Package 'ggpedigree'

June 10, 2025

Title Visualizing Pedigrees with 'ggplot2' and 'plotly'

Version 0.7.0

Date/Publication 2025-06-10 00:10:02 UTC

Description Provides plotting functions for visualizing pedigrees in behavior genetics and kinship research. The package complements 'BGmisc' [Garrison et al. (2024) <doi:10.21105/joss.06203>] by rendering pedigrees using the 'ggplot2' framework and offers a modern alternative to the base-graphics pedigree plot in 'kinship2' [Sinnwell et al. (2014) <doi:10.1159/000363105>]. Features include support for duplicated individuals, complex mating structures, integration with simulated pedigrees, and layout customization.

License GPL-3

URL https://github.com/R-Computing-Lab/ggpedigree/, https://r-computing-lab.github.io/ggpedigree/

BugReports https://github.com/R-Computing-Lab/ggpedigree/issues

Depends R (>= 4.1.0)

Imports BGmisc (>= 1.4.1), kinship2, ggplot2, ggrepel, rlang, dplyr, stringr, utils, plotly, reshape2, scales, tidyr, paletteer

Suggests mockery, patchwork, viridis, knitr, tidyverse, purrr, data.table, discord, OpenMx, NlsyLinks, rmarkdown, tibble, corrplot, readxl, htmlwidgets, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Language en-US

LazyData true

Config/Needs/website rmarkdown

NeedsCompilation no

Author S. Mason Garrison [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-4804-6003>)

Maintainer S. Mason Garrison <garrissm@wfu.edu>

Repository CRAN

Contents

`buildPlotConfig` *build Config*

Description

This function builds a configuration list for ggPedigree plots. It merges a default configuration with user-specified settings, ensuring all necessary parameters are set.

Usage

```
buildPlotConfig(default_config, config, function_name = "ggPedigree")
```

Arguments

`default_config`	A list of default configuration parameters.
`config`	A list of user-specified configuration parameters.
`function_name`	The name of the function for which the configuration is being built.

Value

A complete configuration list with all necessary parameters.

`calculateConnections` *Calculate connections for a pedigree dataset*

Description

Computes graphical connection paths for a pedigree layout, including parent-child, sibling, and spousal connections. Optionally processes duplicate appearances of individuals (marked as 'extra') to ensure relational accuracy.

Usage

```
calculateConnections(
  ped,
  config = list(),
  spouseID = "spouseID",
  personID = "personID",
  momID = "momID",
  famID = "famID",
  twinID = "twinID",
  dadID = "dadID"
)
```

Arguments

`ped`	A data frame containing the pedigree data. Needs personID, momID, and dadID columns
`config`	List of configuration parameters. Currently unused but passed through to internal helpers.
`spouseID`	Character string specifying the column name for spouse IDs. Defaults to "spouseID".
`personID`	Character string specifying the column name for individual IDs.
`momID`	Character string specifying the column name for mother IDs. Defaults to "momID".
`famID`	Character string specifying the column name for family IDs.
`twinID`	Character string specifying the column name for twin IDs. Defaults to "twinID".
`dadID`	Character string specifying the column name for father IDs. Defaults to "dadID".

Value

A 'data.frame' containing connection points and midpoints for graphical rendering. Includes:

- 'x_pos', 'y_pos': positions of focal individual
- 'x_dad', 'y_dad', 'x_mom', 'y_mom': parental positions (if available)
- 'x_spouse', 'y_spouse': spousal positions (if available)

- ‘x_midparent‘, ‘y_midparent‘: midpoint between parents
- ‘x_mid_sib‘, ‘y_mid_sib‘: sibling group midpoint
- ‘x_mid_spouse‘, ‘y_mid_spouse‘: midpoint between spouses

`calculateCoordinates` *Calculate coordinates for plotting individuals in a pedigree*

Description

Extracts and modifies the x and y positions for each individual in a pedigree data frame using the align.pedigree function from the ‘kinship2‘ package. It returns a data.frame with positions for plotting.

Usage

```
calculateCoordinates(
  ped,
  personID = "personID",
  momID = "momID",
  dadID = "dadID",
  spouseID = "spouseID",
  sexVar = "sex",
  twinID = "twinID",
  code_male = NULL,
  config = list()
)
```

Arguments

`ped`	A data frame containing the pedigree data. Needs personID, momID, and dadID columns
`personID`	Character string specifying the column name for individual IDs.
`momID`	Character string specifying the column name for mother IDs. Defaults to "momID".
`dadID`	Character string specifying the column name for father IDs. Defaults to "dadID".
`spouseID`	Character. Name of the column in ‘ped‘ for the spouse ID variable.
`sexVar`	Character. Name of the column in ‘ped‘ for the sex variable.
`twinID`	Character string specifying the column name for twin IDs. Defaults to "twinID".
`code_male`	Value used to indicate male sex. Defaults to NULL.
`config`	List of configuration options: **code_male** Default is 1. Used by BGmisc::recodeSex(). **ped_packed** Logical, default TRUE. Passed to ‘kinship2::align.pedigree‘. **ped_align** Logical, default TRUE. Align generations. **ped_width** Numeric, default 15. Controls spacing.

