Package 'flourishR'

November 26, 2026

Title Tools for Reproducible Wellbeing and Survey Data Analysis

Version 0.2.0

Description Tools for cross-national and longitudinal survey research with dedicated support for the Global Flourishing Study (GFS). Provides data preparation, documented missing-value recoding, country labels, personality scoring, survey design, retention analysis, country-specific regression, random-effects meta-analysis, longitudinal transformations, repeated-measures models and visualisation. Supports reproducible workflows using country-specific survey weights and unweighted mixed-effects models.

License MIT + file LICENSE

Encoding UTF-8

Suggests knitr, rmarkdown, lme4, metafor, survey, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr (>= 1.0.0), ggplot2 (>= 3.4.0), haven, rlang

Config/roxygen2/version 8.1.0

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

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

Depends R (>= 3.5)

VignetteBuilder knitr

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2026-11-26 14:10:08 UTC

Contents

change_scores 2
check_missingness 3

check_retention 4
country_moderation 5
create_lags 6
describe_countries 6
gfs_countries 7
gfs_design 8
gfs_dictionary 9
gfs_prepare 10
gfs_weights 11
join_country_data 12
label_to_factor 13
loop_countries 14
loop_longitudinal 15
meta_countries 17
plot_country_effects 18
plot_cross_national 18
plot_longitudinal 20
plot_missingness 21
plot_retention 22
recode_missing 22
score_tipi 23
within_between 24

Index **26**

`change_scores` *Calculate change from an earlier wave*

Description

Calculate change from an earlier wave

Usage

```
change_scores(data, variables, from = 1L, to = 3L, id = "ID", wave = "wave")
```

Arguments

`data`	Long-format participant-wave data.
`variables`	Character vector of variables.
`from`	Earlier integer wave.
`to`	Later integer wave.
`id`	Participant identifier column.
`wave`	Integer wave column.

Value

Rows at the later wave with change columns named variable_change_from_to. Change is later minus earlier.

Examples

```
d <- data.frame(ID = rep(1:2, each = 3), wave = rep(1:3, 2), x = 1:6)
change_scores(d, "x")
```

---

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, by = 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.
by	Optional character vector of grouping variables. Grouped output includes the group size n.

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")
)
```

`check_retention` *Examine longitudinal survey retention*

Description

Summarise survey participation and retention across longitudinal waves, optionally within groups.

Usage

```
check_retention(
  data,
  id = "ID",
  wave = "wave",
  participation = "WAVE",
  participated_values = NULL,
  group = NULL,
  baseline = 1
)
```

Arguments

`data`	Long-format longitudinal data.
`id`	Character. Participant identifier variable.
`wave`	Character. Survey wave variable.
`participation`	Character. Participation indicator variable. By default, nonmissing values indicate participation.
`participated_values`	Optional vector of values indicating participation.
`group`	Optional grouping variable.
`baseline`	Baseline wave identifier.

Details

Retention is calculated relative to participants observed at baseline. Cumulative retention requires participation in every supplied wave from baseline to the current wave.

For prepared GFS data, the WAVE variable identifies participation in each annual survey.

Value

A data frame containing retention statistics.

Examples

```
data <- data.frame(
  ID = rep(1:3, each = 3),
  wave = rep(1:3, 3),
  WAVE = c(1, 2, 3, 1, NA, 3, 1, 2, NA)
)

check_retention(data)
```

`country_moderation` *Test heterogeneity of country-specific coefficients*

Description

Test heterogeneity of country-specific coefficients

Usage

```
country_moderation(results, term, country_var = "COUNTRY")
```

Arguments

`results`	Country model results from loop_countries.
`term`	Coefficient name.
`country_var`	Country column name.

Details

Tests equality of independent, comparable country coefficients using an asymptotic inverse-variance test. This is a heterogeneity test across separately fitted models, not a pooled individual-level country interaction model.

Value

Data frame containing Cochran Q, degrees of freedom and p-value.

Examples

