

# Package: gg1d (via r-universe)

May 17, 2026

**Title** Exploratory Data Analysis using Tiled One-Dimensional Graphics

**Version** 0.1.0.9000

**Description** Streamlines exploratory data analysis by providing a turnkey approach to visualising n-dimensional data which graphically reveals correlative or associative relationships between 2 or more features. Represents all dataset features as distinct, vertically aligned bar or tile plots, with plot types auto-selected based on whether variables are categorical or numeric.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**URL** <https://github.com/selkamand/gg1d>,  
<https://selkamand.github.io/gg1d/>

**BugReports** <https://github.com/selkamand/gg1d/issues>

**Imports** assertions (>= 0.2.0), cli, ggiraph (>= 0.8.11), ggplot2, ggtext, grDevices, patchwork (>= 1.3.0), rank (>= 0.1.1), rlang

**Suggests** covr, infotheo, knitr, rmarkdown, testthat (>= 3.0.0), TSP

**Config/Needs/website** uwot, datarium, palmerpenguins

**Config/testthat/edition** 3

**Depends** R (>= 2.10)

**LazyData** true

**Config/pak/sysreqs** libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev

**Repository** <https://selkamand.r-universe.dev>

**Date/Publication** 2025-09-03 02:05:31 UTC

**RemoteUrl** <https://github.com/selkamand/gg1d>

**RemoteRef** HEAD

**RemoteSha** 877da215dcb4712a4a04e5bd0b243680b2cd128b

## Contents

beautify . . . . .	2
column_info_table . . . . .	3
ggld . . . . .	4
ggld_options . . . . .	6
ggparallel . . . . .	10
ggparallel_options . . . . .	12
lazy_birdwatcher . . . . .	14
minibeans . . . . .	14
mutinfo . . . . .	15
sensible_2_breaks . . . . .	16
<b>Index</b>	<b>17</b>

---

beautify	<i>Make strings prettier for printing</i>
----------	---

---

### Description

Takes an input string and 'beautify' by converting underscores to spaces and

### Usage

```
beautify(string, autodetect_units = TRUE)
```

### Arguments

string	input string
autodetect_units	automatically detect units (e.g. mm, kg, etc) and wrap in brackets.

### Value

string

---

column\_info\_table      *Parse a tibble and ensure it meets standards*

---

## Description

Parse a tibble and ensure it meets standards

## Usage

```
column_info_table(  
  data,  
  maxlevels = 6,  
  col_id = NULL,  
  cols_to_plot,  
  tooltip_column_suffix = "_tooltip",  
  ignore_column_regex = "_ignore$",  
  palettes,  
  colours_default,  
  colours_default_logical,  
  verbose  
)
```

## Arguments

data	data.frame to autoplot (data.frame)
maxlevels	for categorical variables, what is the maximum number of distinct values to allow (too many will make it hard to find a palette that suits). (number)
col_id	name of column to use as an identifier. If null, artificial IDs will be created based on row-number.
cols_to_plot	names of columns in <b>data</b> that should be plotted. By default plots all valid columns (character)
tooltip_column_suffix	the suffix added to a column name that indicates column should be used as a tooltip (string)
ignore_column_regex	a regex string that, if matches a column name, will cause that column to be excluded from plotting (string). If NULL no regex check will be performed. (default: "_ignore\$")
palettes	A list of named vectors. List names correspond to <b>data</b> column names (categorical only). Vector names to levels of columns. Vector values are colours, the vector names are used to map values in data to a colour.
colours_default	Default colors for categorical variables without a custom palette.
colours_default_logical	Colors for binary variables: a vector of three colors representing TRUE, FALSE, and NA respectively (character).

verbose            Numeric value indicating the verbosity level:

- **2**: Highly verbose, all messages.
- **1**: Key messages only.
- **0**: Silent, no messages.

### Value

tibble with the following columns:

1. colnames
2. coltype (categorical/numeric/tooltip/invalid)
3. ndistinct (number of distinct values)
4. plottable (should this column be plotted)
5. tooltip\_col (the name of the column to use as the tooltip) or NA if no obvious tooltip column found

---

gg1d

*AutoPlot an entire data.frame*

---

### Description

Visualize all columns in a data frame with gg1d's vertically aligned plots and automatic plot selection based on variable type. Plots are fully interactive, and custom tooltips can be added.