Value

A data frame with one or more rows per person, each containing:

- ‘x_order‘, ‘y_order‘: Grid indices representing layout rows and columns.
- ‘x_pos‘, ‘y_pos‘: Continuous coordinate positions used for plotting.
- ‘nid‘: Internal numeric identifier for layout mapping.
- ‘extra‘: Logical flag indicating whether this row is a secondary appearance.

`computeCurvedMidpoint` *Compute midpoint coordinate for curved segment*

Description

Returns x and y midpoint using vectorized curved offset

Usage

```
computeCurvedMidpoint(x0, y0, x1, y1, curvature, angle, shift = 0, t = 0.5)
```

Arguments

`x0`	Numeric vector of x coordinates for start points
`y0`	Numeric vector of y coordinates for start points
`x1`	Numeric vector of x coordinates for end points
`y1`	Numeric vector of y coordinates for end points
`curvature`	Scalar curvature (geom_curve style)
`angle`	Scalar angle in degrees
`shift`	Scalar shift in degrees (default is 0)
`t`	Scalar value between 0 and 1 for interpolation (default is 0.5) setting the midpoint

Value

Numeric vector of midpoints (x or y) @keywords internal

computeDistance *Compute distance between two points*

Description

This function calculates the distance between two points in a 2D space using Minkowski distance. It can be used to compute Euclidean or Manhattan distance. It is a utility function for calculating distances in pedigree layouts. Defaults to Euclidean distance if no method is specified.

Usage

```
computeDistance(method = "euclidean", x1, y1, x2, y2, p = NULL)
```

Arguments

method	Character. Method of distance calculation. Options are "euclidean", "cityblock", and "Minkowski".
x1	Numeric. X-coordinate of the first point.
y1	Numeric. Y-coordinate of the first point.
x2	Numeric. X-coordinate of the second point.
y2	Numeric. Y-coordinate of the second point.
p	Numeric. The order of the Minkowski distance. If NULL, defaults to 2 for Euclidean and 1 for Manhattan. If Minkowski method is used, p should be specified.

countOffspring *Count offspring of each individual*

Description

Count offspring of each individual

Usage

```
countOffspring(ped, personID = "ID", momID = "momID", dadID = "dadID")
```

Arguments

ped	A data frame containing the pedigree information
personID	character. Name of the column in ped for the person ID variable
momID	character. Name of the column in ped for the mother ID variable
dadID	character. Name of the column in ped for the father ID variable

Value

A data frame with an additional column, offspring, that contains the number of offspring for each individual

Examples

```
library(BGmisc)
data("potter")
countOffspring(potter,
  personID = "personID",
  momID = "momID", dadID = "dadID"
)
```

`countSiblings` *Count siblings of each individual*

Description

Count siblings of each individual

Usage

```
countSiblings(ped, personID = "ID", momID = "momID", dadID = "dadID")
```

Arguments

`ped`	A data frame containing the pedigree information
`personID`	character. Name of the column in ped for the person ID variable
`momID`	character. Name of the column in ped for the mother ID variable
`dadID`	character. Name of the column in ped for the father ID variable

Value

A data frame with an additional column, siblings, that contains the number of siblings for each individual

Examples

```
library(BGmisc)
data("potter")
countSiblings(potter, personID = "personID")
```

`generateSpouseList` *Generate a spouselist matrix*

Description

Generate a spouselist matrix

Usage

```
generateSpouseList(
  ped,
  personID = "personID",
  momID = "momID",
  dadID = "dadID",
  spouseID = "spouseID"
)
```

Arguments

`ped`	A data frame containing the pedigree information
`personID`	Character. Name of the column in ped for the person ID variable
`momID`	Character. Name of the column in ped for the mother ID variable
`dadID`	Character. Name of the column in ped for the father ID variable
`spouseID`	Character. Name of the column in ped for the spouse ID variable

Value

A spouselist matrix

Examples

```
library(BGmisc)
data("potter")
generateSpouseList(potter,
  personID = "personID",
  momID = "momID", dadID = "dadID", spouseID = "spouseID"
)
```

`getDefaultPlotConfig` *Shared Default Plotting Configuration*

Description

Centralized configuration list used by all gg-based plotting functions. Returns a named list of default settings used by all gg-based plotting functions. This configuration can be overridden by supplying a list of key-value pairs to plotting functions such as ‘ggPedigree()‘, ‘ggRelatednessMatrix()‘, and ‘ggPhenotypeByDegree()‘. Each key corresponds to a configurable plot, layout, or aesthetic behavior.

