Package: flourishR (via r-universe)

November 20, 2026

Title Tools for Reproducible Wellbeing and Survey Data Analysis

Version 0.1.0

Description Provides lightweight tools for common tasks in wellbeing and survey data analysis, including recoding survey missing values, converting labelled variables, checking missingness, scoring multi-item scales, conducting analyses across countries, and visualising cross-national and longitudinal outcomes. Functions are designed to support transparent and reproducible workflows with large-scale survey and longitudinal data.

License MIT + file LICENSE

Encoding UTF-8

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr, ggplot2, haven, rlang

Config/roxygen2/version 8.1.0

URL https://github.com/hilltaylor27/flourishR

BugReports https://github.com/hilltaylor27/flourishR/issues

NeedsCompilation no

Author Taylor G. Hill [aut, cre, cph]

Maintainer Taylor G. Hill <thill001@dundee.ac.uk>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-20 12:40:11 UTC

RemoteUrl https://github.com/cran/flourishR

RemoteRef HEAD

RemoteSha e0bdcafba0d5fd2cfbbeb216cb52d3165f8e05ab

Contents

check_missingness 2
label_to_factor 3
loop_countries 4
plot_cross_national 5
plot_longitudinal 6
recode_missing 7
score_scale 8
score_swemwbs 10

Index **12**

`check_missingness` *Summarise missing data patterns*

Description

Summarises the number and percentage of missing values for each variable in a data frame, or for a selected set of variables.

Usage

```
check_missingness(data, variables = NULL)
```

Arguments

`data`	A data frame.
`variables`	An optional character vector containing the names of variables to check. If `NULL`, all variables in `data` are checked.

Value

A data frame with one row per variable and three columns:

variable Variable name.

n_missing Number of missing values.

pct_missing Percentage of values that are missing, rounded to two decimal places.

Examples

```
dat <- data.frame(
  id = 1:6,
  life_satisfaction = c(7, 8, NA, 6, NA, 5),
  happiness = c(8, NA, 7, NA, 6, 5)
)

check_missingness(dat)
```

```
check_missingness(
  dat,
  variables = c("life_satisfaction", "happiness")
)
```

`label_to_factor` *Convert labelled survey variables to factors*

Description

Converts selected labelled survey variables to factors using `haven::as_factor()`. This is useful for survey datasets in which categorical variables are stored as numeric values with attached value labels.

Usage

```
label_to_factor(data, variables)
```

Arguments

`data`	A data frame containing the variables to convert.
`variables`	A character vector giving the names of variables to convert to factors.

Details

Only variables specified in `variables` are converted. All other variables are returned unchanged.

Survey data imported from formats such as SPSS or Stata may contain labelled variables in which numeric values have descriptive value labels attached. `label_to_factor()` converts these labelled values into factor levels, making them easier to use in descriptive analyses, tables, plots, and models.

The function returns an error if `data` is not a data frame, if `variables` is not supplied as a character vector, or if any requested variable is not present in the data.

Value

The original data frame with the selected variables converted to factors.

Examples

```
dat <- data.frame(
  id = 1:4,
  gender = haven::labelled(
    c(1, 2, 1, 3),
    labels = c(
      "Woman" = 1,
      "Man" = 2,
      "Other" = 3
    )
```

```
  )
)

label_to_factor(
  data = dat,
  variables = "gender"
)
```

`loop_countries` *Apply an analysis function across countries*

Description

Applies a user-supplied analysis function separately to each country represented in a data frame and combines the results into a single output.

Usage

```
loop_countries(data, country_var, analysis_function, ...)
```

Arguments

`data`	A data frame.
`country_var`	A character value giving the name of the variable identifying countries.
`analysis_function`	A function to apply separately to the data for each country. The function must accept a `data` argument and return an object that can be combined using `dplyr::bind_rows()`.
`...`	Additional arguments passed to `analysis_function`.

Details

Missing values in the country variable are excluded. Additional arguments supplied through `...` are passed to the analysis function.

Value

A data frame containing the combined outputs from `analysis_function`, with the country variable as the first column.

