Package: ggEDA (via r-universe)

May 25, 2026

Title Turnkey Visualisations for Exploratory Data Analysis

Version 0.2.0.9000

Description Provides interactive visualisations for exploratory data analysis of high-dimensional datasets. Includes parallel coordinate plots for exploring large datasets with mostly quantitative features, but also stacked one-dimensional visualisations that more effectively show missingness and complex categorical relationships in smaller datasets.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/CCICB/ggEDA, https://ccicb.github.io/ggEDA/

BugReports https://github.com/CCICB/ggEDA/issues

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

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

Config/Needs/website uwot, datarium, palmerpenguins

Config/testthat/edition 3

Depends R (>= 3.5)

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://ccicb.r-universe.dev

Date/Publication 2026-03-31 13:30:21 UTC

RemoteUrl https://github.com/CCICB/ggEDA

RemoteRef HEAD

RemoteSha 173814b242943cef47dfc7c1bdb7d222653eb887

Contents

`baseballfans` *Baseball Fans Dataset*

Description

An artificially generated dataset describing basic demographics and accessorization choices of baseball fans as part of a a hypothetical market research study from stadium merchandise vendors. None of the data are real; they were produced for illustrative and testing purposes only.

Usage

```
baseballfans
```

Format

`baseballfans`**:**

A data frame with 19 rows and 10 columns:

ID Unique integer identifier for each individual.

Age Age in years at time of observation.

Gender Self-reported gender ("Male" or "Female").

EyeColour Eye color ("Brown", "Green", "Blue"), or missing (NA) if not recorded.

Height Height in centimeters; missing (NA) if not recorded.

HairColour Hair color ("Black", "Blond", "Red", "Brown").

Glasses Logical flag (TRUE/FALSE) indicating whether the individual wears glasses.

WearingHat Logical flag (TRUE/FALSE) indicating whether the individual is wearing a hat.

WearingHat_tooltip Type of hat worn, if any (e.g., "baseball cap", "stetson", "fedora", "top hat"); empty when WearingHat == FALSE.

Date Date of observation in day/month/year format (e.g., 9/05/2023). Stored as character vector

#' @source Synthetic data; no real persons were observed.

Details

This mock dataset was created to demonstrate ggEDA functionality. All entries are fictional.

`beautify` *Make strings prettier for printing*

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 **data** 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 **data** 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

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

`data`	A data frame containing the variables to plot.
`col_id`	The name of the column to use as an identifier. If `NULL`, artificial IDs will be generated based on row numbers. (character)
`col_colour`	Name of the column to use for coloring lines in the plot. If `NULL`, no coloring is applied. (character)
`highlight`	A level from `col_colour` to emphasize in the plot. Ignored if `col_colour` is not set. (character)
`interactive`	Produce interactive ggiraph visualiastion (flag)
`order_columns_by`	Strategy for ordering columns in the plot. Options include: • **"appearance"**: Order columns by their order in `data` (default). • **"random"**: Randomly order columns. • **"auto"**: Automatically order columns based on context: – If `highlight` is set, columns are ordered to maximize separation between the highlighted level and all others, using mutual information.

  - 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"
)
```

```
# Customise appearance using options argument
ggparallel(
  data = minibeans,
  col_colour = "Class",
  order_columns_by = "auto",
  options = ggparallel_options(show_legend = FALSE)
)
```

---

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,
  beautify_values = FALSE,
  beautify_function = beautify,
  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,
  expand_x = ggplot2::waiver(),
  line_alpha = 0.5,
  line_width = 0.5,
  line_type = 1,
  x_axis_gridlines = ggplot2::element_line(colour = "black"),
  interactive_svg_width = NULL,
  interactive_svg_height = NULL
)
```

Arguments

show_legend	Display the legend on the plot (flag).
show_legend_titles	Display titles for legends (flag).
legend_position	Position of the legend ("right", "left", "bottom", "top").
legend_title_position	Position of the legend title ("top", "bottom", "left", "right").
legend_nrow	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 titlesto more human-readable forms (e.g. converting 'my_title' to 'My Title') (flag).
beautify_values	Beautify legend values to more human-readable forms (e.g. converting 'my_value' to 'My Value') (flag)
beautify_function	a function that takes a string and returns a nicely formatted string. Use to beautify axis & legend titles when beautify_text=TRUE, and legend values when beautify_values=TRUE.
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)
expand_x	A vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function ggplot2::expansion() to generate the values for the expand argument. The defaults are to expand the scale by 0.6 units on each side.

line_alpha — Alpha of line geom (number)

line_width — Width of the line geom (number)

line_type — Type of line geom (number or string. see ggplot2::aes_linetype_size_shape() for valid options)

x_axis_gridlines — Customise look of x axis gridlines. Must be either a call to ggplot2::element_line() or ggplot2::element_blank().

interactive_svg_width, interactive_svg_height — Width and height of the interactive graphic region (in inches). Only used when interactive = TRUE.