Usage

```
getDefaultPlotConfig(
  function_name = "getDefaultPlotConfig",
  personID = "personID",
  status_column = NULL,
  alpha_default = 1,
  apply_default_scales = TRUE,
  apply_default_theme = TRUE,
  segment_default_color = "black",
  color_palette_default = c("#440154FF", "#FDE725FF", "#21908CFF"),
  color_palette_low = "#000004FF",
  color_palette_mid = "#56106EFF",
  color_palette_high = "#FCFDBFFF",
  color_scale_midpoint = 0.5,
  color_scale_theme = "ggthemes::calc",
  alpha = alpha_default,
  plot_title = NULL,
  plot_subtitle = NULL,
  value_rounding_digits = 5,
  code_male = 1,
  filter_n_pairs = 500,
  filter_degree_min = 0,
  filter_degree_max = 7,
  drop_classic_kin = FALSE,
  drop_non_classic_sibs = TRUE,
  use_only_classic_kin = TRUE,
  use_relative_degree = TRUE,
  group_by_kin = TRUE,
  match_threshold_percent = 10,
  max_degree_levels = 12,
  grouping_column = "mtdna_factor",
  annotate_include = TRUE,
  annotate_x_shift = -0.1,
  annotate_y_shift = 0.005,
  label_include = TRUE,
```

```
label_column = "personID",
label_method = "ggrepel",
label_max_overlaps = 15,
label_nudge_x = 0,
label_nudge_y = -0.1,
label_segment_color = NA,
label_text_angle = 0,
label_text_size = 2,
label_text_color = "black",
point_size = 4,
outline_include = FALSE,
outline_multiplier = 1.25,
outline_color = "black",
tooltip_include = TRUE,
tooltip_columns = c("ID1", "ID2", "value"),
axis_x_label = NULL,
axis_y_label = NULL,
axis_text_angle_x = 90,
axis_text_angle_y = 0,
axis_text_size = 8,
axis_text_color = "black",
generation_height = 1,
generation_width = 1,
ped_packed = TRUE,
ped_align = TRUE,
ped_width = 15,
segment_linewidth = 0.5,
segment_linetype = 1,
segment_lineend = "round",
segment_linejoin = "round",
segment_offspring_color = segment_default_color,
segment_parent_color = segment_default_color,
segment_self_color = segment_default_color,
segment_sibling_color = segment_default_color,
segment_spouse_color = segment_default_color,
segment_mz_color = segment_default_color,
segment_mz_linetype = 1,
segment_mz_alpha = 1,
segment_mz_t = 0.6,
segment_self_linetype = "dotdash",
segment_self_linewidth = 0.25,
segment_self_alpha = 0.5,
segment_self_angle = 90,
segment_self_curvature = -0.2,
sex_color_include = TRUE,
sex_legend_title = "Sex",
sex_shape_labels = c("Female", "Male", "Unknown"),
sex_color_palette = color_palette_default,
```

```
    sex_shape_female = 16,
    sex_shape_male = 15,
    sex_shape_unknown = 18,
    status_include = TRUE,
    status_code_affected = 1,
    status_code_unaffected = 0,
    status_label_affected = "Affected",
    status_label_unaffected = "Unaffected",
    status_alpha_affected = 1,
    status_alpha_unaffected = 0,
    status_color_palette = c(color_palette_default[1], color_palette_default[2]),
    status_color_affected = "black",
    status_color_unaffected = color_palette_default[2],
    status_shape_affected = 4,
    status_legend_title = "Affected",
    status_legend_show = FALSE,
    overlay_shape = 4,
    overlay_code_affected = 1,
    overlay_code_unaffected = 0,
    overlay_label_affected = "Affected",
    overlay_label_unaffected = "Unaffected",
    overlay_alpha_affected = 1,
    overlay_alpha_unaffected = 0,
    overlay_color = "black",
    overlay_include = FALSE,
    overlay_legend_title = "Overlay",
    overlay_legend_show = FALSE,
    focal_fill_include = FALSE,
    focal_fill_legend_show = TRUE,
    focal_fill_personID = 1,
    focal_fill_legend_title = "Focal Fill",
    focal_fill_high_color = "#FDE725FF",
    focal_fill_mid_color = "#9F2A63FF",
    focal_fill_low_color = "#0D082AFF",
    focal_fill_scale_midpoint = color_scale_midpoint,
    focal_fill_method = "gradient",
    focal_fill_component = "additive",
    focal_fill_n_breaks = NULL,
    focal_fill_na_value = "black",
    focal_fill_shape = 21,
    focal_fill_force_zero = FALSE,
    focal_fill_hue_range = c(0, 360),
    focal_fill_chroma = 50,
    focal_fill_lightness = 50,
    focal_fill_hue_direction = "horizontal",
    focal_fill_viridis_option = "D",
    focal_fill_viridis_begin = 0,
    focal_fill_viridis_end = 1,
```

```
  focal_fill_viridis_direction = 1,
  ci_include = TRUE,
  ci_ribbon_alpha = 0.3,
  tile_color_palette = c("white", "gold", "red"),
  tile_interpolate = TRUE,
  tile_color_border = NA,
  tile_cluster = TRUE,
  tile_geom = "geom_tile",
  matrix_diagonal_include = TRUE,
  matrix_upper_triangle_include = FALSE,
  matrix_lower_triangle_include = TRUE,
  matrix_sparse = FALSE,
  matrix_isChild_method = "partialparent",
  return_static = TRUE,
  return_widget = FALSE,
  return_interactive = FALSE,
  return_midparent = FALSE,
  debug = FALSE,
  override_many2many = FALSE,
  ...
)
```

Arguments

function_name	The name of the function calling this configuration.
personID	The column name for person identifiers in the data.
status_column	The column name for affected status in the data.
alpha_default	Default alpha transparency level.
apply_default_scales	Whether to apply default color scales.
apply_default_theme	Whether to apply default ggplot2 theme.
segment_default_color	A character string for the default color of segments in the plot.
color_palette_default	A character vector of default colors for the plot.
color_palette_low	Color for the low end of a gradient.
color_palette_mid	Color for the midpoint of a gradient.
color_palette_high	Color for the high end of a gradient.
color_scale_midpoint	Midpoint value for continuous color scales.
color_scale_theme	Name of the color scale used (e.g., "ggthemes::calc").

