Package 'vazul'

May 8, 2026

Title Analysis Blinding Tools

Version 1.1.0

Description Provides tools for analysis blinding in confirmatory research contexts by masking and scrambling test-relevant aspects of data. Vector-, data frame-, and row-wise operations support blinding for hierarchical and repeated-measures designs. For more details see MacCoun and Perlmutter (2015) <doi:10.1038/526187a> and Dutilh, Sarafoglou, and Wagenmakers (2019) <doi:10.1007/s11229-019-02456-7>.

License MIT + file LICENSE

URL https://nthun.github.io/vazul/

BugReports https://github.com/nthun/vazul/issues

Depends R (>= 4.1.0)

Imports dplyr, lifecycle, rlang, tidyselect, stats

Suggests testthat, covr, knitr, quarto

VignetteBuilder quarto

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Tamás Nagy [aut, cre] (ORCID: <https://orcid.org/0000-0001-5244-0356>), Alexandra Sarafoglou [aut, dtc] (ORCID: <https://orcid.org/0000-0003-0031-685X>), Márton Kovács [aut] (ORCID: <https://orcid.org/0000-0002-8142-8492>)

Maintainer Tamás Nagy <nagytamas.hungary@gmail.com>

Repository CRAN

Date/Publication 2026-02-07 07:50:02 UTC

Contents

`marp` *MARP: Many analysts religion project dataset*

Description

A cross-cultural dataset from the Many-Analysts Religion Project (MARP), which investigated the relationship between religiosity and well-being across 24 countries and diverse religious traditions.

Usage

```
data(marp)
```

Format

A data frame with 10,535 rows (participants) and 48 variables:

subject Unique subject identifier (integer).

country Country of residence (character string).

rel_1 Importance of religion in daily life (0–10 scale).

rel_2 Frequency of religious service attendance (ordinal).

rel_3 Self-rated religiosity (0–10 scale).

rel_4 Belief in God (binary: yes/no).

rel_5 Prayer frequency (ordinal).

rel_6 Bible/study frequency (ordinal).

rel_7 Religious upbringing (binary: yes/no).

rel_8 Current religious denomination (categorical).

rel_9 Change in religiosity over lifetime (ordinal).

cnorm_1 Perceived cultural norm: importance of religious lifestyle for average person in country (0–10).

cnorm_2 Perceived cultural norm: importance of belief in God for average person in country (0–10).

wb_gen_1 Overall life satisfaction (1–5 Likert).

wb_gen_2 Overall happiness (1–5 Likert).

wb_phys_1 Energy level (1–5).

wb_phys_2 Sleep quality (1–5).

wb_phys_3 Appetite (1–5).

wb_phys_4 Physical pain/discomfort (1–5).

wb_phys_5 General health (1–5).

wb_phys_6 Exercise frequency (1–5).

wb_phys_7 Illness burden (1–5).

wb_psych_1 Positive affect (1–5).

wb_psych_2 Negative affect (reverse coded; 1–5).

wb_psych_3 Meaning in life (1–5).

wb_psych_4 Purpose in life (1–5).

wb_psych_5 Hopefulness (1–5).

wb_psych_6 Anxiety (reverse coded; 1–5).

wb_soc_1 Social support (1–5).

wb_soc_2 Loneliness (reverse coded; 1–5).

wb_soc_3 Community belonging (1–5).

wb_overall_mean Mean of all well-being items (numeric).

wb_phys_mean Mean of physical well-being items (numeric).

wb_psych_mean Mean of psychological well-being items (numeric).

wb_soc_mean Mean of social well-being items (numeric).

age Age in years (integer).

gender Self-reported gender (character: e.g., "Male", "Female", "Other").

ses Socioeconomic status composite (numeric).

education Highest education level completed (ordinal integer).

ethnicity Self-reported ethnicity (character).

denomination Religious denomination (character).

gdp GDP per capita (PPP, USD) for country (numeric).

gdp_scaled Scaled GDP (mean = 0, sd = 1) used in analyses (numeric).

sample_type Recruitment method: e.g., "online panel", "student sample" (character).

compensation Type of compensation: e.g., "monetary", "entry into lottery" (character).

attention_check Score on embedded attention check task (integer).