```
r <- data.frame(COUNTRY = c("Japan", "Sweden", "Brazil"), term = "x",
                estimate = c(.2, .4, .3), std.error = .1, status = "ok")
country_moderation(r, "x")
```

create_lags *Create exact-wave lagged variables*

Description

Create exact-wave lagged variables

Usage

```
create_lags(data, variables, id = "ID", wave = "wave", lag = 1L)
```

Arguments

data	Long-format participant-wave data.
variables	Character vector of variables.
id	Participant identifier column.
wave	Integer wave column.
lag	Positive integer number of waves.

Value

Original data with columns ending in _lag followed by the lag number. Missing target waves produce missing values.

Examples

```
d <- data.frame(ID = rep(1:2, each = 3), wave = rep(1:3, 2), x = 1:6)
create_lags(d, "x")
```

describe_countries *Describe numeric variables by GFS country*

Description

Describe numeric variables by GFS country

Usage

```
describe_countries(
  data,
  variables,
  wave = 1,
  weighted = TRUE,
  conf.level = 0.95,
  lonely_psu = NULL
)
```

Arguments

data	Prepared GFS data.
variables	Numeric variable names.
wave	Annual wave.
weighted	Use the annual cross-sectional survey weight and design.
conf.level	Confidence level.
lonely_psu	Optional survey single-PSU variance method; NULL preserves current settings.

Value

Data frame of country means, standard errors, intervals, sample sizes and status.

Examples

```
d <- data.frame(COUNTRY = rep(c("Japan", "Sweden"), each = 4), x = 1:8)
describe_countries(d, "x", weighted = FALSE)
```

---

gfs_countries *Global Flourishing Study country labels*

Description

Return GFS country identifiers and labels or convert numerical country identifiers into readable labels.

Usage

```
gfs_countries(x = NULL, as_factor = FALSE, unknown = c("NA", "keep", "error"))
```

Arguments

x	Optional vector of GFS country identifiers. If NULL, the complete country lookup table is returned.
as_factor	Logical. Return country labels as a factor.
unknown	How to handle unrecognised country identifiers.

Value

A data frame, character vector or factor.

Examples

```
gfs_countries()
gfs_countries(c(1, 9, 20, 22))
gfs_countries(c(1, 9, 20), as_factor = TRUE)
```

gfs_design *Construct a Global Flourishing Study survey design*

Description

Construct a country-specific survey design using GFS sampling weights, strata and primary sampling units.

Usage

```
gfs_design(
  data,
  country,
  waves = 1,
  type = NULL,
  analysis_wave = max(waves),
  nest = TRUE
)
```

Arguments

data	Prepared GFS data in wide or long format.
country	Country identifier or country label.
waves	Annual waves determining the survey weight.
type	Optional survey weight type.
analysis_wave	Wave to retain from long-format data.
nest	Logical. Nest PSU identifiers within strata.

Details

The returned design contains one record per participant with a valid survey weight. For long-format data, analysis_wave determines which participant record is retained.

Weights support within-country inference. Variances use the supplied strata and PSU identifiers with the survey package's variance settings. The function does not prescribe a singleton-PSU adjustment; choose one based on the sampling design. Records with missing weights are excluded.

This function constructs a stratified and clustered survey design. It does not construct a repeated-measures or multilevel longitudinal survey model.

Value

A survey.design object.

Examples

```
if (requireNamespace("survey", quietly = TRUE)) {
  d <- data.frame(ID = 1:8, COUNTRY = "Japan", STRATA = rep(1:2, each = 4),
                  PSU = rep(1:4, each = 2), ANNUAL_WEIGHT_C1 = 1, x = 1:8)
  survey::svymean(~x, gfs_design(d, "Japan"))
}
```

`gfs_dictionary` *Global Flourishing Study variable dictionary*

Description

Access variable availability, missing response codes, country identifiers and survey weight metadata for the Global Flourishing Study.

Usage