`alpha` Default alpha transparency for plot elements.

`plot_title` Main title of the plot.

`plot_subtitle` Subtitle of the plot.

`value_rounding_digits`
Number of digits to round displayed values.

`code_male` Integer code for males in data.

`filter_n_pairs` Threshold to filter maximum number of pairs.

`filter_degree_min`
Minimum degree value used in filtering.

`filter_degree_max`
Maximum degree value used in filtering.

`drop_classic_kin`
Whether to exclude classic kin categories.

`drop_non_classic_sibs`
Whether to exclude non-classic sibs.

`use_only_classic_kin`
Whether to restrict analysis to classic kinship.

`use_relative_degree`
Whether to use relative degrees instead of absolute.

`group_by_kin` Whether to group output by kinship group.

`match_threshold_percent`
Kinbin matching threshold as a percentage.

`max_degree_levels`
Maximum number of degree levels to show.

`grouping_column`
Name of column used for grouping.

`annotate_include`
Whether to include annotations.

`annotate_x_shift`
Horizontal shift applied to annotation text.

`annotate_y_shift`
Vertical shift applied to annotation text.

`label_include` Whether to display labels on plot points.

`label_column` Column to use for text labels.

`label_method` Method used for labeling (e.g., ggrepel, geom_text).

`label_max_overlaps`
Maximum number of overlapping labels.

`label_nudge_x` Horizontal nudge for label text.

`label_nudge_y` Vertical nudge for label text.

`label_segment_color`
Segment color for label connectors.

`label_text_angle`
Text angle for labels.

`label_text_size`
: Font size for labels.

`label_text_color`
: Color of the label text.

`point_size`
: Size of points drawn in plot.

`outline_include`
: Whether to include outlines around points.

`outline_multiplier`
: Multiplier to compute outline size from point size.

`outline_color`
: Color used for point outlines.

`tooltip_include`
: Whether tooltips are shown in interactive plots.

`tooltip_columns`
: Columns to include in tooltips.

`axis_x_label`
: Label for the X-axis.

`axis_y_label`
: Label for the Y-axis.

`axis_text_angle_x`
: Angle of X-axis text.

`axis_text_angle_y`
: Angle of Y-axis text.

`axis_text_size`
: Font size of axis text.

`axis_text_color`
: Color of axis text.

`generation_height`
: Vertical spacing of generations.

`generation_width`
: Horizontal spacing of generations.

`ped_packed`
: Whether the pedigree should use packed layout.

`ped_align`
: Whether to align pedigree generations.

`ped_width`
: Plot width of the pedigree block.

`segment_linewidth`
: Line width for segments.

`segment_linetype`
: Line type for segments.

`segment_lineend`
: Line end type for segments.

`segment_linejoin`
: Line join type for segments.

`segment_offspring_color`
: Color for offspring segments.

`segment_parent_color`
: Color for parent segments.

`segment_self_color`
: Color for self-loop segments.

segment_sibling_color
: Color for sibling segments.

segment_spouse_color
: Color for spouse segments.

segment_mz_color
: Color for monozygotic twin segments.

segment_mz_linetype
: Line type for MZ segments.

segment_mz_alpha
: Alpha for MZ segments.

segment_mz_t
: Tuning parameter for MZ segment layout.

segment_self_linetype
: Line type for self-loop segments.

segment_self_linewidth
: Width of self-loop segment lines.

segment_self_alpha
: Alpha value for self-loop segments.

segment_self_angle
: Angle of self-loop segment.

segment_self_curvature
: Curvature of self-loop segment.

sex_color_include
: Whether to color nodes by sex.

sex_legend_title
: Title of the sex legend.

sex_shape_labels
: Labels used in sex legend.

sex_color_palette
: A character vector of colors for sex.

sex_shape_female
: Shape for female nodes.

sex_shape_male
: Shape for male nodes.

sex_shape_unknown
: Shape for unknown sex nodes.

status_include
: Whether to display affected status.

status_code_affected
: Value that encodes affected status.

status_code_unaffected
: Value that encodes unaffected status.

status_label_affected
: Label for affected status.

status_label_unaffected
: Label for unaffected status.

status_alpha_affected
: Alpha for affected individuals.

`status_alpha_unaffected`
: Alpha for unaffected individuals. Default is 0 (transparent).

`status_color_palette`
: A character vector of colors for affected status.

`status_color_affected`
: Color for affected individuals.

`status_color_unaffected`
: Color for unaffected individuals.

`status_shape_affected`
: Shape for affected individuals.