### Usage

```
gg1d(
  data,
  col_id = NULL,
  col_sort = NULL,
  order_matches_sort = TRUE,
  maxlevels = 7,
  verbose = 2,
  drop_unused_id_levels = FALSE,
  interactive = TRUE,
  return = c("plot", "column_info", "data"),
  palettes = NULL,
  sort_type = c("frequency", "alphabetical"),
  desc = TRUE,
  limit_plots = TRUE,
  max_plottable_cols = 10,
  cols_to_plot = NULL,
  tooltip_column_suffix = "_tooltip",
  ignore_column_regex = "_ignore$",
  convert_binary_numeric_to_factor = TRUE,
  options = gg1d_options(show_legend = !interactive)
)
```

**Arguments**

<code>data</code>	data.frame to autoplot (data.frame)
<code>col_id</code>	name of column to use as an identifier. If null, artificial IDs will be created based on row-number.
<code>col_sort</code>	name of columns to sort on. To do a hierarchical sort, supply a vector of column names in the order they should be sorted (character).
<code>order_matches_sort</code>	should the column plots be stacked top-to-bottom in the order they appear in <code>col_sort</code> (flag)
<code>maxlevels</code>	for categorical variables, what is the maximum number of distinct values to allow (too many will make it hard to find a palette that suits). (number)
<code>verbose</code>	Numeric value indicating the verbosity level: <ul style="list-style-type: none"> <li>• <b>2</b>: Highly verbose, all messages.</li> <li>• <b>1</b>: Key messages only.</li> <li>• <b>0</b>: Silent, no messages.</li> </ul>
<code>drop_unused_id_levels</code>	if <code>col_id</code> is a factor with unused levels, should these be dropped or included in visualisation
<code>interactive</code>	produce interactive ggiraph visualiastion (flag)
<code>return</code>	a string describing what this function should return. Options include: <ul style="list-style-type: none"> <li>• <b>plot</b>: Return the ggld visualisation (default)</li> <li>• <b>column_info</b>: Return a data.frame describing the columns the dataset.</li> <li>• <b>data</b>: Return the processed dataset used for plotting.</li> </ul>
<code>palettes</code>	A list of named vectors. List names correspond to <b>data</b> column names (categorical only). Vector names to levels of columns. Vector values are colours, the vector names are used to map values in data to a colour.
<code>sort_type</code>	controls how categorical variables are sorted. Numerical variables are always sorted in numerical order irrespective of the value given here. Options are alphabetical or frequency
<code>desc</code>	sort in descending order (flag)
<code>limit_plots</code>	throw an error when there are > <code>max_plottable_cols</code> in dataset (flag)
<code>max_plottable_cols</code>	maximum number of columns that can be plotted (default: 10) (number)
<code>cols_to_plot</code>	names of columns in <b>data</b> that should be plotted. By default plots all valid columns (character)
<code>tooltip_column_suffix</code>	the suffix added to a column name that indicates column should be used as a tooltip (string)
<code>ignore_column_regex</code>	a regex string that, if matches a column name, will cause that column to be excluded from plotting (string). If NULL no regex check will be performed. (default: "_ignore\$")

`convert_binary_numeric_to_factor`  
 If a numeric column contains only values 0, 1, & NA, then automatically convert to a factor.

`options`  
 a list of additional visual parameters created by calling `gg1d_options()`. See [gg1d\\_options](#) for details.

## Value

ggiraph interactive visualisation

## Examples

```
path_gg1d <- system.file("example.csv", package = "gg1d")
df <- read.csv(path_gg1d, header = TRUE, na.strings = "")

# Create Basic Plot
gg1d(df, col_id = "ID", col_sort = "Glasses")

# Configure plot gg1d_options()
gg1d(
  lazy_birdwatcher,
  col_sort = "Magpies",
  palettes = list(
    Birdwatcher = c(Robert = "#E69F00", Catherine = "#999999"),
    Day = c(Weekday = "#999999", Weekend = "#009E73")
  ),
  options = gg1d_options(
    show_legend = TRUE,
    fontsize_barplot_y_numbers = 12,
    legend_text_size = 16,
    legend_key_size = 1,
    legend_nrow = 1,
  )
)
```

---

gg1d\_options

*Visual Parameters for gg1d Plots*

---

## Description

Configures aesthetic and layout settings for plots generated by `gg1d`.