```
gfs_dictionary(
  variable = NULL,
  wave = NULL,
  search = NULL,
  type = c("variables", "missing", "countries", "weights"),
  release = 3L
)
```

Arguments

`variable`	Character vector of GFS variable names. If NULL, all variables are returned.
`wave`	Integer vector specifying annual survey waves (1, 2 or 3).
`search`	Character string used to search variable names.
`type`	Metadata table to return.
`release`	Codebook release for missing response metadata, either 2 or 3.

Value

A data frame containing the requested metadata.

References

Global Flourishing Study documentation: https://www.cos.io/gfs-resources

Examples

```
gfs_dictionary()
gfs_dictionary(variable = "LIFE_SAT")
gfs_dictionary(wave = 3)
gfs_dictionary(search = "REL")
gfs_dictionary(type = "missing", variable = "AGE")
gfs_dictionary(type = "weights")
```

`gfs_prepare` *Prepare Global Flourishing Study data*

Description

Prepare combined annual Global Flourishing Study data for cross-sectional and longitudinal analyses. The function identifies wave-specific variables, recodes documented missing values, labels countries and returns data in wide or long format.

Usage

```
gfs_prepare(
  data,
  waves = 1:3,
  variables = NULL,
  format = c("long", "wide"),
  recode_missing = TRUE,
  label_countries = TRUE,
  keep_weights = TRUE,
  keep_design = TRUE,
  release = 3L
)
```

Arguments

`data`	A data frame or path to a GFS CSV file.
`waves`	Integer vector specifying annual waves.
`variables`	Character vector of variable names without annual wave suffixes. NULL retains all available variables.
`format`	Character. Either "long" or "wide".
`recode_missing`	Logical. Recode documented GFS missing-value codes to NA.
`label_countries`	Logical. Convert country identifiers into country labels.
`keep_weights`	Logical. Retain available survey weights.
`keep_design`	Logical. Retain strata and PSU identifiers.
`release`	Integer. GFS codebook release, either 2 or 3.

Details

Long-format data contain one record per participant per selected wave. Unsuffixed participant-level variables are retained across waves. Variables measured exclusively at baseline retain their original Year 1 suffix.

Wave-specific variables are not carried forward between waves. Records are retained irrespective of participation in individual waves.

Survey weights retain their original names and values. Item response directions are not changed.

Value

A data frame containing prepared GFS data.

Examples

```
data <- data.frame(
  ID = 1:3,
  COUNTRY = c(1, 9, 20),
  LIFE_SAT_Y1 = c(7, 8, 99),
  LIFE_SAT_Y2 = c(8, 9, 6),
  LIFE_SAT_Y3 = c(9, 8, 7)
)

gfs_prepare(data)
gfs_prepare(data, format = "wide")
gfs_prepare(data, waves = c(1, 3))
```

---

gfs_weights *Select Global Flourishing Study survey weights*

Description

Identify the appropriate annual GFS survey weight for cross-sectional, longitudinal or all-record analyses.

Usage

```
gfs_weights(waves = 1, type = NULL, data = NULL)
```

Arguments

waves	Integer vector specifying annual waves.
type	Character. Weight type. If NULL, the type is determined from the requested waves.
data	Optional data frame in which to validate the selected weight.

Details

GFS survey weights are intended for within-country inference. Longitudinal weights correspond to specific combinations of annual survey participation.

Value

A character string containing the weight variable name.

Examples

```
gfs_weights(1)
gfs_weights(3)
gfs_weights(1:3)
gfs_weights(3, type = "all records")
```

`join_country_data` *Join country-level contextual data*

Description

Join country-level contextual data

Usage

```
join_country_data(
  data,
  context,
  country_var = "COUNTRY",
  context_country = country_var,
  gfs_labels = TRUE,
  unmatched = c("error", "keep")
)
```

Arguments

`data`	Participant or country-level data.
`context`	One row per country containing contextual variables.
`country_var`	Country column in data.
`context_country`	Country column in context.
`gfs_labels`	Translate GFS codes to labels before matching.
`unmatched`	Treatment of unmatched countries.

Value

Data with contextual columns appended, preserving row order.

Examples

```
d <- data.frame(COUNTRY = c("Japan", "Sweden"), x = 1:2)
context <- data.frame(COUNTRY = c("Japan", "Sweden"), z = c(3, 4))
join_country_data(d, context)
```

---

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 = NULL,
  ...,
  formula = NULL,
  wave = NULL,
  weighted = FALSE,
  family = stats::gaussian(),
  conf.level = 0.95,
  lonely_psu = NULL
)
```

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`.
`formula`	Optional model formula, instead of analysis_function.
`wave`	Annual wave to analyse.
`weighted`	Use a country-specific GFS survey design.
`family`	GLM family.
`conf.level`	Confidence level.
`lonely_psu`	Optional survey single-PSU variance method; NULL preserves current settings.