`status_legend_title`
: Title of the status legend.

`status_legend_show`
: Whether to show the status legend.

`overlay_shape` Shape used for overlaying points in the plot. Default is 4 (cross).

`overlay_code_affected`
: Code for affected individuals in overlay. Default is 1.

`overlay_code_unaffected`
: Code for unaffected individuals in overlay. Default is 0.

`overlay_label_affected`
: Label for affected individuals in overlay. Default is "Affected".

`overlay_label_unaffected`
: Label for unaffected individuals in overlay. Default is "Unaffected".

`overlay_alpha_affected`
: Alpha for affected individuals in overlay. Default is 1.

`overlay_alpha_unaffected`
: Alpha for unaffected individuals in overlay. Default is 0.

`overlay_color` Color for overlay points. Default is "black".

`overlay_include`
: Whether to include overlay points in the plot. Default is FALSE.

`overlay_legend_title`
: Title of the overlay legend. Default is "Overlay".

`overlay_legend_show`
: Whether to show the overlay legend. Default is FALSE.

`focal_fill_include`
: Whether to fill focal individuals.

`focal_fill_legend_show`
: Whether to show legend for focal fill.

`focal_fill_personID`
: ID of focal individual.

`focal_fill_legend_title`
: Title of focal fill legend.

`focal_fill_high_color`
: High-end color for focal gradient.

`focal_fill_mid_color`
: Midpoint color for focal gradient.

`focal_fill_low_color`
: Low-end color for focal gradient.

`focal_fill_scale_midpoint`
: Midpoint for focal fill scale.

`focal_fill_method`
: Method used for focal fill gradient.

`focal_fill_component`
: Component type for focal fill.

`focal_fill_n_breaks`
: Number of breaks in focal fill scale.

`focal_fill_na_value`
: Color for NA values in focal fill.

`focal_fill_shape`
: Shape used for focal fill points.

`focal_fill_force_zero`
: Whether to force zero to NA in focal fill.

`focal_fill_hue_range`
: Hue range for focal fill colors.

`focal_fill_chroma`
: Chroma value for focal fill colors.

`focal_fill_lightness`
: Lightness value for focal fill colors.

`focal_fill_hue_direction`
: Direction of focal fill gradient.

`focal_fill_viridis_option`
: Option for viridis color scale.

`focal_fill_viridis_begin`
: Start of viridis color scale.

`focal_fill_viridis_end`
: End of viridis color scale.

`focal_fill_viridis_direction`
: Direction of viridis color scale (1 for left to right, -1 for right to left).

`ci_include` Whether to show confidence intervals.

`ci_ribbon_alpha`
: Alpha level for CI ribbons.

`tile_color_palette`
: Color palette for matrix plots.

`tile_interpolate`
: Whether to interpolate colors in matrix tiles.

`tile_color_border`
: Color border for matrix tiles.

`tile_cluster` Whether to sort by clusters the matrix.

`tile_geom` Geometry type for matrix tiles (e.g., "geom_tile", "geom_raster").

`matrix_diagonal_include`
Whether to include diagonal in matrix plots.

`matrix_upper_triangle_include`
Whether to include upper triangle in matrix plots.

`matrix_lower_triangle_include`
Whether to include lower triangle in matrix plots.

`matrix_sparse` Whether matrix input is sparse.

`matrix_isChild_method`
Method used for isChild matrix derivation.

`return_static` Whether to return a static plot.

`return_widget` Whether to return a widget object.

`return_interactive`
Whether to return an interactive plot.

`return_midparent`
Whether to return midparent values in the plot.

`debug` Whether to enable debugging mode.

`override_many2many`
Whether to override many-to-many link logic.

`...` Additional arguments for future extensibility.

Value

A named list of default plotting and layout parameters.

`ggPedigree` *Plot a custom pedigree diagram*

Description

Generates a ggplot2-based diagram of a pedigree using custom coordinate layout, calculated relationship connections, and flexible styling via ‘config‘. It processes the data using ‘ped2fam()‘. This function supports multiple families and optionally displays affected status and sex-based color/shape.

Usage

```
ggPedigree(
  ped,
  famID = "famID",
  personID = "personID",
  momID = "momID",
  dadID = "dadID",
  spouseID = "spouseID",
  matID = "matID",
```

```
  patID = "patID",
  twinID = "twinID",
  status_column = NULL,
  focal_fill_column = NULL,
  tooltip_columns = NULL,
  overlay_column = NULL,
  return_widget = FALSE,
  config = list(),
  debug = FALSE,
  hints = NULL,
  interactive = FALSE,
  ...
)

ggpedigree(
  ped,
  famID = "famID",
  personID = "personID",
  momID = "momID",
  dadID = "dadID",
  spouseID = "spouseID",
  matID = "matID",
  patID = "patID",
  twinID = "twinID",
  status_column = NULL,
  focal_fill_column = NULL,
  tooltip_columns = NULL,
  overlay_column = NULL,
  return_widget = FALSE,
  config = list(),
  debug = FALSE,
  hints = NULL,
  interactive = FALSE,
  ...
)
```

Arguments

ped	A data frame containing the pedigree data. Needs personID, momID, and dadID columns
famID	Character string specifying the column name for family IDs.
personID	Character string specifying the column name for individual IDs.
momID	Character string specifying the column name for mother IDs. Defaults to "momID".
dadID	Character string specifying the column name for father IDs. Defaults to "dadID".
spouseID	Character string specifying the column name for spouse IDs. Defaults to "spouseID".