Source

Hoogeveen, S., Sarafoglou, A., Aczel, B., et al. (2022). A many-analysts approach to the relation between religiosity and well-being. *Religion, Brain & Behavior*. doi:10.1080/2153599X.2023.2254980

Examples

```
library(dplyr)
data(marp)
# Dimensions
dim(marp)
# Quick overview
if (requireNamespace("dplyr", quietly = TRUE)) {
  library(dplyr)

  marp |>
    group_by(country) |>
    summarise(
      mean_wb = mean(wb_overall_mean, na.rm = TRUE),
      .groups = "drop"
    )
}
```

`mask_labels` *Mask categorical labels with random labels*

Description

Assigns random new labels to each unique value in a character or factor vector. The purpose is to blind data so analysts are not aware of treatment allocation or categorical outcomes. Each unique original value gets a random new label, and the assignment order is randomized to prevent correspondence with the original order.

Usage

```
mask_labels(x, prefix = "masked_group_")
```

Arguments

`x`	a character or factor vector
`prefix`	character string to use as prefix for masked labels. Default is "masked_group_"

Value

a vector of the same type as input with masked labels

See Also

`mask_variables` for masking multiple variables in a data frame, `mask_names` for masking variable names.

Examples

```
# Example with character vector
set.seed(123)
treatment <- c("control", "treatment", "control", "treatment")
mask_labels(treatment)

# Example with custom prefix
set.seed(456)
condition <- c("A", "B", "C", "A", "B", "C")
mask_labels(condition, prefix = "group_")

# Example with factor vector
set.seed(789)
ecology <- factor(c("Desperate", "Hopeful", "Desperate", "Hopeful"))
mask_labels(ecology)

# Using with dataset column
data(williams)
set.seed(123)
williams$ecology_masked <- mask_labels(williams$ecology)
head(williams[c("ecology", "ecology_masked")])
```

---

mask_names *Mask variable names with anonymous labels*

Description

Assigns new masked names to selected variables in a data frame. All selected variables are combined into a single set and renamed with a common prefix. To mask different variable groups with different prefixes, call the function separately for each group.

Usage

```
mask_names(data, ..., prefix)
```

Arguments

`data`	A data frame.
`...`	Columns to mask using tidyselect semantics. All arguments are combined into a single set. Each can be: • Bare column names (e.g., `var1, var2`) • A tidyselect expression (e.g., `starts_with("treatment_")`) • A character vector of column names (e.g., `c("var1", "var2")`)
`prefix`	character string to use as prefix for masked names. This becomes the base prefix, with numeric suffixes appended (e.g., `prefix = "treatment_"` produces "treatment_01", "treatment_02", etc.). The prefix is used as-is, so include a separator (e.g., underscore) if desired.

Value

A data frame with the specified variables renamed to masked names.

See Also

mask_labels for masking values in a vector, mask_variables for masking values in multiple variables.

Examples

```
df <- data.frame(
  treat_1 = c(1, 2, 3),
  treat_2 = c(4, 5, 6),
  outcome_a = c(7, 8, 9),
  outcome_b = c(10, 11, 12),
  id = 1:3
)

# Mask one set of variables
library(dplyr)
mask_names(df, starts_with("treat_"), prefix = "A_")

# Using character vectors
mask_names(df, c("treat_1", "treat_2"), prefix = "A_")

# Mask multiple sets separately
# Note that the order of masking matters
# Try to mix up the order of prefixes
# for different sets to ensure proper masking.
df |>
  mask_names(starts_with("treat_"), prefix = "B_") |>
  mask_names(starts_with("outcome_"), prefix = "A_")

# Example with the 'williams' dataset
data(williams)
set.seed(42)

williams |>
  mask_names(starts_with("SexUnres"), prefix = "A_") |>
  mask_names(starts_with("Impul"), prefix = "B_") |>
  colnames()
```

---

mask_variables *Mask categorical variables with random labels in a data frame*

Description

Applies masked labels to multiple categorical variables in a data frame using the mask_labels() function. Each variable gets independent random masked labels by default, or can optionally use the same masked labels across all selected variables.

Usage

```
mask_variables(data, ..., .across_variables = FALSE)
```

Arguments

`data`	a data frame
`...`	Columns to mask using tidyselect semantics. Each can be: • Bare column names (e.g., `var1`, `var2`) • A tidyselect expression (e.g., `starts_with("treat_")`) • A character vector of column names (e.g., `c("var1", "var2")`) • Multiple sets can be provided as separate arguments Only character and factor columns will be processed.
`.across_variables`	logical. If `TRUE`, all selected variables will use the same set of masked labels. If `FALSE` (default), each variable gets its own independent set of masked labels using the column name as prefix.

