Package ‘ggpedigree’

May 23, 2025

Title Visualizing Pedigrees with 'ggplot2' and 'plotly'

Version 0.4.0

Date/Publication 2025-05-23 18:22:01 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, kinship2, ggplot2, ggrepel, rlang, dplyr, stringr, utils, plotly, reshape2

Suggests EasyMx, viridis, knitr, tidyverse, 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

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

`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())
```

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.

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 = "ID",
  momID = "momID",
  dadID = "dadID",
  spouseID = "spouseID",
  sexVar = "sex",
  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.
`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.

`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"
)
```

`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",
  status_col = NULL,
  tooltip_cols = NULL,
  as_widget = FALSE,
  config = list(),
  debug = FALSE,
  hints = NULL,
  interactive = FALSE,
  ...
)

ggpedigree(
  ped,
  famID = "famID",
  personID = "personID",
  momID = "momID",
  dadID = "dadID",
  status_col = NULL,
  tooltip_cols = NULL,
  as_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".
`status_col`	Character string specifying the column name for affected status. Defaults to NULL.
`tooltip_cols`	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.
`as_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, segment_sibling_color, segment_parent_color, segment_** Character. Line colors for respective connection types. **label_text_size, point_size, line_width** 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, affected_shape** Integers. Shape codes for plotting each group. **sex_shape_labs** Character vector of labels for the sex variable. (default: c("Female", "Male", "Unknown") **unaffected, affected** Values indicating unaffected/affected status. **sex_color** 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",
  status_col = NULL,
  tooltip_cols = NULL,
  config = list(),
  debug = FALSE,
  as_widget = TRUE,
  ...
)

ggpedigreeinteractive(
  ped,
  famID = "famID",
  personID = "personID",
  momID = "momID",
  dadID = "dadID",
  status_col = NULL,
  tooltip_cols = NULL,
  config = list(),
```

```
    debug = FALSE,
    as_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".
`status_col`	Character string specifying the column name for affected status. Defaults to NULL.
`tooltip_cols`	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.
`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, segment_sibling_color, segment_parent_color, segment_** Character. Line colors for respective connection types. **label_text_size, point_size, line_width** 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, affected_shape** Integers. Shape codes for plotting each group. **sex_shape_labs** Character vector of labels for the sex variable. (default: c("Female", "Male", "Unknown") **unaffected, affected** Values indicating unaffected/affected status. **sex_color** 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.
`as_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 ‘as_widget = FALSE‘).

Examples

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

`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_cols = NULL,
  ...
)

ggrelatednessmatrix(
  mat,
  config = list(),
  interactive = FALSE,
  tooltip_cols = NULL,
  ...
)
```

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_cols`	A character vector of column names to include in tooltips.
`...`	Additional arguments passed to ggplot2 layers.

Details

Config options include:

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

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

title Plot title

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

xlab, ylab Axis labels

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(
    color_palette = c("white", "gold", "red"),
    scale_midpoint = 0.5,
    cluster = TRUE,
    title = "Relatedness Matrix",
    xlab = "Individuals",
    ylab = "Individuals",
    text_size = 8
  )
)
```

`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