`matID` Character string specifying the column name for maternal lines Defaults to "matID".

`patID` Character string specifying the column name for paternal lines Defaults to "patID".

`twinID` Character string specifying the column name for twin IDs. Defaults to "twinID".

`status_column` Character string specifying the column name for affected status. Defaults to NULL.

`focal_fill_column`
Character string specifying the column name for focal fill color.

`tooltip_columns`
Character vector of column names to show when hovering. Defaults to c("personID", "sex"). Additional columns present in 'ped' can be supplied – they will be added to the Plotly tooltip text. Defaults to NULL, which uses the default tooltip columns.

`overlay_column` Character string specifying the column name for overlay alpha values.

`return_widget` Logical; if TRUE (default) returns a plotly htmlwidget. If FALSE, returns the underlying plotly object (useful for further customization before printing).

`config` A list of configuration options for customizing the plot. The list can include:

code_male Integer or string. Value identifying males in the sex column. (typically 0 or 1) Default: 1.

segment_spouse_color, segment_self_color Character. Line colors for respective connection types.

segment_sibling_color, segment_parent_color, segment_offspring_color Character. Line colors for respective connection types.

label_text_size, point_size, segment_linewidth Numeric. Controls text size, point size, and line thickness.

generation_height Numeric. Vertical spacing multiplier between generations. Default: 1.

shape_unknown, shape_female, shape_male, status_shape_affected Integers. Shape codes for plotting each group.

sex_shape_labels Character vector of labels for the sex variable. (default: c("Female", "Male", "Unknown")

unaffected, affected Values indicating unaffected/affected status.

sex_color_include Logical. If TRUE, uses color to differentiate sex.

label_max_overlaps Maximum number of overlaps allowed in repelled labels.

label_segment_color Color used for label connector lines.

`debug` Logical. If TRUE, prints debugging information. Default: FALSE.

`hints` Data frame with hints for layout adjustments. Default: NULL.

`interactive` Logical. If TRUE, generates an interactive plot using 'plotly'. Default: FALSE.

`...` Additional arguments passed to 'ggplot2' functions.

Value

A 'ggplot' object rendering the pedigree diagram.

Examples

```
library(BGmisc)
data("potter")
ggPedigree(potter, famID = "famID", personID = "personID")

data("hazard")
ggPedigree(hazard, famID = "famID", personID = "ID", config = list(code_male = 0))
```

---

ggPedigreeInteractive *Interactive pedigree plot (Plotly wrapper around ggPedigree)*

Description

Generates an interactive HTML widget built on top of the static ggPedigree output. All layout, styling, and connection logic are inherited from ggPedigree(); this function simply augments the plot with Plotly hover, zoom, and pan functionality.

Usage

```
ggPedigreeInteractive(
  ped,
  famID = "famID",
  personID = "personID",
  momID = "momID",
  dadID = "dadID",
  patID = "patID",
  matID = "matID",
  twinID = "twinID",
  status_column = NULL,
  tooltip_columns = NULL,
  focal_fill_column = NULL,
  overlay_column = NULL,
  config = list(),
  debug = FALSE,
  return_widget = TRUE,
  ...
)

ggpedigreeinteractive(
  ped,
  famID = "famID",
  personID = "personID",
  momID = "momID",
  dadID = "dadID",
  patID = "patID",
  matID = "matID",
```

```
  twinID = "twinID",
  status_column = NULL,
  tooltip_columns = NULL,
  focal_fill_column = NULL,
  overlay_column = NULL,
  config = list(),
  debug = FALSE,
  return_widget = TRUE,
  ...
)

ggpedigreeInteractive(
  ped,
  famID = "famID",
  personID = "personID",
  momID = "momID",
  dadID = "dadID",
  patID = "patID",
  matID = "matID",
  twinID = "twinID",
  status_column = NULL,
  tooltip_columns = NULL,
  focal_fill_column = NULL,
  overlay_column = NULL,
  config = list(),
  debug = FALSE,
  return_widget = TRUE,
  ...
)
```

Arguments

ped	A data frame containing the pedigree data. Needs personID, momID, and dadID columns
famID	Character string specifying the column name for family IDs.
personID	Character string specifying the column name for individual IDs.
momID	Character string specifying the column name for mother IDs. Defaults to "momID".
dadID	Character string specifying the column name for father IDs. Defaults to "dadID".
patID	Character string specifying the column name for paternal lines Defaults to "patID".
matID	Character string specifying the column name for maternal lines Defaults to "matID".
twinID	Character string specifying the column name for twin IDs. Defaults to "twinID".
status_column	Character string specifying the column name for affected status. Defaults to NULL.

`tooltip_columns`
: Character vector of column names to show when hovering. Defaults to c("personID", "sex"). Additional columns present in 'ped' can be supplied – they will be added to the Plotly tooltip text. Defaults to NULL, which uses the default tooltip columns.

