Package 'rdss'

October 26, 2026

Title Companion Datasets and Functions for Research Design in the Social Sciences

Version 1.0.16

Description Helper functions to accompany the Blair, Coppock, and Humphreys (2023) ``Research Design in the Social Sciences: Declaration, Diagnosis, and Redesign'' <https://book.declaredesign.org>. 'rdss' includes datasets, helper functions, and plotting components to enable use and replication of the book.

Imports dplyr, rlang (>= 1.0.0), generics, ggplot2, tibble, tidyr, dataverse, readr, broom, purrr, estimatr, randomizr

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat (>= 3.0.0), rdrobust, DIDmultiplegt, broom.mixed, marginaleffects, grf, CausalQueries, metafor, cjoint, lme4, rstanarm, spdep, DeclareDesign, curl, polars

Depends R (>= 4.1.0)

Config/testthat/edition 3

URL https://declaredesign.org/r/rdss/, https://github.com/DeclareDesign/rdss

BugReports https://github.com/DeclareDesign/rdss/issues

Additional_repositories https://rpolars.r-universe.dev

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Graeme Blair [aut] (ORCID: <https://orcid.org/0000-0001-9164-2102>), Alexander Coppock [aut, cre] (ORCID: <https://orcid.org/0000-0002-5733-2386>), Macartan Humphreys [aut] (ORCID: <https://orcid.org/0000-0001-7029-2326>)

Maintainer Alexander Coppock <acoppock@gmail.com>

Repository CRAN

Date/Publication 2026-10-26 19:50:02 UTC

Contents

add_parens . . . 2
best_predictor . . . 3
bonilla_tillery . . . 4
causal_forest_handler . . . 5
clingingsmith_etal . . . 6
conjoint_assignment . . . 6
conjoint_inquiries . . . 7
conjoint_measurement . . . 8
dd_palette . . . 9
did_multiplegt_tidy . . . 10
estimator_AS_tidy . . . 11
fairfax . . . 12
foos_etal . . . 13
format_num . . . 13
get_exposure_AS . . . 14
get_rdss_file . . . 15
hex_add_alpha . . . 18
lag_by_group . . . 19
lapop_brazil . . . 20
la_voter_file . . . 21
make_interval_entry . . . 21
make_se_entry . . . 22
post_stratification_helper . . . 23
process_tracing_estimator . . . 24
rdrobust_helper . . . 25
rdss . . . 26
rma_helper . . . 26
rma_mu_tau . . . 27
theme_dd . . . 28
tidy.amce . . . 29
tidy.rdrobust . . . 30
tidy_stan . . . 31

Index **32**

`add_parens` *Add parentheses around standard error estimates*

Description

Add parentheses around standard error estimates

Usage

```
add_parens(x, digits = 3)
```

Arguments

x	A numeric vector.
digits	The number of decimal places. Defaults to 3.

Value

A character vector, e.g. "(0.120)".

Examples

```
std.error <- c(0.12, 0.001, 1.2)
add_parens(std.error)
```

---

best_predictor *Find the covariate that best predicts treatment effect heterogeneity*

Description

An inquiry handler. For each covariate, regresses the true unit-level effect tau on the covariate cut into intervals, and reports which covariate gives the highest R-squared. Pair it with causal_forest_handler(), whose var_imp column is the estimate of the same quantity.

Usage

```
best_predictor(data, covariate_names, cuts = 20)
```

Arguments

data	A data frame with a column tau holding each unit's treatment effect, and the covariates named in covariate_names.
covariate_names	A character vector of covariates to assess.
cuts	Either a numeric vector of two or more unique cut points or a single number (greater than or equal to 2) giving the number of intervals into which each covariate is to be cut. Defaults to 20.

Details

See https://book.declaredesign.org/complex-designs.html#discovery-using-causal-forests

Value

A data frame with one row: inquiry ("best_predictor") and estimand, the position in covariate_names of the best predictor.

Examples

```
set.seed(343)
dat <- data.frame(A = rnorm(500), B = rnorm(500))
dat$tau <- 1 + dat$A

# The effect varies with A, so A (position 1) is the best predictor
best_predictor(dat, covariate_names = c("A", "B"), cuts = 4)
```

`bonilla_tillery` *Replication data for Bonilla and Tillery (2020), American Political Science Review (obtained from Dataverse 10.7910/DVN/IUZDQI)*

Description

One row per survey respondent. `Z` is the randomly assigned message (a factor with four levels) and `blm_support` the outcome; the remaining columns are respondent characteristics.

Usage

```
bonilla_tillery
```

Format

A data frame with 849 rows and 10 columns: `female`, `lgbtq`, `age`, `religiosity`, `income`, `college`, `blm_familiarity`, `linked_fate`, `Z`, and `blm_support`.

Source

Bonilla, Tabitha, and Alvin B. Tillery. 2020. "Which Identity Frames Boost Support for and Mobilization in the #BlackLivesMatter Movement? An Experimental Test." *American Political Science Review* 114(4): 947-962. doi:10.1017/S0003055420000544. Data from Dataverse, doi:10.7910/DVN/IUZDQI.