Details

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

In formula mode, returns coefficient estimates, standard errors, confidence intervals, sample sizes and model status. Additional arguments are passed to glm or svyglm.

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
)
```

---

loop_longitudinal *Fit prospective country-specific survey regressions*

Description

Fit prospective country-specific survey regressions

Usage

```
loop_longitudinal(
  data,
  formula,
  waves = 1:3,
```

```
  family = stats::gaussian(),
  conf.level = 0.95,
  lonely_psu = NULL,
  ...,
  model = c("prospective", "mixed")
)
```

Arguments

data	Prepared long-format GFS data, including any lagged or change variables used in the formula.
formula	Model formula evaluated at the final selected wave.
waves	Annual waves defining longitudinal weight eligibility; either 1:2 or 1:3.
family	GLM family.
conf.level	Confidence level.
lonely_psu	Optional single-PSU variance method; NULL preserves current settings.
...	Additional arguments to survey::svyglm.
model	Prospective survey regression or unweighted Gaussian random-intercept model.

Details

Prospective mode uses the longitudinal weight for the selected waves and fits endpoint regressions. Prepare lagged predictors before calling. Mixed mode fits unweighted Gaussian participant random-intercept models with Wald confidence intervals. It does not incorporate the GFS survey design. Singularity is reported separately from other convergence problems. In prospective mode, lonely_psu controls the survey variance adjustment temporarily; the caller's setting is restored. The appropriate adjustment depends on the sampling design.

Value

Country coefficient table, with one model per country and one record per eligible participant in each model.

Examples

```
if (requireNamespace("survey", quietly = TRUE)) {
  d <- data.frame(ID = rep(1:12, each = 2), COUNTRY = "Japan",
                  wave = rep(1:2, 12), STRATA = rep(rep(1:2, each = 6), each = 2),
                  PSU = rep(rep(1:6, each = 2), each = 2), ANNUAL_WEIGHT_L2 = 1)
  d$x <- rep(1:12, each = 2)
  d$y <- d$x + sin(d$ID) + d$wave
  loop_longitudinal(d, y ~ x, waves = 1:2)
}
```

`meta_countries` *Pool country-specific coefficients*

Description

Pool country-specific coefficients

Usage

```
meta_countries(
  results,
  term,
  country_var = "COUNTRY",
  conf.level = 0.95,
  moderators = NULL
)
```

Arguments

`results`	Country model results from loop_countries.
`term`	Coefficient name.
`country_var`	Country column name.
`conf.level`	Confidence level.
`moderators`	Optional one-sided formula for country-level meta-regression using columns in results.

Details

Uses random-effects meta-analysis with Knapp-Hartung inference. Estimates describe the distribution of included country effects, not a population-weighted global effect. Coefficients must be on comparable scales with consistent model coding.

Value

Data frame containing REML pooled estimate, confidence interval and heterogeneity statistics.

Examples

```
r <- data.frame(COUNTRY = c("Japan", "Sweden", "Brazil"), term = "x",
                estimate = c(.2, .4, .3), std.error = .1, status = "ok")
if (requireNamespace("metafor", quietly = TRUE)) meta_countries(r, "x")
```

plot_country_effects *Plot country-specific coefficients*

Description

Plot country-specific coefficients

Usage

```
plot_country_effects(
  results,
  term,
  pooled = NULL,
  country_var = "COUNTRY",
  x_label = "Regression coefficient"
)
```

Arguments

results	Country model results.
term	Coefficient to plot.
pooled	Optional intercept-only meta_countries result for the same term.
country_var	Country column.
x_label	Coefficient axis label.

Value

A ggplot object.

Examples

```
r <- data.frame(COUNTRY = c("Japan", "Sweden"), term = "x",
                estimate = c(.2, .4), std.error = .1, status = "ok",
                conf.low = c(.004, .204), conf.high = c(.396, .596))
plot_country_effects(r, "x")
```

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,
  estimates = FALSE,
  label_countries = FALSE
)
```

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.
estimates	Logical; use precomputed country means and confidence intervals.
label_countries	Logical; translate GFS country codes.