`focal_fill_column`
: Character string specifying the column name for focal fill color.

`overlay_column`
: Character string specifying the column name for overlay alpha values.

`config`
: A list of configuration options for customizing the plot. The list can include:

  code_male Integer or string. Value identifying males in the sex column. (typically 0 or 1) Default: 1.

  segment_spouse_color, segment_self_color Character. Line colors for respective connection types.

  segment_sibling_color, segment_parent_color, segment_offspring_color Character. Line colors for respective connection types.

  label_text_size, point_size, segment_linewidth Numeric. Controls text size, point size, and line thickness.

  generation_height Numeric. Vertical spacing multiplier between generations. Default: 1.

  shape_unknown, shape_female, shape_male, status_shape_affected Integers. Shape codes for plotting each group.

  sex_shape_labels Character vector of labels for the sex variable. (default: c("Female", "Male", "Unknown")

  unaffected, affected Values indicating unaffected/affected status.

  sex_color_include Logical. If TRUE, uses color to differentiate sex.

  label_max_overlaps Maximum number of overlaps allowed in repelled labels.

  label_segment_color Color used for label connector lines.

`debug`
: Logical. If TRUE, prints debugging information. Default: FALSE.

`return_widget`
: Logical; if TRUE (default) returns a plotly htmlwidget. If FALSE, returns the underlying plotly object (useful for further customization before printing).

`...`
: Additional arguments passed to 'ggplot2' functions.

Value

A plotly htmlwidget (or plotly object if 'return_widget = FALSE').

Examples

```
library(BGmisc)
data("potter")
ggPedigreeInteractive(potter, famID = "famID", personID = "personID")
```

`ggPhenotypeByDegree` *Plot correlation of genetic relatedness by phenotype*

Description

This function plots the phenotypic correlation as a function of genetic relatedness.

Usage

```
ggPhenotypeByDegree(
  df,
  y_var,
  y_se = NULL,
  y_stem_se = NULL,
  y_ci_lb = NULL,
  y_ci_ub = NULL,
  config = list(),
  data_prep = TRUE,
  ...
)
```

Arguments

`df`	Data frame containing pairwise summary statistics. Required columns: **addRel_min** Minimum relatedness per group **addRel_max** Maximum relatedness per group **n_pairs** Number of pairs at that relatedness **cnu** Indicator for shared nuclear environment (1 = yes, 0 = no) **mtdna** Indicator for shared mitochondrial DNA (1 = yes, 0 = no)
`y_var`	Name of the y-axis variable column (e.g., "r_pheno_rho").
`y_se`	Name of the standard error column (e.g., "r_pheno_se").
`y_stem_se`	Optional; base stem used to construct SE ribbon bounds. (e.g., "r_pheno")
`y_ci_lb`	Optional; lower bound for confidence interval (e.g., "r_pheno_ci_lb").
`y_ci_ub`	Optional; upper bound for confidence interval (e.g., "r_pheno_ci_ub").
`config`	A list of configuration overrides. Valid entries include: **filter_n_pairs** Minimum number of pairs to include (default: 500) **filter_degree_min** Minimum degree of relatedness (default: 0) **filter_degree_max** Maximum degree of relatedness (default: 7) **plot_title** Plot title **plot_subtitle** Plot subtitle **color_scale** Paletteer color scale name (e.g., "ggthemes::calc") **use_only_classic_kin** If TRUE, only classic kin are shown **group_by_kin** If TRUE, use classic kin × mtDNA for grouping

drop_classic_kin If TRUE, remove classic kin rows
drop_non_classic_sibs If TRUE, remove non-classic sibs (default: TRUE)
annotate_include If TRUE, annotate mother/father/sibling points
annotate_x_shift Relative x-axis shift for annotations
annotate_y_shift Relative y-axis shift for annotations
point_size Size of geom_point points (default: 1)
use_relative_degree If TRUE, x-axis uses degree-of-relatedness scaling
grouping_column Grouping column name (default: mtdna_factor)
value_rounding_digits Number of decimal places for rounding (default: 2)
match_threshold_percent Tolerance % for matching known degrees
max_degree_levels Maximum number of degrees to consider

`data_prep` Logical; if TRUE, performs data preparation steps.

`...` Additional arguments passed to ‘ggplot2‘ functions.

Value

A ggplot object containing the correlation plot.

`ggRelatednessMatrix` *Plot a relatedness matrix as a heatmap (ggpedigree style)*

Description

Plots a relatedness matrix using ggplot2 with config options.

Usage

```
ggRelatednessMatrix(
  mat,
  config = list(),
  interactive = FALSE,
  tooltip_columns = NULL,
  personID = "personID",
  ...
)

ggrelatednessmatrix(
  mat,
  config = list(),
  interactive = FALSE,
  tooltip_columns = NULL,
  personID = "personID",
  ...
)
```

Arguments

`mat`	A square numeric matrix of relatedness values (precomputed, e.g., from ped2add).
`config`	A list of graphical and display parameters. See Details for available options.
`interactive`	Logical; if TRUE, returns an interactive plotly object.
`tooltip_columns`	A character vector of column names to include in tooltips.
`personID`	Character; name of the column containing unique person identifiers.
`...`	Additional arguments passed to ggplot2 layers.