Examples

```
estimatr::lm_robust(blm_support ~ Z, data = bonilla_tillery)
```

`causal_forest_handler` *Tidy helper function for causal_forest function*

Description

Splits the data at random into a training and a test set, fits `grf::causal_forest()` on the training set, and predicts each unit's treatment effect. The data must contain the outcome as `Y` and the treatment as `Z`.

Usage

```
causal_forest_handler(data, covariate_names, share_train = 0.5, ...)
```

Arguments

`data`	A data frame with columns `Y`, `Z`, and the covariates named in `covariate_names`.
`covariate_names`	A character vector naming the covariate columns the forest may split on.
`share_train`	The share of units assigned to the training set. Defaults to 0.5.
`...`	Further arguments passed to `grf::causal_forest()`.

Details

See https://book.declaredesign.org/complex-designs.html#discovery-using-causal-forests

Value

`data` with four columns added: `pred` (the predicted treatment effect), `var_imp` (the position in `covariate_names` of the most important covariate), and the logical indicators `train` and `test`.

Examples

```
library(DeclareDesign)
library(ggplot2)

dat <- fabricate(
   N = 1000,
   A = rnorm(N),
   B = rnorm(N),
   Z = complete_rs(N),
   Y = A*Z + rnorm(N))

# note: remove num.threads = 1 to use more processors
estimates <- causal_forest_handler(data = dat, covariate_names = c("A", "B"), num.threads = 1)

ggplot(data = estimates, aes(A, pred)) + geom_point()
```

clingingsmith_etal — *Replication data for Clingingsmith, Khwaja, and Kremer (2009), Quarterly Journal of Economics*

Description

From "Estimating the Impact of the Hajj: Religion and Tolerance in Islam's Global Gathering," The Quarterly Journal of Economics 124(3): 1133-1170. One row per applicant to the Hajj visa lottery. `success` is 1 for lottery winners; the `views_` columns are attitudes toward each group, and `views` is their sum.

Usage

```
clingingsmith_etal
```

Format

A data frame with 958 rows and 8 columns: `success`, `views_saudi`, `views_indonesian`, `views_turkish`, `views_african`, `views_chinese`, `views_european`, and `views`.

Source

Clingingsmith, David, Asim Ijaz Khwaja, and Michael Kremer. 2009. "Estimating the Impact of the Hajj: Religion and Tolerance in Islam's Global Gathering." *The Quarterly Journal of Economics* 124(3): 1133-1170. doi:10.1162/qjec.2009.124.3.1133.

Examples

```
estimatr::difference_in_means(views ~ success, data = clingingsmith_etal)
```

conjoint_assignment — *Conjoint experiment assignment handler: conducts complete random assignment of all attribute levels*

Description

For each attribute, assigns every row one of that attribute's levels by complete random assignment, so the levels appear in as equal numbers as possible.

Usage

```
conjoint_assignment(data, levels_list)
```

Arguments

`data`	A data frame with one row per profile.
`levels_list`	A named list with one element per attribute, each a vector of that attribute's levels, e.g. `list(gender = c("woman", "man"), age = c("young", "old"))`.

Details

See https://book.declaredesign.org/experimental-descriptive.html#conjoint-experiments

Value

`data` with one column per attribute, named as in `levels_list`.

Examples

```
set.seed(343)
levels_list <- list(gender = c("woman", "man"), age = c("young", "old"))

# 25 subjects, each choosing between two profiles in each of four tasks
profiles <- expand.grid(profile = 1:2, task = 1:4, subject = 1:25)

head(conjoint_assignment(profiles, levels_list))
```

`conjoint_inquiries` *Conjoint experiment inquiries handler*

Description

Computes the true average marginal component effect (AMCE) of every attribute level relative to that attribute's first level. Each AMCE is the average change in the probability that the second profile in a task is chosen when the attribute moves from its first level to the given level, with the other attributes drawn at random by `conjoint_assignment()`.

Usage

```
conjoint_inquiries(data, levels_list, utility_fn)
```

Arguments

`data`	A data frame with one row per profile and columns `subject`, `task`, and `profile` (1 or 2).
`levels_list`	A named list of attribute levels, as in `conjoint_assignment()`. The first level of each attribute is the reference.
`utility_fn`	A function that adds the utility column `U`, as in `conjoint_measurement()`.

Details

See https://book.declaredesign.org/experimental-descriptive.html#conjoint-experiments

Value

A data frame with one row per non-reference level: `attribute`, `reference`, `level`, `inquiry`, and `estimand`.

Examples

```
set.seed(343)
levels_list <- list(gender = c("woman", "man"), age = c("young", "old"))

# 25 subjects, each choosing between two profiles in each of four tasks
profiles <- expand.grid(profile = 1:2, task = 1:4, subject = 1:25)

utility_fn <- function(data) {
  data$U <- 0.5 * (data$gender == "woman") - 0.2 * (data$age == "old") +
    rnorm(nrow(data))
  data
}

conjoint_inquiries(profiles, levels_list, utility_fn)
```

---

conjoint_measurement *Conjoint experiment measurement handler: records which profile in each task is chosen*

Description

Applies `utility_fn` and, within each subject and task, compares the two profiles and sets `choice` to 1 for the one with the higher utility `U` (the second profile on a tie).