Value

A data frame with the specified categorical columns masked. Only character and factor columns can be processed.

See Also

`mask_labels` for masking a single vector, `mask_names` for masking variable names.

Examples

```
# Create example data
df <- data.frame(
  treatment = c("control", "intervention", "control"),
  outcome = c("success", "failure", "success"),
  score = c(1, 2, 3)  # numeric, won't be masked
)

set.seed(123)
# Independent masking for each variable (default - uses column names as
# prefixes)
# Using bare names
mask_variables(df, treatment, outcome)
# Or using character vector
mask_variables(df, c("treatment", "outcome"))

set.seed(456)
# Shared masking across variables
mask_variables(df, c("treatment", "outcome"), .across_variables = TRUE)

# Using tidyselect helpers
mask_variables(df, where(is.character))
```

```
# Example with multiple categorical columns
df2 <- data.frame(
  group = c("A", "B", "A", "B"),
  condition = c("ctrl", "test", "ctrl", "test")
)
set.seed(123)
result <- mask_variables(df2, c("group", "condition"))
print(result)

# Example with williams dataset (multiple categorical columns)
data(williams)
set.seed(456)
# Using bare names (recommended for interactive use)
williams_masked <- mask_variables(williams, subject, ecology)
head(williams_masked[c("subject", "ecology")])
```

`scramble_values` *Scramble a vector of values*

Description

Scramble a vector of values

Usage

```
scramble_values(x)
```

Arguments

x	a vector x

Value

the scrambled vector

See Also

`scramble_variables` for scrambling multiple variables in a data frame.

Examples

```
# Example with character vector
set.seed(123)

x <- letters[1:10]
scramble_values(x)

# Example with numeric vector
```

```
nums <- 1:5
scramble_values(nums)

# Scramble a column in the 'williams' dataset
data(williams)

# Simple scrambling of a single column
set.seed(123)
williams$ecology_scrambled <- scramble_values(williams$ecology)
head(williams[c("ecology", "ecology_scrambled")])
```

`scramble_variables` *Scrambling the content of several variables in a data frame*

Description

Scramble the values of several selected variables in a data frame simultaneously. Supports independent scrambling, joint scrambling, and within-group scrambling.

Usage

```
scramble_variables(
  data,
  ...,
  .groups = NULL,
  .together = FALSE,
  .byrow = FALSE
)
```

Arguments

`data`	a data frame
`...`	Columns to scramble using tidyselect semantics. Each can be: • Bare column names (e.g., `var1, var2`) • A tidyselect expression (e.g., `starts_with("treat_")`) • A character vector of column names (e.g., `c("var1", "var2")`) • Multiple sets can be provided as separate arguments
`.groups`	Optional grouping columns. Scrambling will be done within each group. Supports the same tidyselect syntax as column selection. Grouping columns must not overlap with the columns selected in `...`. If `data` is already a grouped `dplyr` data frame, existing grouping is ignored unless `.groups` is explicitly provided. Ignored if `.byrow = TRUE`.
`.together`	logical. If `TRUE`, variables are scrambled together as a unit per row. Values across different variables are kept intact but assigned to different rows. If `FALSE` (default), each variable is scrambled independently.

`.byrow` logical. If `TRUE`, values are scrambled rowwise across the selected columns. For each row, the values in the selected columns are shuffled among themselves. This requires selected columns to have compatible types. Cannot be combined with `.together = TRUE`.

Value

A data frame with the specified columns scrambled. If grouping is specified, scrambling is done within each group.

See Also

`scramble_values` for scrambling a single vector.

