Package ‘discord’

June 10, 2025

Type Package

Title Functions for Discordant Kinship Modeling

Version 1.2.4.1

Description Functions for discordant kinship modeling (and other sibling-based quasi-experimental designs). Contains data restructuring functions and functions for generating biometrically informed data for kin pairs. See [Garrison and Rodgers, 2016 <doi:10.1016/j.intell.2016.08.008>], [Sims, Trattner, and Garrison, 2024 <doi:10.3389/fpsyg.2024.1430978>] for empirical examples, and Garrison and colleagues for theoretical work <`https://osf.io/zpdwt/`>.

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

License GPL-3

LazyData TRUE

RoxygenNote 7.3.2

Encoding UTF-8

Depends R (>= 3.50)

Imports stats

Suggests NlsyLinks, ggpedigree, BGmisc, broom, dplyr, grid, gridExtra, ggplot2, janitor, kableExtra, knitr, magrittr, rmarkdown, scales, stargazer, snakecase, testthat, tidyverse

VignetteBuilder knitr

NeedsCompilation no

Author S. Mason Garrison [aut, cre, cph] (ORCID: <`https://orcid.org/0000-0002-4804-6003`>),
Jonathan Trattner [aut] (ORCID: <`https://orcid.org/0000-0002-1097-7603`>, url: https://www.jdtrat.com/),
Yoo Ri Hwang [aut],
Cermet Ream [ctb]

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

Repository CRAN

Date/Publication 2025-06-10 16:30:02 UTC

Contents

`check_discord_errors` *Check Discord Errors*

Description

This function checks for common errors in the provided data, including the correct specification of identifiers (ID, sex, race) and their existence in the data.

Usage

```
check_discord_errors(data, id, sex, race, pair_identifiers)
```

Arguments

`data`	The data to perform a discord regression on.
`id`	A unique kinship pair identifier.
`sex`	A character string for the sex column name.
`race`	A character string for the race column name.
`pair_identifiers`	A character vector of length two that contains the variable identifier for each kinship pair.

Value

An error message if one of the conditions are met.

`check_sibling_order` *Check Sibling Order*

Description

This function determines the order of sibling pairs based on an outcome variable. The function checks which of the two kinship pairs has more of a specified outcome variable. It adds a new column named ‘order‘ to the dataset, indicating which sibling (identified as "s1" or "s2") has more of the outcome. If the two siblings have the same amount of the outcome, it randomly assigns one as having more.

Usage

```
check_sibling_order(..., fast = FALSE)
```

Arguments

`...`	Additional arguments to be passed to the function.
`fast`	Logical. If TRUE, uses a faster method for data processing.

Value

A one-row data frame with a new column order indicating which familial member (1, 2, or neither) has more of the outcome.

`data_flu_ses` *Flu Vaccination and SES Data*

Description

A data frame that accompanies the regression vignette. It contains data on SES and flu vaccination.

Usage

```
data_flu_ses
```

Format

A data frame.

Kinship pairs and their relatedness, SES, and flu vaccination information.

Source

NLSY/R Lab

`data_sample` *Sample Data from NLSY*

Description

A data frame output from the NlsyLinks package that contains data for kinship pairs' height and weight.

Usage

```
data_sample
```

Format

A data frame.

Kinship pairs and their relatedness, height, and weight information.

Source

NLSY/R Lab

`discord_between_model` *Perform a Between-Family Linear Regression within the Discordant Kinship Framework*

Description

Perform a Between-Family Linear Regression within the Discordant Kinship Framework

Usage

```
discord_between_model(
  data,
  outcome,
  predictors,
  demographics = NULL,
  id = NULL,
  sex = "sex",
  race = "race",
  pair_identifiers = c("_s1", "_s2"),
  data_processed = FALSE,
  coding_method = "none",
  fast = TRUE
)
```

Arguments

`data`	The data set with kinship pairs
`outcome`	A character string containing the outcome variable of interest.
`predictors`	A character vector containing the column names for predicting the outcome.
`demographics`	Indicator variable for if the data has the sex and race demographics. If both are present (default, and recommended), value should be "both". Other options include "sex", "race", or "none".
`id`	Default’s to NULL. If supplied, must specify the column name corresponding to unique kinship pair identifiers.
`sex`	A character string for the sex column name.
`race`	A character string for the race column name.
`pair_identifiers`	A character vector of length two that contains the variable identifier for each kinship pair
`data_processed`	Logical operator if data are already preprocessed by discord_data , default is FALSE
`coding_method`	A character string that indicates what kind of additional coding schemes should be used. Default is none. Other options include "binary" and "multi".
`fast`	Logical. If TRUE, uses a faster method for data processing.