Usage

```
conjoint_measurement(data, utility_fn)
```

Arguments

`data`	A data frame with one row per profile and columns `subject`, `task`, and `profile`. Each task must hold exactly two profiles.
`utility_fn`	A function that takes `data` and returns it with a column `U` added, the utility the subject gets from each profile.

Details

See https://book.declaredesign.org/experimental-descriptive.html#conjoint-experiments

Value

data with U and choice added.

Examples

```
set.seed(343)
levels_list <- list(gender = c("woman", "man"), age = c("young", "old"))

# 25 subjects, each choosing between two profiles in each of four tasks
profiles <- expand.grid(profile = 1:2, task = 1:4, subject = 1:25)
profiles <- conjoint_assignment(profiles, levels_list)

utility_fn <- function(data) {
  data$U <- 0.5 * (data$gender == "woman") - 0.2 * (data$age == "old") +
    rnorm(nrow(data))
  data
}

head(conjoint_measurement(profiles, utility_fn))
```

---

dd_palette *Color palettes used in the book Research Design in the Social Sciences*

Description

Based on Karthik Ram's wesanderson package (https://github.com/karthik/wesanderson)

Usage

```
dd_palette(name, n)
```

Arguments

name	The palette name, as a string. One of the names listed below.
n	The number of colors to return, from the start of the palette. Defaults to all of them.

Details

Palettes:

- three_color_palette: light blue, orange, pink
- grey_palette: light blue, orange, pink, light gray
- quilt_palette: light gray, pink, purple, light blue, orange
- two_color_palette: dark blue, pink
- quilt_three_color_palette: light gray, translucent light blue, light blue

- two_color_gray: dark blue, light gray

Single colors: dd_dark_blue ("#3564ED"), dd_light_blue ("#72B4F3"), dd_orange ("#F38672"), dd_purple ("#7E43B6"), dd_gray (gray(0.2)), dd_pink ("#C6227F"), dd_light_gray (gray(0.8)), and the translucent dd_dark_blue_alpha and dd_light_blue_alpha.

Value

A character vector of hex color codes.

Examples

```
dd_palette("three_color_palette")
dd_palette("quilt_palette", n = 3)
```

---

did_multiplegt_tidy *Tidy helper function for did_multiplegt*

Description

Runs DIDmultiplegt::did_multiplegt(), the de Chaisemartin and D'Haultfoeuille estimator, and returns its estimates as a data frame.

Usage

```
did_multiplegt_tidy(data, ...)
```

Arguments

data	A data frame, passed to did_multiplegt() as df.
...	Further arguments to did_multiplegt(). These must include mode and, for mode = "dyn", the column names of the outcome (outcome), group (group), time period (time), and treatment (treatment), as strings, and the number of effects to estimate (effects). graph_off defaults to TRUE, so that no plot is drawn.

Details

Use mode = "dyn". It estimates the effect l periods after a group first switches treatment, for l from 1 to effects; effects = 1 is the effect in the period of the switch. This mode runs did_multiplegt_dyn() from the 'DIDmultiplegtDYN' package, which also needs the 'polars' package. polars is not on CRAN; install it with install.packages("polars", repos = "https://rpolars.r-universe.d and attach it with library(polars).

mode = "old" is still passed through, but in 'DIDmultiplegt' 2.1.0 it returns NaN: it takes first differences with stats::lag(), which does not shift the data, so no group is ever seen to switch. The function warns when that happens.

See https://book.declaredesign.org/observational-causal.html#difference-in-differences

Value

For `mode = "dyn"`, a data frame with one row per effect and columns `term` (`"Effect_1"`, `"Effect_2"`, ...), `estimate`, `std.error`, `conf.low`, `conf.high`, `n` (the number of group-period cells used), and `n_switchers` (the number of switching groups). For `mode = "old"`, a data frame with one column, `estimate`.

Examples

```
library(polars)

set.seed(343)
# 20 units in 10 periods; units switch into treatment at different times
panel <- data.frame(unit = rep(1:20, times = 10), period = rep(1:10, each = 20))
panel$D <- as.numeric(panel$period > (panel$unit %% 5) + 4)
panel$Y <- 0.5 * panel$D + rnorm(nrow(panel))

did_multiplegt_tidy(
  panel, mode = "dyn",
  outcome = "Y", group = "unit", time = "period", treatment = "D",
  effects = 2
)
```

---

estimator_AS_tidy *Tidy helper function for estimator_AS function*

Description

Runs the Aronow and Samii estimator from the 'interference' package and returns Horvitz-Thompson and Hajek estimates of the total, direct, and indirect effects.

