = 40, shape.groups = c(19, 21),
       color.groups = c("black", "black"), fill.groups = c("#444444", NA,
       "#A33838"))
```

**Arguments**

dataset	Define your data set which should be a gathered tibble
comparison	Specify the comparison you would like to make (e.g., Genotype)
group.by	Specify the variable to group by (e.g., Tissue).
levs	Specify the order of the grouping variables
val	Specify column name that contains values (optional)
geom	Define the list of geoms you want to plot
p	Specify representation of pvalue (p.signif = astrisk representation of the raw p value; p.format = 'p = 0.05'; p.adj = adjusted p-value; p.adj.signif = astrisk representation of the adjusted p value)
ref.group	Specify a reference group to compare all other comparisons to
p.adjust.method	Method used for adjusting the pvalue
comparisons	Specify which of the available comparisons within your data you would like to plot

<code>method</code>	Specify the statistical test to be used
<code>paired</code>	Specify whether or not the statistical comparisons should be paired
<code>errortype</code>	Specify the method of statistical error to plot
<code>y.lim</code>	Specify the min and max values to be used for the y-axis
<code>y.lab</code>	Specify a custom y-axis label to use
<code>trans.y</code>	Specify the transformation to perform on the dependent variable
<code>x.lim</code>	Specify the min and max values to be used for the x-axis
<code>expand.y</code>	Specify values to expand the y-axis
<code>x.lab</code>	Specify a custom x-axis label to use
<code>trans.x</code>	Specify the transformation to perform on the independent variable
<code>sci</code>	Specify whether or not to display the dependent variable using scientific notation
<code>angle.x</code>	Specify whether or not to angle the x-axis text 45deg
<code>levs.comps</code>	Specify the order in which to plot the comparisons
<code>group.labs</code>	Specify custom labels for the independent variables
<code>stats</code>	Specify whether or not to output the statistics table
<code>split</code>	Specify whether or not to split the x-axis label text
<code>split_str</code>	Specify the string to split the x-axis label text by; uses regex
<code>trim</code>	Specify the string to trim text from the right side of the x-axis label text; uses regex
<code>leg.pos</code>	Specify where to place the legend
<code>stroke</code>	Specify the line thickness to use
<code>font_size</code>	Specify the font size to use
<code>size</code>	Specify the size of the points to use
<code>width</code>	Specify the width of groups to be plotted
<code>dodge</code>	Specify the width to dodge the comparisons by
<code>plotWidth</code>	Specify the length of the x-axis in mm
<code>plotHeight</code>	Specify the length of the y-axis in mm
<code>shape.groups</code>	Specify the default shapes to use for the comparisons
<code>color.groups</code>	Specify the default colors to use for the comparisons
<code>fill.groups</code>	Specify the default fills to use for the comparisons

**Value**

Table grob of the plot

**Examples**

```
iris %>% dplyr::mutate(Species = as.character(Species)) %>%
dplyr::group_by(Species) %>%
dplyr::mutate(Sample = paste0(Species, "_", dplyr::row_number()),
Sheet = "iris") %>%
dplyr::select(Sample, Sheet, Species, dplyr::everything()) %>%
tidyr::gather(variable, value, -c(Sample, Sheet, Species)) %>%
dplyr::filter(variable == "Sepal.Length") %>%
plotGrouper::gplot(
comparison = "Species",
group.by = "variable",
shape.groups = c(19,21,17),
color.groups = c(rep("black",3)),
fill.groups = c("black","#E016BE", "#1243C9")) %>%
gridExtra::grid.arrange()
```

---

organizeData	<i>A function to organize a tibble into tidy format and perform count transformations</i>
--------------	---

---

**Description**

This function will organize a tibble into tidy format and perform count transformations if appropriate columns are specified.

**Usage**

```
organizeData(data = NULL, exclude = NULL, comp = NULL,
comps = NULL, variables = NULL, id = NULL, beadColumn = NULL,
dilutionColumn = NULL)
```

**Arguments**

data	A tibble
exclude	A list of columns to exclude from gather
comp	the name of comparison column
comps	A vector of names of the comparisons
variables	A vector of the variables to be plotted
id	The name of unique identifier column
beadColumn	The column name that has total number of beads/sample
dilutionColumn	The column name that has dilution factor for each sample 1/x

**Value**

Tibble in tidy format based on columns chosen to be excluded. Count data will be transformed if appropriate columns are present.

## Examples

```
iris %>% dplyr::mutate(Species = as.character(Species)) %>%
dplyr::group_by(Species) %>%
dplyr::mutate(Sample = paste0(Species, "_", dplyr::row_number()),
Sheet = "iris") %>%
dplyr::select(Sample, Sheet, Species, dplyr::everything()) %>%
plotGrouper::organizeData(data = .,
exclude = c("Sample", "Sheet", "Species"),
comp = "Species",
comps = c("setosa", "versicolor", "virginica"),
variables = "Sepal.Length",
id = "Sample",
beadColumn = "none",
dilutionColumn = "none")
```

---

plotGrouper

*A function to run the plotGrouper shiny app*

---

## Description

This function runs the plotGrouper app

## Usage

```
plotGrouper(...)
```

## Arguments

... Any argument that you can pass to shiny::runApp

## Value

Runs the plotGrouper shiny app.

## Examples

```
# plotGrouper()
```

---

readData	<i>A function to read an excel file and combine its sheets into a single dataframe.</i>
----------	---

---

**Description**

This function will read an excel file and combine its sheets into a single dataframe.

**Usage**

```
readData(file = NULL, sheet = NULL)
```

**Arguments**

file	Takes an excel file to be read from
sheet	Takes a vector of sheets to be read

**Value**

Tibble assembled from the sheets selected from the file

**Examples**

```
datasets <- readData_example("iris.xlsx")  
readData(datasets, "iris")
```

---

readData_example	<i>Get path to readData example</i>
------------------	-------------------------------------

---

**Description**

readData comes bundled with a example files in its 'inst/applications/www' directory. This function makes them easy to access.

**Usage**

```
readData_example(path = NULL)
```

**Arguments**

path	Name of file. If 'NULL', the example files will be listed.
------	--

**Value**

Located example excel file in package

**Examples**

```
readData_example(path = "iris.xlsx")
```

# Index

- \* **organizeData**
  - organizeData, 5
- \* **readData**
  - readData, 7
- %>%, 2
- gplot, 3
- organizeData, 5
- plotGrouper, 6
- readData, 7
- readData\_example, 7