## Usage

```
gg1d_options(
  colours_default = c("#66C2A5", "#FC8D62", "#8DA0CB", "#E78AC3", "#A6D854", "#FFD92F",
    "#E5C494"),
  colours_default_logical = c(`TRUE` = "#648fff", `FALSE` = "#dc267f"),
```

```

colours_missing = "grey90",
show_legend_titles = FALSE,
legend_title_position = c("top", "bottom", "left", "right"),
legend_nrow = 4,
legend_ncol = NULL,
legend_title_size = NULL,
legend_text_size = NULL,
legend_key_size = 0.3,
legend_orientation_heatmap = c("horizontal", "vertical"),
show_legend = TRUE,
legend_position = c("right", "left", "bottom", "top"),
na_marker = "!",
na_marker_size = 8,
na_marker_colour = "black",
show_na_marker_categorical = FALSE,
show_na_marker_heatmap = FALSE,
colours_heatmap_low = "purple",
colours_heatmap_high = "seagreen",
transform_heatmap = c("identity", "log10", "log2"),
fontsize_values_heatmap = 3,
show_values_heatmap = FALSE,
colours_values_heatmap = "white",
vertical_spacing = 0,
numeric_plot_type = c("bar", "heatmap"),
y_axis_position = c("left", "right"),
width = 0.9,
relative_height_numeric = 4,
cli_header = "Running gg1d",
interactive_svg_width = NULL,
interactive_svg_height = NULL,
fontsize_barplot_y_numbers = 8,
max_digits_barplot_y_numbers = 3,
fontsize_y_title = 12,
beautify_text = TRUE
)

```

## Arguments

`colours_default`  
Default colors for categorical variables without a custom palette.

`colours_default_logical`  
Colors for binary variables: a vector of three colors representing TRUE, FALSE, and NA respectively (character).

`colours_missing`  
Color for missing (NA) values in categorical plots (string).

`show_legend_titles`  
Display titles for legends (flag).

<code>legend_title_position</code>	Position of the legend title ("top", "bottom", "left", "right").
<code>legend_nrow</code>	Number of rows in the legend (number).
<code>legend_ncol</code>	Number of columns in the legend. If set, <code>legend_nrow</code> should be NULL (number).
<code>legend_title_size</code>	Size of the legend title text (number).
<code>legend_text_size</code>	Size of the text within the legend (number).
<code>legend_key_size</code>	Size of the legend key symbols (number).
<code>legend_orientation_heatmap</code>	should legend orientation be "horizontal" or "vertical".
<code>show_legend</code>	Display the legend on the plot (flag).
<code>legend_position</code>	Position of the legend ("right", "left", "bottom", "top").
<code>na_marker</code>	Text used to mark NA values in numeric plots (string).
<code>na_marker_size</code>	Size of the text marker for NA values (number).
<code>na_marker_colour</code>	Color of the NA text marker (string).
<code>show_na_marker_categorical</code>	Show a marker for NA values on categorical tiles (flag).
<code>show_na_marker_heatmap</code>	Show a marker for NA values on heatmap tiles (flag).
<code>colours_heatmap_low</code>	Color for the lowest value in heatmaps (string).
<code>colours_heatmap_high</code>	Color for the highest value in heatmaps (string).
<code>transform_heatmap</code>	Transformation to apply before visualizing heatmap values ("identity", "log10", "log2").
<code>fontsize_values_heatmap</code>	Font size for heatmap values (number).
<code>show_values_heatmap</code>	Display numerical values on heatmap tiles (flag).
<code>colours_values_heatmap</code>	Color for heatmap values (string).
<code>vertical_spacing</code>	Space between each data row in points (number).
<code>numeric_plot_type</code>	Type of visualization for numeric data: "bar" or "heatmap".
<code>y_axis_position</code>	Position of the y-axis ("left" or "right").