Examples

```
dat <- data.frame(
  country = c(
    "England", "England",
    "Scotland", "Scotland",
    "Wales", "Wales"
  ),
```

```
  happiness = c(7, NA, 6, 7, 5, NA)
)

mean_happiness <- function(data) {
  data.frame(
    mean_happiness = mean(data$happiness, na.rm = TRUE)
  )
}

loop_countries(
  data = dat,
  country_var = "country",
  analysis_function = mean_happiness
)
```

`plot_cross_national` *Plot mean outcomes across countries*

Description

Creates a horizontal bar chart comparing the mean of an outcome across countries. Countries are ordered by mean outcome.

Usage

```
plot_cross_national(
  data,
  country_var,
  outcome_var,
  plot_title = NULL,
  y_label = NULL
)
```

Arguments

`data`	A data frame.
`country_var`	A character value giving the name of the variable identifying countries.
`outcome_var`	A character value giving the name of the numeric outcome variable to plot.
`plot_title`	Optional plot title. If `NULL`, a title is generated from `outcome_var`.
`y_label`	Optional label for the outcome axis. If `NULL`, a label is generated from `outcome_var`.

Details

Countries with no non-missing observations for the outcome are excluded from the plot.

Value

A ggplot object containing a horizontal bar chart of mean outcome values by country.

Examples

```
dat <- data.frame(
  country = c(
    "England", "England",
    "Scotland", "Scotland",
    "Wales", "Wales"
  ),
  happiness = c(7, 8, 6, 7, 5, 6)
)

plot_cross_national(
  data = dat,
  country_var = "country",
  outcome_var = "happiness"
)
```

---

plot_longitudinal *Plot longitudinal outcome trajectories*

Description

Plots mean outcome values across waves with standard error bars. Individual participant trajectories can optionally be displayed behind the mean trajectory.

Usage

```
plot_longitudinal(
  data,
  id_var,
  wave_var,
  outcome_var,
  show_individuals = FALSE,
  plot_title = NULL,
  y_label = NULL
)
```

Arguments

data	A data frame containing longitudinal data.
id_var	A character value giving the name of the participant identifier variable.
wave_var	A character value giving the name of the wave or time variable.
outcome_var	A character value giving the name of the numeric outcome variable.

show_individuals
: Logical. If TRUE, individual participant trajectories are displayed. Defaults to FALSE.

plot_title
: Optional plot title. If NULL, a title is generated from outcome_var.

y_label
: Optional label for the outcome axis. If NULL, a label is generated from outcome_var.

Details

Wave means and standard errors are calculated using non-missing observations for the outcome.

Value

A ggplot object containing the longitudinal trajectory plot.

Examples

```
dat <- data.frame(
  id = rep(1:3, each = 3),
  wave = rep(1:3, times = 3),
  happiness = c(
    6, 7, 8,
    5, 6, 7,
    7, 7, 8
  )
)

plot_longitudinal(
  data = dat,
  id_var = "id",
  wave_var = "wave",
  outcome_var = "happiness"
)
```

---

recode_missing *Recode survey-specific missing values to NA*

Description

Replaces specified survey missing-value codes with NA in selected variables.

Usage

```
recode_missing(data, variables, missing_codes = c(-9, -8, -7, -2, -1))
```

Arguments

`data`	A data frame containing the variables to recode.
`variables`	A character vector giving the names of variables in which missing-value codes should be recoded.
`missing_codes`	A vector of values to replace with `NA`. Defaults to `c(-9, -8, -7, -2, -1)`.

Details

Many survey datasets use numeric codes such as `-9`, `-8`, or `-1` to represent different forms of missing data. `recode_missing()` converts these codes to standard R missing values before analysis.

Only variables specified in `variables` are modified. All other variables are returned unchanged.

By default, `recode_missing()` treats `-9`, `-8`, `-7`, `-2`, and `-1` as missing-value codes. Alternative codes can be supplied using `missing_codes`.

Existing `NA` values are preserved. Values in variables not specified in `variables` are not modified.

The function returns an error if `data` is not a data frame, if `variables` is not supplied as a character vector, or if any requested variable is not present in the data.