Examples

```
df <- data.frame(
  x = 1:6,
  y = letters[1:6],
  group = c("A", "A", "A", "B", "B", "B")
)

set.seed(123)
# Example without grouping. Variables scrambled across the entire data frame.
# Using bare names
df |> scramble_variables(x, y)
# Or using character vector
df |> scramble_variables(c("x", "y"))

# Example with .together = TRUE. Variables scrambled together as a unit per row.
df |> scramble_variables(c("x", "y"), .together = TRUE)

# Example with grouping. Variable only scrambled within groups.
df |> scramble_variables("y", .groups = "group")

# Example combining grouping and together parameters
df |> scramble_variables(c("x", "y"), .groups = "group", .together = TRUE)

# Example with tidyselect helpers
library(dplyr)
df |> scramble_variables(starts_with("x"))
df |> scramble_variables(where(is.numeric), .groups = "group")

# Example with the 'williams' dataset
data(williams)
williams |> scramble_variables(c("ecology", "age"))
williams |> scramble_variables(1:5)
williams |> scramble_variables(c("ecology", "age"), .groups = "gender")
williams |> scramble_variables(c(1, 2), .groups = 3)
williams |> scramble_variables(c("ecology", "age"), .together = TRUE)
williams |> scramble_variables(c("ecology", "age"), .groups = "gender", .together = TRUE)
```

```
# Rowwise scrambling
df_row <- data.frame(a = 1:3, b = 4:6, c = 7:9)
df_row |> scramble_variables(a, b, c, .byrow = TRUE)
```

`williams` *Stereotyping of high-wealth individuals across ecologies*

Description

Data from a study by Williams et al. testing whether high-wealth individuals are perceived as having faster life history strategies (e.g., more impulsive, less invested) when associated with "desperate" ecological conditions compared to "hopeful" ones.

Usage

```
data(williams)
```

Format

A data frame with 224 rows (one per participant) and 25 variables:

subject Unique subject identifier (integer).

ecology Experimental condition: `"Desperate"` or `"Hopeful"` (character).

age Participant's age in years (numeric).

gender Self-reported gender: 1 = Male, 2 = Female (numeric); may be recoded as factor.

duration_in_seconds Time taken to complete the survey (numeric).

attention_1 First attention check response: 1 = correct, 0 = incorrect (numeric).

attention_2 Second attention check response: 1 = correct, 0 = incorrect (numeric).

SexUnres_1 Perceived sexual unrestrictedness: "likely to have short-term relationships" (1–7 Likert).

SexUnres_2 "likely to engage in casual sex" (1–7).

SexUnres_3 "not interested in long-term commitment" (1–7).

SexUnres_4_r "faithful to romantic partners" — reverse-coded (1–7).

SexUnres_5_r "committed in relationships" — reverse-coded (1–7).

Impuls_1 "acts without thinking" (1–7).

Impuls_2_r "thinks carefully before acting" — reverse-coded (1–7).

Impul_3_r "plans ahead" — reverse-coded (1–7).% Note: likely typo in original; was Impul not Impuls?

Opport_1 "opportunities for long-term planning exist" (1–7).

Opport_2 "can save money for the future" (1–7).

Opport_3 "can make career plans" (1–7).

Opport_4 "can plan for retirement" (1–7).

Opport_5 "has control over future outcomes" (1–7).

Opport_6_r "life is unpredictable" — reverse-coded (1–7).

InvEdu_1_r "invests in education" — reverse-coded (1–7).

InvEdu_2_r "values academic achievement" — reverse-coded (1–7).

InvChild_1 "invests time and resources in children" (1–7).

InvChild_2_r "neglects parental responsibilities" — reverse-coded (1–7).

Source

Williams, S. A., Galak, J., & Kruger, D. J. (2019). The influence of ecology on social perceptions: When wealth signals faster life history strategies. *Evolutionary Behavioral Sciences*, 13(4), 313–325. doi:10.1037/ebs0000148

Data based on materials available at: https://osf.io/xyz12 (replace with real link if known)

Examples

```
data(williams)
str(williams)
table(williams$ecology)

# Compute composite scores (example)
if (requireNamespace("dplyr", quietly = TRUE)) {
  library(dplyr)

  williams_composites <- williams |>
    rowwise() |>
    mutate(
      sexual_unrestrictedness = mean(c(SexUnres_1, SexUnres_2, SexUnres_3,
                                       8 - SexUnres_4_r, 8 - SexUnres_5_r), na.rm = TRUE),
      impulsivity = mean(c(Impuls_1, 8 - Impuls_2_r, 8 - Impul_3_r), na.rm = TRUE),
      opportunity = mean(c(Opport_1, Opport_2, Opport_3, Opport_4, Opport_5,
                           8 - Opport_6_r), na.rm = TRUE),
      investment = mean(c(8 - InvEdu_1_r, 8 - InvEdu_2_r, InvChild_1,
                          8 - InvChild_2_r), na.rm = TRUE)
    ) |>
    ungroup()

  summary(williams_composites[ , c("sexual_unrestrictedness", "impulsivity")])
}
```

Index