width	controls how much space is present between bars and tiles within each plot. Can be 0-1 where values of 1 makes bars/tiles take up 100% of available space (no gaps between bars).
relative_height_numeric	how many times taller should numeric plots be relative to categorical tile plots. Only taken into account if numeric_plot_type == "bar" (number)
cli_header	Text used for h1 header. Included so it can be tweaked by packages that use gg1d, so they can customise how the info messages appear.
interactive_svg_width, interactive_svg_height	width and height of the interactive graphic region (in inches). Only used when interactive = TRUE.
fontsize_barplot_y_numbers	fontsize of the text describing numeric barplot max & min values (number).
max_digits_barplot_y_numbers	Number of digits to round the numeric barplot max and min values to (number).
fontsize_y_title	fontsize of the y axis titles (a.k.a the data.frame column names) (number).
beautify_text	Beautify y-axis text and legend titles by capitalizing words and adding spaces (flag).

### Value

A list of visualization parameters for gg1d.

### Examples

```
path_gg1d <- system.file("example.csv", package = "gg1d")
df <- read.csv(path_gg1d, header = TRUE, na.strings = "")

# Create Basic Plot
gg1d(df, col_id = "ID", col_sort = "Glasses")

# Configure plot gg1d_options()
gg1d(
  lazy_birdwatcher,
  col_sort = "Magpies",
  palettes = list(
    Birdwatcher = c(Robert = "#E69F00", Catherine = "#999999"),
    Day = c(Weekday = "#999999", Weekend = "#009E73")
  ),
  options = gg1d_options(
    show_legend = TRUE,
    fontsize_barplot_y_numbers = 12,
    legend_text_size = 16,
    legend_key_size = 1,
    legend_nrow = 1,
  )
)
```

ggparallel

*Parallel Coordinate Plots***Description**

Visualize relationships between numeric variables and categorical groupings using parallel coordinate plots.

**Usage**

```
ggparallel(
  data,
  col_id = NULL,
  col_colour = NULL,
  highlight = NULL,
  interactive = TRUE,
  order_columns_by = c("appearance", "random", "auto"),
  order_observations_by = c("frequency", "original"),
  verbose = TRUE,
  palette_colour = palette.colors(palette = "Set2"),
  palette_highlight = c("red", "grey90"),
  convert_binary_numeric_to_factor = TRUE,
  scaling = c("uniminmax", "none"),
  return = c("plot", "data"),
  options = ggparallel_options()
)
```

**Arguments**

<code>data</code>	A data frame containing the variables to plot.
<code>col_id</code>	The name of the column to use as an identifier. If NULL, artificial IDs will be generated based on row numbers. (character)
<code>col_colour</code>	Name of the column to use for coloring lines in the plot. If NULL, no coloring is applied. (character)
<code>highlight</code>	A level from <code>col_colour</code> to emphasize in the plot. Ignored if <code>col_colour</code> is not set. (character)
<code>interactive</code>	Produce interactive ggiraph visualization (flag)
<code>order_columns_by</code>	Strategy for ordering columns in the plot. Options include: <ul style="list-style-type: none"> <li>• <b>"appearance"</b>: Order columns by their order in data (default).</li> <li>• <b>"random"</b>: Randomly order columns.</li> <li>• <b>"auto"</b>: Automatically order columns based on context: <ul style="list-style-type: none"> <li>– If <code>highlight</code> is set, columns are ordered to maximize separation between the highlighted level and all others, using mutual information.</li> </ul> </li> </ul>

- If `col_colour` is set but `highlight` is not, columns are ordered based on mutual information with all classes in `col_colour`.
- If neither `highlight` nor `col_colour` is set, columns are ordered to minimize the estimated number of crossings, using a repetitive nearest neighbour approach with two-opt refinement.

`order_observations_by` Strategy for ordering lines in the plot. Options include:

- **"frequency"**: Draw the largest groups first.
- **"original"**: Preserve the original order in data.

Ignored if `highlight` is set.

`verbose` Logical; whether to display informative messages during execution. (default: TRUE)

`palette_colour` A named vector of colors for categorical levels in `col_colour`. (default: Set2 palette)

`palette_highlight` A two-color vector for highlighting (`highlight` and others). (default: `c("red", "grey90")`)

`convert_binary_numeric_to_factor` Logical; whether to convert numeric columns containing only 0, 1, and NA to factors. (default: TRUE)

`scaling` Method for scaling numeric variables. Options include:

- **"uniminmax"**: Rescale each variable to range [0, 1].
- **"none"**: No rescaling. Use raw values.