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"
)

# Customise appearance using options argument
ggparallel(
  data = minibeans,
  col_colour = "Class",
  order_columns_by = "auto",
  options = ggparallel_options(show_legend = FALSE)
)
```

---

ggstack *AutoPlot an entire data.frame*

Description

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

Usage

```
ggstack(
  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 = ggstack_options(show_legend = !interactive)
)
```

Arguments

`data`	data.frame to autoplot (data.frame)
`col_id`	name of column to use as an identifier. If null, artificial IDs will be created based on row-number.
`col_sort`	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).
`order_matches_sort`	should the column plots be stacked top-to-bottom in the order they appear in `col_sort` (flag)
`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)
`verbose`	Numeric value indicating the verbosity level: • **2**: Highly verbose, all messages. • **1**: Key messages only. • **0**: Silent, no messages.
`drop_unused_id_levels`	if col_id is a factor with unused levels, should these be dropped or included in visualisation
`interactive`	produce interactive ggiraph visualiastion (flag)
`return`	a string describing what this function should return. Options include: • **plot**: Return the ggEDA visualisation (default)

- **colum_info**: Return a data.frame describing the columns the dataset.
- **data**: Return the processed dataset used for plotting.

`palettes` A list of named vectors. List names correspond to **data** 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.

`sort_type` 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`

`desc` sort in descending order (flag)

`limit_plots` throw an error when there are > `max_plottable_cols` in dataset (flag)

`max_plottable_cols` maximum number of columns that can be plotted (default: 10) (number)

`cols_to_plot` names of columns in **data** 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$")

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

`options` a list of additional visual parameters created by calling `ggstack_options()`. See `ggstack_options` for details.

Value

ggiraph interactive visualisation

Examples

```
# Create Basic Plot
ggstack(baseballfans, col_id = "ID", col_sort = "Glasses")

# Configure plot ggstack_options()
ggstack(
  lazy_birdwatcher,
  col_sort = "Magpies",
  palettes = list(
    Birdwatcher = c(Robert = "#E69F00", Catherine = "#999999"),
    Day = c(Weekday = "#999999", Weekend = "#009E73")
  ),
  options = ggstack_options(
    show_legend = TRUE,
    fontsize_barplot_y_numbers = 12,
```

```
    legend_text_size = 16,
    legend_key_size = 1,
    legend_nrow = 1,
  )
)
```

---

ggstack_options *Visual Parameters for ggstack Plots*

Description

Configures aesthetic and layout settings for plots generated by ggstack.

Usage

```
ggstack_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 ggstack",
  inter_plot_spacing = 10,
  expand_x = ggplot2::waiver(),
  interactive_svg_width = NULL,
  interactive_svg_height = NULL,
  fontsize_barplot_y_numbers = 8,
  max_digits_barplot_y_numbers = 3,
  fontsize_y_title = 12,
  fontface_y_title = c("plain", "italic", "bold", "bold.italic"),
  beautify_text = TRUE,
  beautify_values = FALSE,
  beautify_function = beautify,
  margin_y_title = NULL,
  margin_y_numbers = NULL
)
```

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).

`legend_title_position`
: Position of the legend title ("top", "bottom", "left", "right").

`legend_nrow`
: Number of rows in the legend (number).

`legend_ncol`
: Number of columns in the legend. If set, `legend_nrow` should be `NULL` (number).

`legend_title_size`
: Size of the legend title text (number).

`legend_text_size`
: Size of the text within the legend (number).

`legend_key_size`
: Size of the legend key symbols (number).

`legend_orientation_heatmap`
: should legend orientation be "horizontal" or "vertical".

`show_legend`
: Display the legend on the plot (flag).

`legend_position`
: Position of the legend ("right", "left", "bottom", "top").

`na_marker`
: Text used to mark `NA` values in numeric plots (string).

`na_marker_size`
: Size of the text marker for `NA` values (number).

`na_marker_colour`
: Color of the `NA` text marker (string).

`show_na_marker_categorical`
: Show a marker for `NA` and `Infinite` values on categorical tiles (flag).

`show_na_marker_heatmap`
: Show a marker for `NA` values on heatmap tiles (flag).

`colours_heatmap_low`
: Color for the lowest value in heatmaps (string).

`colours_heatmap_high`
: Color for the highest value in heatmaps (string).

`transform_heatmap`
: Transformation to apply before visualizing heatmap values ("identity", "log10", "log2").

`fontsize_values_heatmap`
: Font size for heatmap values (number).

`show_values_heatmap`
: Display numerical values on heatmap tiles (flag).

`colours_values_heatmap`
: Color for heatmap values (string).

`vertical_spacing`
: Space between each data row in points (number).

`numeric_plot_type`
: Type of visualization for numeric data: "bar" or "heatmap".

`y_axis_position`
: 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 ggstack, so they can customise how the info messages appear.

`inter_plot_spacing`
: How vertical space to add between plots. Measured in pts (numeric)

`expand_x`
: A vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `ggplot2::expansion()` to generate the values for the expand argument. The defaults are to expand the scale by 0.6 units on each side.

`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`
: Font size of the y axis titles (a.k.a the data.frame column names) (number).

`fontface_y_title`
: Font face of the y axis titles (a.k.a the data.frame column names). One of "plain", "italic", "bold", "bold.italic".

`beautify_text`
: Beautify y-axis text and legend titles to more human-readable forms (e.g. converting 'my_title' to 'My Title') (flag).

`beautify_values`
: Beautify legend values to more human-readable forms (e.g. converting 'my_value' to 'My Value') (flag)

`beautify_function`
: a function that takes a string and returns a nicely formatted string. Use to beautify axis & legend titles when `beautify_text=TRUE`, and legend values when `beautify_values=TRUE`.

`margin_y_title`
: Margin of y axis titles for discrete properties (or numeric properties when numeric_plot_type = "heatmap"). Expects NULL, or a call to `ggplot2::margin()`.

`margin_y_numbers`
: Margin of y axis titles and numbers and title of numeric properties (when numeric_plot_type = "bar"). Expects NULL, or a call to `ggplot2::margin()`.

Value

A list of visualization parameters for `ggstack`.

Examples

```
# Create Basic Plot
ggstack(baseballfans, col_id = "ID", col_sort = "Glasses")

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

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: (4piA)/(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, Dermosan, 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 `infotheo::discretize()`).
`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 descripts upper break position, lower describes lower break

Index