Usage

```
estimator_AS_tidy(
  data,
  permutatation_matrix = NULL,
  adj_matrix,
  obs_prob_exposure = NULL
)
```

Arguments

`data`	A data frame with columns `Z` and `Y`.
`permutatation_matrix`	(The spelling is the published one and is kept so existing code runs.) A matrix of random treatment assignments with one row per permutation of the treatment vector and one column per unit.

`adj_matrix` An adjacency matrix defining the network structure.

`obs_prob_exposure`
Optional precomputed exposure probabilities.

Details

The 'interference' package is not available on CRAN, so in the CRAN version of rdss this function only explains how to install the GitHub versions of both packages, where it works: remotes::install_github('szonszein/interfer remotes::install_github('DeclareDesign/rdss')

It returns nothing, so a design that includes it still declares and diagnoses.

See https://book.declaredesign.org/experimental-causal.html#experiments-over-networks

Value

Nothing, invisibly, in the CRAN version. The GitHub version returns a data frame with columns `term`, `inquiry`, `estimator`, and `estimate`.

`fairfax` *Shapefile of Fairfax County, Virginia, voting precincts*

Description

Boundaries of the 238 voting precincts in Fairfax County, Virginia, with each precinct's ID, name, district, and polling place. One row per precinct. The book uses it to build the adjacency matrix for the experiment over networks; see `estimator_AS_tidy()`.

Usage

```
fairfax
```

Format

An sf object with 238 rows and 10 columns:

Source

Fairfax County Government, voting precincts layer (`REGMGR.VOTING_PRECINCTS`).

Examples

```
names(fairfax)
```

foos_etal — *Replication data for Foos, John, Muller, and Cunningham (2021), Journal of Politics (derived from Dataverse 10.7910/DVN/NDPXND)*

Description

One row per voter. `treat` is the randomly assigned treatment, `marked_register_2014` the turnout outcome, and `ward` and `street` (a hashed identifier) the levels at which voters are clustered.

Usage

```
foos_etal
```

Format

A data frame with 8,375 rows and 5 columns: `marked_register_2014`, `ward`, `street`, `treat`, and `weights`.

Source

Foos, Florian, Peter John, Christian Müller, and Kevin Cunningham. 2021. "Social Mobilization in Partisan Spaces." *The Journal of Politics* 83(3): 1190-1197. doi:10.1086/710970. Data from Dataverse, doi:10.7910/DVN/NDPXND.

Examples

```
estimatr::lm_robust(marked_register_2014 ~ treat, clusters = street, data = foos_etal)
```

format_num — *Round and pad a number to a specific decimal place*

Description

Rounds to `digits` decimal places and keeps trailing zeros, so a column of numbers lines up in a table.

Usage

```
format_num(x, digits = 3)
```

Arguments

`x`	A numeric vector.
`digits`	The number of decimal places. Defaults to 3.

Value

A character vector.

Examples

```
std.error <- c(0.12, 0.001, 1.2)
format_num(std.error)
```

`get_exposure_AS` *Helper function to obtain the observed exposure for the Aronow and Samii estimator*

Description

Converts the exposure map from `interference::make_exposure_map_AS()`, a 0/1 matrix with one row per unit and one column per exposure condition, into a vector naming each unit's condition (e.g. `"dir_ind1"` or `"no"`).

Usage

```
get_exposure_AS(obs_exposure)
```

Arguments

`obs_exposure`	An exposure map, as returned by `interference::make_exposure_map_AS()`.

Details

See https://book.declaredesign.org/experimental-causal.html#experiments-over-networks

Value

A character vector with one element per unit.

Examples

```
# An exposure map for three units, one column per exposure condition
exposure <- cbind(dir_ind1 = c(1, 0, 0), isol_dir = c(0, 1, 0), no = c(0, 0, 1))
get_exposure_AS(exposure)
```

`get_rdss_file` *Download a replication file from the dataverse archive for Research Design in the Social Sciences: Declaration, Diagnosis, and Redesign*

Description

Downloads one of the book's design declarations or diagnosis objects from its Dataverse archive and returns it as an R object. Requires an internet connection.

Usage

```
get_rdss_file(name, verbose = TRUE)
```

Arguments

`name`	The name of the file, as a string, e.g. `"declaration_2.1"`. The names are listed above.
`verbose`	If `TRUE`, print the code that creates a declaration when one is requested.

Details

See https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/HYVPO5 for further details and the code used to create these files.