Value

Resulting ‘lm‘ object from performing the between-family regression.

Examples

```
discord_between_model(
  data = data_sample,
  outcome = "height",
  predictors = "weight",
  pair_identifiers = c("_s1", "_s2"),
  sex = NULL,
  race = NULL
)
```

`discord_cond` *Custom Conditions for the discord package*

Description

Custom Conditions for the discord package

Usage

```
discord_cond(type, msg, class = paste0("discord-", type), call = NULL, ...)
```

Arguments

`type`	One of the following conditions: c("error", "warning", "message")
`msg`	Message
`class`	Default is to prefix the 'type' argument with "discord", but can be more specific to the problem at hand.
`call`	What triggered the condition?
`...`	Additional arguments that can be coerced to character or single condition object.

Value

A condition for discord.

Examples

```
## Not run:

derr <- function(x) discord_cond("error", x)
dwarn <- function(x) discord_cond("warning", x)
dmess <- function(x) discord_cond("message", x)

return_class <- function(func) {
  tryCatch(func,
    error = function(cond) class(cond),
    warning = function(cond) class(cond),
    message = function(cond) class(cond)
  )
}

return_class(derr("error-class"))
return_class(dwarn("warning-class"))
return_class(dmess("message-class"))

## End(Not run)
```

`discord_data` *Restructure Data to Determine Kinship Differences*

Description

Restructure Data to Determine Kinship Differences

Usage

```
discord_data(
  data,
  outcome,
  predictors,
  id = NULL,
  sex = "sex",
  race = "race",
  pair_identifiers,
  demographics = "both",
  coding_method = "none",
  fast = TRUE,
  ...
)
```

Arguments

`data`	The data set with kinship pairs
`outcome`	A character string containing the outcome variable of interest.
`predictors`	A character vector containing the column names for predicting the outcome.
`id`	Default's to NULL. If supplied, must specify the column name corresponding to unique kinship pair identifiers.
`sex`	A character string for the sex column name.
`race`	A character string for the race column name.
`pair_identifiers`	A character vector of length two that contains the variable identifier for each kinship pair
`demographics`	Indicator variable for if the data has the sex and race demographics. If both are present (default, and recommended), value should be "both". Other options include "sex", "race", or "none".
`coding_method`	A character string that indicates what kind of additional coding schemes should be used. Default is none. Other options include "binary" and "multi".
`fast`	Logical. If TRUE, uses a faster method for data processing.
`...`	Additional arguments to be passed to the function.

Value

A data frame that contains analyzable, paired data for performing kinship regressions.

Examples

```
discord_data(
  data = data_sample,
  outcome = "height",
  predictors = "weight",
  pair_identifiers = c("_s1", "_s2"),
```

```
  sex = NULL,
  race = NULL,
  demographics = "none"
)
```

`discord_regression` *Perform a Linear Regression within the Discordant Kinship Framework*

Description

Perform a Linear Regression within the Discordant Kinship Framework

Usage

```
discord_regression(
  data,
  outcome,
  predictors,
  demographics = NULL,
  id = NULL,
  sex = "sex",
  race = "race",
  pair_identifiers = c("_s1", "_s2"),
  data_processed = FALSE,
  coding_method = "none",
  fast = TRUE
)

discord_within_model(
  data,
  outcome,
  predictors,
  demographics = NULL,
  id = NULL,
  sex = "sex",
  race = "race",
  pair_identifiers = c("_s1", "_s2"),
  data_processed = FALSE,
  coding_method = "none",
  fast = TRUE
)
```

Arguments

`data`	The data set with kinship pairs

`outcome`	A character string containing the outcome variable of interest.
`predictors`	A character vector containing the column names for predicting the outcome.
`demographics`	Indicator variable for if the data has the sex and race demographics. If both are present (default, and recommended), value should be "both". Other options include "sex", "race", or "none".
`id`	Default's to NULL. If supplied, must specify the column name corresponding to unique kinship pair identifiers.
`sex`	A character string for the sex column name.
`race`	A character string for the race column name.
`pair_identifiers`	A character vector of length two that contains the variable identifier for each kinship pair
`data_processed`	Logical operator if data are already preprocessed by discord_data , default is FALSE
`coding_method`	A character string that indicates what kind of additional coding schemes should be used. Default is none. Other options include "binary" and "multi".
`fast`	Logical. If TRUE, uses a faster method for data processing.