Details

Config options include:

matrix_color_palette A vector of colors for the heatmap (default: Reds scale)

color_scale_midpoint Numeric midpoint for diverging color scale (default: 0.25)

plot_title Plot title

matrix_cluster Logical; should rows/cols be clustered (default: TRUE)

axis_x_label, axis_y_label Axis labels

axis_text_size Axis text size

Value

A ggplot object displaying the relatedness matrix as a heatmap.

Examples

```
# Example relatedness matrix
set.seed(123)
mat <- matrix(runif(100, 0, 1), nrow = 10)
rownames(mat) <- paste0("ID", 1:10)
colnames(mat) <- paste0("ID", 1:10)

# Plot the relatedness matrix
ggRelatednessMatrix(mat,
  config = list(
    matrix_color_palette = c("white", "gold", "red"),
    color_scale_midpoint = 0.5,
    matrix_cluster = TRUE,
    plot_title = "Relatedness Matrix",
    axis_x_label = "Individuals",
    axis_y_label = "Individuals",
    axis_text_size = 8
  )
)
```

plotPedigree *plotPedigree A wrapped function to plot simulated pedigree from function* simulatePedigree. *This function require the installation of package* kinship2.

Description

plotPedigree A wrapped function to plot simulated pedigree from function simulatePedigree. This function require the installation of package kinship2.

Usage

```
plotPedigree(
  ped,
  code_male = NULL,
  verbose = FALSE,
  affected = NULL,
  cex = 0.5,
  col = 1,
  symbolsize = 1,
  branch = 0.6,
  packed = TRUE,
  align = c(1.5, 2),
  width = 8,
  density = c(-1, 35, 65, 20),
  mar = c(2.1, 1, 2.1, 1),
  angle = c(90, 65, 40, 0),
  keep.par = FALSE,
  pconnect = 0.5,
  ...
)
```

Arguments

ped	The simulated pedigree data.frame from function simulatePedigree. Or a pedigree dataframe with the same colnames as the dataframe simulated from function simulatePedigree.
code_male	This optional input allows you to indicate what value in the sex variable codes for male. Will be recoded as "M" (Male). If NULL, no recoding is performed.
verbose	logical If TRUE, prints additional information. Default is FALSE.
affected	This optional parameter can either be a string specifying the column name that indicates affected status or a numeric/logical vector of the same length as the number of rows in 'ped'. If NULL, no affected status is assigned.
cex	The font size of the IDs for each individual in the plot.
col	color for each id. Default assigns the same color to everyone.

symbolsize	controls symbolsize. Default=1.
branch	defines how much angle is used to connect various levels of nuclear families.
packed	default=T. If T, uniform distance between all individuals at a given level.
align	these parameters control the extra effort spent trying to align children underneath parents, but without making the pedigree too wide. Set to F to speed up plotting.
width	default=8. For a packed pedigree, the minimum width allowed in the realignment of pedigrees.
density	defines density used in the symbols. Takes up to 4 different values.
mar	margin parmeters, as in the par function
angle	defines angle used in the symbols. Takes up to 4 different values.
keep.par	Default = F, allows user to keep the parameter settings the same as they were for plotting (useful for adding extras to the plot)
pconnect	when connecting parent to children the program will try to make the connecting line as close to vertical as possible, subject to it lying inside the endpoints of the line that connects the children by at least pconnect people. Setting this option to a large number will force the line to connect at the midpoint of the children.
...	Extra options that feed into the plot function.

Value

A plot of the provided pedigree

---

redsquirrels *Kluane Red Squirrel Data*

Description

A tidy data frame of life-history and reproductive metrics for 7,799 individual red squirrels from the Kluane Red Squirrel Project (1987–present). Each row corresponds to one squirrel with associated pedigree links and reproductive success summaries. The original data are published under a CC0 1.0 Universal Public Domain Dedication:

Usage

```
data(redsquirrels)
```

Format

'redsquirrels' A data frame with 7799 rows and 16 columns:

personID Unique identifier for each squirrel

momID, dadID Unique identifiers for each squirrel's parents

sex Biological sex of the squirrel

famID Unique identifier for each family. Derived from ped2fam

byear Birth year of the squirrel

dyear Death year of the squirrel

lrs lifetime reproductive success for the squirrel

ars_mean Mean annual reproductive success for the squirrel

ars_max Maximum ARS value for the squirrel

ars_med Median ARS value for the squirrel

ars_min Minimum ARS value for the squirrel

ars_sd Standard deviation of ARS values for the squirrel

ars_n Number of ARS values for the squirrel

year_first First year of ARS data for the squirrel

year_last Last year of ARS data for the squirrel ...

Details

McFarlane, S. Eryn; Boutin, Stan; Humphries, Murray M. et al. (2015). Data from: Very low levels of direct additive genetic variance in fitness and fitness components in a red squirrel population [Dataset]. Dryad. <https://doi.org/10.5061/dryad.n5q05>

Source

<https://doi.org/10.5061/dryad.n5q05>

Index