The available names include:

Design declaration objects:

declaration_9.5
declaration_2.1
declaration_2.2
declaration_4.1
declaration_5.1
declaration_7.1
declaration_9.1
declaration_9.2
declaration_9.3
declaration_9.4
declaration_9.6
declaration_9.7
declaration_10.1
declaration_10.2
declaration_10.3
declaration_10.4
declaration_10a
declaration_11.1
declaration_11.2
declaration_11.3
declaration_11.4

declaration_11.5
declaration_12.1a
declaration_12.1b
declaration_12.1c
declaration_12.1d
declaration_13.1
declaration_13.2
declaration_15.1
declaration_15.2
declaration_15.3a
declaration_15.3b
declaration_15.3c
declaration_15.4
declaration_15.5
declaration_15.6
declaration_16.1a
declaration_16.1b
declaration_16.2
declaration_16.3
declaration_16.4
declaration_16.5
declaration_16.6
declaration_17.1
declaration_17.2
declaration_17.3
declaration_17.4
declaration_17.5
declaration_17.6_a
declaration_17.6_b
declaration_18.1
declaration_18.2
declaration_18.3
declaration_18.4
declaration_18.5
declaration_18.6
declaration_18.7
declaration_18.8
declaration_18.9a
declaration_18.9b
declaration_18.9c
declaration_18.10
declaration_18.11
declaration_18.12
declaration_18.13
declaration_19.1
declaration_19.2
declaration_19.3
declaration_19.4

declaration_23.1a
declaration_23.1b
declaration_23.1c
declaration_23.1d

Diagnosis objects:

diagnosis_2.1
diagnosis_4.1
diagnosis_9.1
diagnosis_9.2
diagnosis_9.3
diagnosis_9.4
diagnosis_9.5
diagnosis_9.6
diagnosis_9.7
simulation_10.1
diagnosis_10.1
diagnosis_10.2
diagnosis_10.3
diagnosis_10.4
diagnosis_10.5
diagnosis_10a
diagnosis_11.1
diagnosis_11.2
diagnosis_11.3
diagnosis_11.4
diagnosis_11.5
diagnosis_12.1
diagnosis_12.2
diagnosis_13.1
diagnosis_15.1
diagnosis_15.2
diagnosis_15.3
diagnosis_15.4
diagnosis_15.5
diagnosis_16.1
diagnosis_16.2
diagnosis_16.3
diagnosis_16.4
diagnosis_16.5
diagnosis_17.1
diagnosis_17.2
diagnosis_17.3
diagnosis_17.4
diagnosis_17.5
diagnosis_18.1
diagnosis_18.10_encouragment
diagnosis_18.10_placebo

diagnosis_18.11
diagnosis_18.12
diagnosis_18.13
diagnosis_18.2
diagnosis_18.3
diagnosis_18.4
diagnosis_18.5
diagnosis_18.6
diagnosis_18.7
diagnosis_18.8
diagnosis_18.9
diagnosis_19.1
diagnosis_19.2
diagnosis_19.3
diagnosis_19.4
diagnosis_19a
diagnosis_21a
diagnosis_21b
diagnosis_23.1
diagnosis_23a

Value

The requested object: a design declaration, or a diagnosis or simulation data frame.

Examples

```
## Not run:
# Requires internet access
if(curl::has_internet()) {
  diagnosis_2.1 <- get_rdss_file("diagnosis_2.1")
  diagnosis_2.1
}

## End(Not run)
```

---

hex_add_alpha *Add alpha transparency to a color defined in hexadecimal*

Description

Add alpha transparency to a color defined in hexadecimal

Usage

```
hex_add_alpha(col, alpha)
```

Arguments

col	A color as a six-digit hex code, e.g. "#72B4F3".
alpha	Opacity, from 0 (transparent) to 1 (opaque).

Value

The color as an eight-digit hex code.

Examples

```
hex_add_alpha("#72B4F3", 0.5)
```

---

lag_by_group *Generate lags in grouped data*

Description

Lags x within each group, ordering by order_by rather than by row order. Write the variables as bare column names inside fabricatr::fabricate() or dplyr::mutate(); see the README for an example.

Usage

```
lag_by_group(x, groups, n = 1, order_by, default = NA)
```

Arguments

x	The variable to lag.
groups	The grouping variable, e.g. a unit identifier.
n	A positive integer: the number of periods to lag by. Defaults to 1.
order_by	Ordering variable within group (e.g., time)
default	The value for rows with no earlier period in their group. Defaults to NA.

Details

See https://book.declaredesign.org/observational-causal.html#difference-in-differences

Value

A vector the length of x, in the original row order.

Examples

```
# Three units observed in four periods; unit u is treated after period u
panel <- expand.grid(unit = 1:3, period = 1:4)
panel$D <- as.numeric(panel$period > panel$unit)

dplyr::mutate(panel, D_lag = lag_by_group(D, groups = unit, order_by = period))
```

---

lapop_brazil *Teaching data based on the 2018/19 LAPOP AmericasBarometer survey of Brazil*

Description

Used in the book's student exercises. These data were resampled with replacement from the LAPOP data, to 10,000 rows, for a subset of variables. These data cannot be used for scientific inferences, and are only useful for teaching purposes. ID numbers were scrambled so that individuals and municipalities cannot easily be identified.

Usage

```
lapop_brazil
```

Format

A data frame with 10,000 rows and 10 columns: `ID`, `municipality`, and eight survey items (`trust_police`, `govt_responsive`, `ideology`, `govt_pride`, `self_efficacy_political`, `trust_military`, `trust_supreme_court`, and `support_political_system`).

Details

Download the original data from https://www.vanderbilt.edu/cgd/data-access/

See https://www.vanderbilt.edu/lapop/core-surveys.php for the survey questionnaire.