Value

The original data frame with specified missing-value codes replaced by `NA` in the selected variables.

Examples

```
dat <- data.frame(
  life_satisfaction = c(7, 8, -9, 6, -8),
  happiness = c(8, -9, 7, -1, 6),
  age = c(25, 30, 35, 40, 45)
)

recode_missing(
  data = dat,
  variables = c("life_satisfaction", "happiness")
)
```

`score_scale` *Score a multi-item scale*

Description

Scores a multi-item scale using a specified minimum-item rule and, optionally, a valid response range. Values outside the valid range are treated as missing for scoring.

Usage

```
score_scale(data, items, prefix, min_items = length(items), valid_range = NULL)
```

Arguments

`data`	A data frame containing the scale items.
`items`	A character vector containing the names of the scale items.
`prefix`	A single non-empty character value used to name the generated scoring variables.
`min_items`	Minimum number of valid items required to calculate a score. Defaults to the total number of items, requiring complete data.
`valid_range`	Optional numeric vector of length two giving the minimum and maximum valid item values. Values outside this range are treated as missing for scoring. If `NULL`, no range restriction is applied.

Details

The function returns the original data with variables containing the number of valid items, sum score, mean score, and prorated sum score.

Value

The original data frame with four additional variables. Variable names are constructed using the supplied `prefix`:

`prefix.n.valid` Number of valid item responses.

`prefix.sum` Sum of valid item responses when the minimum-item rule is met.

`prefix.mean` Mean of valid item responses when the minimum-item rule is met.

`prefix.prorated` Mean of valid item responses multiplied by the total number of scale items.

In these names, `prefix` is replaced by the value supplied to the function. For example, `prefix = "wellbeing"` generates `wellbeing.n.valid`, `wellbeing.sum`, `wellbeing.mean`, and `wellbeing.prorated`.

Examples

```
dat <- data.frame(
  item.1 = c(1, 2, 3),
  item.2 = c(2, 3, NA),
  item.3 = c(3, 4, 5),
  item.4 = c(4, 5, 4)
)

score_scale(
  data = dat,
  items = c("item.1", "item.2", "item.3", "item.4"),
  prefix = "scale",
  min_items = 3,
  valid_range = c(1, 5)
)
```

`score_swemwbs` *Score the Short Warwick-Edinburgh Mental Wellbeing Scale*

Description

Scores the seven-item Short Warwick-Edinburgh Mental Wellbeing Scale (SWEMWBS) using `score_scale()`.

Usage

```
score_swemwbs(data, items, min_items = 5)
```

Arguments

`data`	A data frame containing the SWEMWBS items.
`items`	A character vector containing the names of exactly seven SWEMWBS items.
`min_items`	Minimum number of valid items required to calculate a score. Defaults to 5.

Details

SWEMWBS items are expected to have values from 1 to 5. Values outside this range are treated as missing for scoring. By default, at least five of the seven items must be valid for a score to be calculated.

The function calculates raw sum, mean, and prorated sum scores. It does not apply the published SWEMWBS raw-to-metric score conversion. The resulting scores should not be interpreted as transformed SWEMWBS metric scores.

Value

The original data frame with four additional variables:

swemwbs.n.valid Number of valid SWEMWBS item responses.

swemwbs.sum Sum of valid item responses when the minimum-item rule is met.

swemwbs.mean Mean of valid item responses when the minimum-item rule is met.

swemwbs.prorated Mean of valid item responses multiplied by seven.

Examples

```
dat <- data.frame(
  sw1 = c(1, 5),
  sw2 = c(2, 5),
  sw3 = c(3, 5),
  sw4 = c(4, 5),
  sw5 = c(5, 5),
  sw6 = c(1, 5),
  sw7 = c(2, 5)
)
```

```
score_swemwbs(
  data = dat,
  items = c("sw1", "sw2", "sw3", "sw4", "sw5", "sw6", "sw7")
)
```

Index

check_missingness, 2

haven::as_factor(), *3*

label_to_factor, 3
loop_countries, 4

plot_cross_national, 5
plot_longitudinal, 6

recode_missing, 7

score_scale, 8
score_swemwbs, 10