`return` What to return. Options include:

- **"plot"**: Return the ggplot object (default).
- **"data"**: Return the processed data used for plotting.

`options` A list of additional visualization parameters created by `ggparallel_options()`.

**Value**

A ggplot object or a processed data frame, depending on the return parameter.

**Examples**

```
ggparallel(
  data = minibeans,
  col_colour = "Class",
  order_columns_by = "auto"
)

ggparallel(
  data = minibeans,
  col_colour = "Class",
  highlight = "DERMASON",
  order_columns_by = "auto"
)
```

---

**ggparallel\_options**      *Visual Parameters for ggparallel Plots*

---

**Description**

Configures aesthetic and layout settings for plots generated by ggparallel.

**Usage**

```
ggparallel_options(  
  show_legend = TRUE,  
  show_legend_titles = FALSE,  
  legend_position = c("bottom", "right", "left", "top"),  
  legend_title_position = c("left", "top", "bottom", "right"),  
  legend_nrow = NULL,  
  legend_ncol = NULL,  
  legend_key_size = 1,  
  beautify_text = TRUE,  
  max_digits_bounds = 1,  
  x_axis_text_angle = 90,  
  x_axis_text_hjust = 0,  
  x_axis_text_vjust = 0.5,  
  fontsize_x_axis_text = 12,  
  show_column_names = TRUE,  
  show_points = FALSE,  
  show_bounds_labels = FALSE,  
  show_bounds_rect = FALSE,  
  line_alpha = 0.5,  
  line_width = NULL,  
  line_type = 1,  
  x_axis_gridlines = ggplot2::element_line(colour = "black"),  
  interactive_svg_width = NULL,  
  interactive_svg_height = NULL  
)
```

**Arguments**

<code>show_legend</code>	Display the legend on the plot (flag).
<code>show_legend_titles</code>	Display titles for legends (flag).
<code>legend_position</code>	Position of the legend ("right", "left", "bottom", "top").
<code>legend_title_position</code>	Position of the legend title ("top", "bottom", "left", "right").
<code>legend_nrow</code>	Number of rows in the legend (number).

legend_ncol	Number of columns in the legend. If set, legend_nrow should be NULL (number).
legend_key_size	Size of the legend key symbols. (number).
beautify_text	Beautify y-axis text and legend titles by capitalizing words and adding spaces (flag).
max_digits_bounds	Number of digits to round the axis bounds label text to (number)
x_axis_text_angle	Angle of the x axis text describing column names (number)
x_axis_text_hjust	Horizontal Justification of the x axis text describing column names (number)
x_axis_text_vjust	Vertical Justification of the x axis text describing column names (number)
fontsize_x_axis_text	fontsize of the x-axis text describing column names (number)
show_column_names	Show column names as x axis text (flag)
show_points	Show points (flag)
show_bounds_labels	Show bounds (min and max value) of each feature with labels above / below the axes (flag)
show_bounds_rect	Show bounds (min and max value) of each feature with a rectangular graphic (flag)
line_alpha	Alpha of line geom (number)
line_width	Width of the line geom (number)
line_type	Type of line geom (number)
x_axis_gridlines	Customise look of x axis gridlines. Must be either a call to <code>ggplot2::element_line()</code> or <code>ggplot2::element_blank()</code> .
interactive_svg_width, interactive_svg_height	Width and height of the interactive graphic region (in inches). Only used when <code>interactive = TRUE</code> .

**Value**

A list of visualization parameters for ggparallel.

**Examples**

```
ggparallel(
  data = minibeans,
  col_colour = "Class",
  order_columns_by = "auto"
)
```

```
ggparallel(  
  data = minibeans,  
  col_colour = "Class",  
  highlight = "DERMASON",  
  order_columns_by = "auto"  
)
```

---

lazy\_birdwatcher      *Lazy Birdwatcher Dataset*

---

### Description

A simulated dataset describing the number of magpies observed by two birdwatchers.