Source

LAPOP AmericasBarometer 2018/19 survey of Brazil (fieldwork January to March 2019), Vanderbilt University, https://www.vanderbilt.edu/cgd/data-access/.

Examples

```
summary(lapop_brazil$trust_police)
```

la_voter_file *Voter file sample for Los Angeles County*

Description

Party registration, age, census tract, and 2012 turnout for 1,000 registered voters sampled at random from the Los Angeles County voter file. One row per voter.

Usage

```
la_voter_file
```

Format

A data frame with 1000 rows and 4 variables:

party political party registration

age age of voter in years

census_tract US Census tract number

voted_2012 1 if the voter voted in the 2012 general election, 0 otherwise

Source

California Secretary of State.

Examples

```
table(la_voter_file$party)
```

make_interval_entry *Format confidence intervals for nice printing*

Description

Format confidence intervals for nice printing

Usage

```
make_interval_entry(conf.low, conf.high, digits = 2)
```

Arguments

conf.low	A numeric vector of lower bounds.
conf.high	A numeric vector of upper bounds.
digits	The number of decimal places. Defaults to 2.

Value

A character vector, e.g. `"[-0.17, 0.31]"`.

Examples

```
conf.low <- c(-0.1652, 0.00304, -6.352)
conf.high <- c(0.3052, 0.00696, -1.648)

make_interval_entry(conf.low, conf.high)
```

`make_se_entry` *Format estimates and standard errors for nice printing*

Description

Format estimates and standard errors for nice printing

Usage

```
make_se_entry(estimate, std.error, digits = 2)
```

Arguments

`estimate`	A numeric vector of estimates.
`std.error`	A numeric vector of standard errors, the same length as `estimate`.
`digits`	The number of decimal places. Defaults to 2.

Value

A character vector, e.g. `"0.07 (0.12)"`.

Examples

```
estimate <- c(0.07, 0.005, -4)
std.error <- c(0.12, 0.001, 1.2)

make_se_entry(estimate, std.error)
```

post_stratification_helper *Post stratification estimator helper*

Description

Predicts each row's outcome from a fitted model, then averages the predictions within each group using the post-stratification weights. In the book, the rows are demographic cells and the groups are states.

Usage

```
post_stratification_helper(model_fit, data, group, weights)
```

Arguments

model_fit	A fitted model that marginaleffects::predictions() accepts, e.g. from lme4::glmer() or estimatr::lm_robust(). For a mixed model, random effects are left out of the predictions (re.form = NA); for any other model no such argument is passed.
data	A data frame of post-stratification cells to predict for.
group	The bare column name of the grouping variable.
weights	The bare column name of the post-stratification weights, e.g. each cell's population share.

Details

See https://book.declaredesign.org/observational-descriptive.html#multi-level-regression-and-poststratification

Value

A data frame with one row per group and columns for the group and estimate.

Examples

```
set.seed(343)
# Post-stratification cells: high school graduates and non-graduates in ten
# states, weighted by each state's share of graduates
cells <- expand.grid(state = state.name[1:10], HS = 0:1)
prob_HS <- rep(seq(0.5, 0.9, length.out = 10), 2)
cells$PS_weight <- ifelse(cells$HS == 1, prob_HS, 1 - prob_HS)

survey <- cells[sample(nrow(cells), 1000, replace = TRUE), ]
survey$Y <- rbinom(1000, 1, prob = plogis(0.5 * survey$HS))

model_fit <- lme4::glmer(Y ~ HS + (1 | state), data = survey, family = binomial)
post_stratification_helper(model_fit, data = cells, group = state, weights = PS_weight)
```

process_tracing_estimator *Process tracing estimator*

Description

Answers a causal query about one case using a causal model from 'CausalQueries', once for each process tracing strategy. A strategy names the nodes whose values are observed, so the estimate shows how the answer changes as more of the case's data is examined.

Usage

```
process_tracing_estimator(causal_model, query, data, strategies)
```

Arguments

causal_model	A model from `CausalQueries::make_model()`.
query	The causal query, in 'CausalQueries' syntax, e.g. `"Y[X=1] > Y[X=0]"`.
data	A one-row data frame giving the case's value on each observed node. Nodes that are `NA` are treated as unobserved.
strategies	A character vector of strategies, each listing the nodes observed, separated by hyphens, e.g. `c("X-Y", "X-M-Y")`.

Details

See https://book.declaredesign.org/observational-causal.html#process-tracing

Value

A data frame with one row per strategy: `term` (the strategy), `XY` (the case's values of `X` and `Y`, e.g. `"X1Y1"`), and `estimate` (the posterior mean of the query).