Value

Resulting 'lm' object from performing the discordant regression.

Examples

```
discord_regression(
  data = data_sample,
  outcome = "height",
  predictors = "weight",
  pair_identifiers = c("_s1", "_s2"),
  sex = NULL,
  race = NULL
)
```

`kinsim` *Simulate Biometrically Informed Multivariate Data*

Description

Generates paired multivariate data for kinship pairs based on specified ACE (Additive genetic, Common environment, unique Environment) parameters with covariance structure.

Usage

```
kinsim(
  r_all = c(1, 0.5),
  c_all = 1,
  npg_all = 500,
  npergroup_all = rep(npg_all, length(r_all)),
  mu_all = 0,
  variables = 2,
  mu_list = rep(mu_all, variables),
  r_vector = NULL,
  c_vector = NULL,
  ace_all = c(1, 1, 1),
  ace_list = matrix(rep(ace_all, variables), byrow = TRUE, nrow = variables),
  cov_a = 0,
  cov_c = 0,
  cov_e = 0,
  ...
)
```

Arguments

`r_all`	Numeric vector. Levels of genetic relatedness for each group; default is c(1, 0.5) representing MZ and DZ twins respectively.
`c_all`	Numeric. Default shared variance for common environment; default is 1.
`npg_all`	Integer. Default sample size per group; default is 500.
`npergroup_all`	Numeric vector. Sample sizes by group; default repeats `npg_all` for all groups in `r_all`.
`mu_all`	Numeric. Default mean value for all generated variables; default is 0.
`variables`	Integer. Number of variables to generate; default is 2. Currently limited to a maximum of two variables.
`mu_list`	Numeric vector. Means for each variable; default repeats `mu_all` for all variables.
`r_vector`	Numeric vector. Alternative specification providing genetic relatedness coefficients for the entire sample; default is NULL.
`c_vector`	Numeric vector. Alternative specification providing shared-environmental relatedness
`ace_all`	Numeric vector. Default variance components in order c(a, c, e) for all variables; default is c(1, 1, 1).
`ace_list`	Matrix. ACE variance components by variable, where each row represents a variable and columns are a, c, e components; default repeats `ace_all` for each variable.
`cov_a`	Numeric. Shared variance for additive genetics between variables; default is 0.
`cov_c`	Numeric. Shared variance for shared-environment between variables; default is 0.

`cov_e`	Numeric. Shared variance for non-shared-environment between variables; default is 0.
`...`	Additional arguments passed to other methods.

Details

This function extends the univariate ACE model to multivariate data, allowing simulation of correlated phenotypes across kinship pairs with different levels of genetic relatedness. It supports simulation of up to two phenotypic variables with specified genetic and environmental covariance structures.

Value

A data frame with the following columns:

Ai_1 genetic component for variable i for kin1

Ai_2 genetic component for variable i for kin2

Ci_1 shared-environmental component for variable i for kin1

Ci_2 shared-environmental component for variable i for kin2

Ei_1 non-shared-environmental component for variable i for kin1

Ei_2 non-shared-environmental component for variable i for kin2

yi_1 generated variable i for kin1

yi_2 generated variable i for kin2

r level of relatedness for the kin pair

id Unique identifier for each kinship pair

Examples

```
# Generate basic multivariate twin data with default parameters
twin_data <- kinsim()

# Generate data with genetic correlation between variables
correlated_data <- kinsim(cov_a = 0.5)

# Generate data for different relatedness groups with custom parameters
family_data <- kinsim(
  r_all = c(1, 0.5, 0.25), # MZ twins, DZ twins, and half-siblings
  npergroup_all = c(100, 100, 150), # Sample sizes per group
  ace_list = matrix(
    c(
      1.5, 0.5, 1.0, # Variable 1 ACE components
      0.8, 1.2, 1.0
    ), # Variable 2 ACE components
    nrow = 2, byrow = TRUE
  ),
  cov_a = 0.3, # Genetic covariance
  cov_c = 0.2 # Shared environment covariance
)
```

`make_mean_diffs` *Make Mean Differences*

Description

This function calculates differences and means of a given variable for each kinship pair. The order of subtraction and the variables' names in the output dataframe depend on the order column set by check_sibling_order(). If the demographics parameter is set to "race", "sex", or "both", it also prepares demographic information accordingly, swapping the order of demographics as per the order column.

Usage

```
make_mean_diffs(..., fast = FALSE)
```

Arguments

`...`	Additional arguments to be passed to the function.
`fast`	Logical. If TRUE, uses a faster method for data processing.

Index