### Usage

```
lazy_birdwatcher
```

### Format

lazy\_birdwatcher:

A data frame with 45 rows and 3 columns:

**Magpies** Number of magpies observed

**Day** Was the day of observation a weekday or a weekend?

**Birdwatcher** Name of the birdwatcher

---

minibeans      *Dry Beans Dataset*

---

### Description

A subsample of the Koklu & Ozkan (2020) dry beans dataset produced by imaging a total of 13,611 grains from 7 varieties of dry beans. The original dataset contains 13,611 observations, but here we include a random subsample of 1000.

### Usage

```
minibeans
```

**Format**

minibeans:

A data frame with 1000 rows and 17 columns:

**Area** The area of a bean zone and the number of pixels within its boundaries.

**Perimeter** Bean circumference is defined as the length of its border.

**Major axis length** The distance between the ends of the longest line that can be drawn from a bean.

**Minor axis length** The longest line that can be drawn from the bean while standing perpendicular to the main axis.

**Aspect ratio** Defines the relationship between L and l.

**Eccentricity** Eccentricity of the ellipse having the same moments as the region.

**Convex area** Number of pixels in the smallest convex polygon that can contain the area of a bean seed.

**Equivalent diameter** The diameter of a circle having the same area as a bean seed area.

**Extent** The ratio of the pixels in the bounding box to the bean area.

**Solidity** Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.

**Roundness** Calculated with the following formula:  $(4\pi A)/(P^2)$ .

**Compactness** Measures the roundness of an object:  $Ed/L$ .

**ShapeFactor1** Shape factor 1.

**ShapeFactor2** Shape factor 2.

**ShapeFactor3** Shape factor 3.

**ShapeFactor4** Shape factor 4.

**Class** Seker, Barbunya, Bombay, Cali, Derosan, Horoz, and Sira.

**Source**

Koklu, M, and IA Ozkan. 2020. Multiclass Classification of Dry Beans Using Computer Vision and Machine Learning Techniques. *Computers and Electronics in Agriculture*, 174: 105507. doi: 10.1016/j.compag.2020.105507, <https://doi.org/10.24432/C50S4B>

---

mutinfo

*Compute Mutual Information*

---

**Description**

Computes mutual information between each feature in the features data frame and the target vector. The features are discretized using the "equalfreq" method from `infotheo::discretize()`.

**Usage**

```
mutinfo(features, target, return_colnames = FALSE)
```

**Arguments**

features	A data frame of features. These will be discretized using the "equalfreq" method (see <code>infotheo::discretize()</code> ).
target	A vector (character or factor) representing the variable to compute mutual information with.
return_colnames	Logical; if TRUE, returns the column names from features ordered by their mutual information with target (highest to lowest). If FALSE, returns mutual information values. (default: FALSE)

**Value**

If `return_colnames = FALSE`, a named numeric vector of mutual information scores is returned (one for each column in `features`), sorted in descending order. The names of the vector correspond to the column names of `features`. If `return_colnames = TRUE`, only the ordered column names of `features` are returned.

**Examples**

```
data(iris)
# Compute mutual information scores
mutinfo(iris[1:4], iris[[5]])

# Get column names ordered by mutual information with target column (most mutual info first)
mutinfo(iris[1:4], iris[[5]], return_colnames = TRUE)
```

---

sensible\_2\_breaks      *GGplot breaks*

---

**Description**

Find sensible values to add 2 breaks at for a ggplot2 axis

**Usage**

```
sensible_2_breaks(vector)
```

**Arguments**

vector	vector fed into ggplot axis you want to define sensible breaks for
--------	--

**Value**

vector of length 2. first element describes upper break position, lower describes lower break

# Index

## \* datasets

lazy\_birdwatcher, 14  
minibeans, 14

beautify, 2

column\_info\_table, 3

gg1d, 4

gg1d\_options, 6, 6

gg1d\_options(), 6

ggparallel, 10

ggparallel\_options, 12

ggplot2::element\_blank(), 13

ggplot2::element\_line(), 13

infotheo::discretize(), 15, 16

lazy\_birdwatcher, 14

minibeans, 14

mutinfo, 15

sensible\_2\_breaks, 16