Examples

```
# Simple example showing ambiguity in attribution
process_tracing_estimator(
  causal_model = CausalQueries::make_model("X -> Y"),
  query = "Y[X=1] > Y[X=0]",
  data = data.frame(X=1, Y = 1),
  strategies = "X-Y")

# Example where M=1 acts as a hoop test
process_tracing_estimator(
  causal_model = CausalQueries::make_model("X -> M -> Y") |>
   CausalQueries::set_restrictions("Y[M=1] < Y[M=0]") |>
   CausalQueries::set_restrictions("M[X=1] < M[X=0]"),
  query = "Y[X=1] > Y[X=0]",
  data = data.frame(X=1, Y = 1, M = 0),
```

```
  strategies = c("Y", "X-Y", "X-M-Y"))
```

`rdrobust_helper` *Helper function for using rdrobust as a model in* `declare_estimator`

Description

Lets `rdrobust::rdrobust()`, which takes vectors, be called with a data frame and column names. Pass it as `.method` in `declare_estimator()`; `tidy.rdrobust()` then tidies the fit.

Usage

```
rdrobust_helper(data, y, x, subset = NULL, ...)
```

Arguments

`data`	A data frame.
`y`	The bare column name of the outcome.
`x`	The bare column name of the running variable.
`subset`	An optional logical expression in terms of the columns of `data`, selecting the rows to use.
`...`	Further arguments to `rdrobust::rdrobust()`, e.g. `c` for the cutoff.

Details

See https://book.declaredesign.org/observational-causal.html#regression-discontinuity-designs

Value

rdrobust model fit object

Examples

```
set.seed(343)
rd <- data.frame(X = runif(1000, min = -1, max = 1))
rd$Y <- 0.5 * (rd$X > 0) + rd$X + rnorm(1000, sd = 0.5)

fit <- rdrobust_helper(rd, y = Y, x = X, c = 0)
tidy(fit)
```

rdss *rdss package*

Description

Companion datasets and functions for the book *Research Design in the Social Sciences: Declaration, Diagnosis, and Redesign* by Graeme Blair, Alexander Coppock, and Macartan Humphreys (Princeton University Press, 2023), free to read at https://book.declaredesign.org.

Details

Most functions fill one step of a design declaration in the book, and each help page links to the section that uses it. The Get started article on the package website, https://declaredesign.org/r/rdss/articles/rdss.html, lists them chapter by chapter.

Author(s)

Maintainer: Alexander Coppock <acoppock@gmail.com> (ORCID)

Authors:

- Alexander Coppock <acoppock@gmail.com> (ORCID)
- Graeme Blair <graeme.blair@gmail.com> (ORCID)
- Macartan Humphreys <macartan@gmail.com> (ORCID)

See Also

Useful links:

- https://declaredesign.org/r/rdss/
- https://github.com/DeclareDesign/rdss
- Report bugs at https://github.com/DeclareDesign/rdss/issues

rma_helper *Helper function for rma function in metafor package*

Description

See https://book.declaredesign.org/complex-designs.html#meta-analysis

Usage

```
rma_helper(data, yi, sei, method = "REML", ...)
```

Arguments

data	A data frame with one row per study.
yi	The bare column name of the study estimates.
sei	The bare column name of their standard errors.
method	character string to specify whether a fixed- or a random/mixed-effects model should be fitted. A fixed-effects model (with or without moderators) is fitted when using method = "FE". Random/mixed-effects models are fitted by setting method equal to one of the following: "DL", "HE", "SJ", "ML", "REML", "EB", "HS", "HSk", or "GENQ". Default is "REML".
...	Further arguments to metafor::rma().

Details

Fits metafor::rma() to study-level estimates and standard errors. If the fit fails, it returns the error as an object rma_mu_tau() recognizes, so one failed simulation does not stop a diagnosis.

Value

An rma.uni fit; pass it to rma_mu_tau() to tidy it.

Examples

```
set.seed(343)
studies <- data.frame(
  est = rnorm(10, mean = 0.2, sd = 0.1),
  se = runif(10, min = 0.05, max = 0.15)
)
fit <- rma_helper(studies, yi = est, sei = se)
fit
```

---

rma_mu_tau *Extract mu and tau-squared from a random-effects meta-analysis*

Description

A tidier for fits from rma_helper(): mu is the average effect across studies and tau_sq the variance of the true effects across studies.

Usage

```
rma_mu_tau(fit)
```

Arguments

fit	A fit from rma_helper() or metafor::rma().

Details

See https://book.declaredesign.org/complex-designs.html#meta-analysis

Value

A data frame with two rows, `mu` and `tau_sq`, with estimates, standard errors, and confidence intervals (for `tau_sq`, intervals only when `method = "REML"`). If the fit failed, one row with `estimate = NA` and `error = TRUE`.

Examples

```
set.seed(343)
studies <- data.frame(
  est = rnorm(10, mean = 0.2, sd = 0.1),
  se = runif(10, min = 0.05, max = 0.15)
)
fit <- rma_helper(studies, yi = est, sei = se)
rma_mu_tau(fit)
```

---

theme_dd *ggplot theme used in the book Research Design in the Social Sciences*

Description

`ggplot2::theme_minimal()` with light gray major gridlines only, no axis ticks, and no legend. The book labels series directly, so it hides the legend; add `theme(legend.position = "right")` after it to get one back.

Usage

```
theme_dd()
```

Value

A ggplot2 theme.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) + geom_point() + theme_dd()
```

`tidy.amce` *Tidy estimates from the amce estimator*

Description

A `generics::tidy()` method for fits from `cjoint::amce()`: one row per attribute level, with normal-approximation confidence intervals.

Usage

```
## S3 method for class 'amce'
tidy(x, alpha = 0.05, ...)
```

Arguments

`x`	A fit from `cjoint::amce()`.
`alpha`	The significance level for the confidence intervals. Defaults to 0.05, which gives 95 percent intervals.
`...`	Not used.

Details

See https://book.declaredesign.org/experimental-descriptive.html#conjoint-experiments

Value

A data frame with columns `attribute`, `level`, `estimate`, `std.error`, `statistic`, `p.value`, `conf.low`, and `conf.high`.

Examples

```
library(cjoint)

data(immigrationconjoint)
data(immigrationdesign)

# Run AMCE estimator using all attributes in the design
results <- amce(Chosen_Immigrant ~  Gender + Education + `Language Skills` +
                  `Country of Origin` + Job + `Job Experience` + `Job Plans` +
                  `Reason for Application` + `Prior Entry`, data = immigrationconjoint,
                cluster = TRUE, respondent.id = "CaseID", design = immigrationdesign)

tidy(results)
```

`tidy.rdrobust` *Tidy helper function for rdrobust function*

Description

A `generics::tidy()` method for fits from `rdrobust::rdrobust()`, so they can be used in `declare_estimator()`. Use `rdrobust_helper()` to fit the model.

Usage

```
## S3 method for class 'rdrobust'
tidy(x, ...)
```

Arguments

`x`	A fit from `rdrobust::rdrobust()`.
`...`	Not used.

Details

See https://book.declaredesign.org/observational-causal.html#regression-discontinuity-designs

Value

A data frame with one row per estimate rdrobust reports (conventional, bias-corrected, and robust) and columns `term`, `estimate`, `std.error`, `statistic`, `p.value`, `conf.low`, `conf.high`, and `cutoff`.

Examples

```
set.seed(343)
rd <- data.frame(X = runif(1000, min = -1, max = 1))
rd$Y <- 0.5 * (rd$X > 0) + rd$X + rnorm(1000, sd = 0.5)

fit <- rdrobust_helper(rd, y = Y, x = X, c = 0)
tidy(fit)
```

tidy_stan *Tidy results from a stanreg regression (deprecated)*

Description

Deprecated: use `broom.mixed::tidy()`, which this function now calls with the same arguments.

Usage

```
tidy_stan(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

`x`	A fit from `rstanarm::stan_glm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
`conf.level`	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the coefficient estimates. Defaults to FALSE.
`...`	Further arguments to `broom.mixed::tidy()`.

Details

See https://book.declaredesign.org/choosing-an-answer-strategy.html#bayesian-formalizations

Value

A data frame with one row per coefficient.

Examples

```
fit <- rstanarm::stan_glm(mpg ~ wt, data = mtcars, chains = 1, iter = 1000,
                          refresh = 0, seed = 343)

# Prefer broom.mixed::tidy(fit, conf.int = TRUE)
tidy_stan(fit, conf.int = TRUE)
```

Index

∗ **datasets**
  bonilla_tillery, 4
  clingingsmith_etal, 6
  fairfax, 12
  foos_etal, 13
  la_voter_file, 21
  lapop_brazil, 20

add_parens, 2

best_predictor, 3
bonilla_tillery, 4

causal_forest_handler, 5
causal_forest_handler(), *3*
cjoint::amce(), *29*
clingingsmith_etal, 6
conjoint_assignment, 6
conjoint_assignment(), *7*
conjoint_inquiries, 7
conjoint_measurement, 8
conjoint_measurement(), *7*

dd_palette, 9
did_multiplegt_tidy, 10
DIDmultiplegt::did_multiplegt(), *10*
dplyr::mutate(), *19*

estimator_AS_tidy, 11
estimator_AS_tidy(), *12*

fabricatr::fabricate(), *19*
fairfax, 12
foos_etal, 13
format_num, 13

generics::tidy(), *29, 30*
get_exposure_AS, 14
get_rdss_file, 15
ggplot2::theme_minimal(), *28*
grf::causal_forest(), *5*

hex_add_alpha, 18

la_voter_file, 21
lag_by_group, 19
lapop_brazil, 20

make_interval_entry, 21
make_se_entry, 22
marginaleffects::predictions(), *23*
metafor::rma(), *27*

post_stratification_helper, 23
process_tracing_estimator, 24

rdrobust::rdrobust(), *25*, *30*
rdrobust_helper, 25
rdrobust_helper(), *30*
rdss, 26
rdss-package *(*rdss*)*, 26
rma_helper, 26
rma_helper(), *27*
rma_mu_tau, 27
rma_mu_tau(), *27*

theme_dd, 28
tidy.amce, 29
tidy.rdrobust, 30
tidy.rdrobust(), *25*
tidy_stan, 31