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,
  country_var = NULL,
  estimates = FALSE,
  label_countries = FALSE
)
```

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`.
`country_var`	Optional country column for separate trajectories.
`estimates`	Logical; use precomputed means and confidence intervals.
`label_countries`	Logical; translate GFS country codes.

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"
)
```

---

plot_missingness *Plot item missingness*

Description

Plot item missingness

Usage

```
plot_missingness(data, country_var = "COUNTRY", wave_var = "wave")
```

Arguments

data	Grouped check_missingness output.
country_var	Country column.
wave_var	Optional wave column.

Value

A ggplot object.

Examples

```
d <- data.frame(COUNTRY = rep(c("Japan", "Sweden"), each = 2), wave = 1, x = c(1, NA, 2, 3))
plot_missingness(check_missingness(d, "x", by = c("COUNTRY", "wave")))
```

plot_retention *Plot longitudinal retention*

Description

Plot longitudinal retention

Usage

```
plot_retention(data, group = "group", cumulative = FALSE)
```

Arguments

data	Output of check_retention.
group	Group column.
cumulative	Plot cumulative rather than wave-specific retention.

Value

A ggplot object.

Examples

```
d <- data.frame(ID = rep(1:3, each = 2), wave = rep(1:2, 3), WAVE = c(1,2,1,2,1,NA))
plot_retention(check_retention(d))
```

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_tipi *Score the Ten-Item Personality Inventory*

Description

Calculate five personality trait means from ten TIPI items.

Usage

```
score_tipi(data, items = NULL, coding = c("gfs", "standard"))
```

Arguments

data	A data frame containing numeric item responses.
items	Character vector of ten column names in TIPI item order. NULL detects TRAITS1 through TRAITS10, with or without _Y1 suffixes.
coding	Either "gfs" (1 = strongly agree) or "standard" (7 = strongly agree).

Details

Both items must be observed for each trait. Scores range from 1 to 7. GFS missing codes -98, 98 and 99 are converted to NA for scoring only. Other invalid responses cause an error. Baseline items retained in long GFS data produce baseline scores on each row, not repeated personality measurements. Original items are unchanged.

Value

The input data frame with five additional numeric columns: extraversion, agreeableness, conscientiousness, emotional_stability, and openness.

References

Gosling, S. D., Rentfrow, P. J., and Swann, W. B. (2003). A very brief measure of the Big-Five personality domains. Journal of Research in Personality, 37, 504-528. doi:10.1016/S0092-6566(03)00046-1.

Examples

```
d <- as.data.frame(as.list(stats::setNames(rep(4, 10), paste0("TRAITS", 1:10))))
score_tipi(d)
```

---

within_between *Decompose time-varying variables into within- and between-person components*

Description

Decompose time-varying variables into within- and between-person components

Usage

```
within_between(data, variables, id = "ID", wave = "wave")
```

Arguments

data	Long-format participant-wave data.
variables	Character vector of variables.
id	Participant identifier column.
wave	Integer wave column.

Details

Person means are calculated before any subsequent model-specific complete-case selection. Restrict the panel first if means should reflect only the intended analytical sample. A single observed value gives a zero within-person deviation; entirely missing histories give missing components.

Value

Original data with _between person means and _within deviations. Means use observed values in the supplied panel.

Examples

```
d <- data.frame(ID = rep(1:2, each = 3), wave = rep(1:3, 2), x = 1:6)
within_between(d, "x")
```

Index

change_scores, 2
check_missingness, 3
check_retention, 4
country_moderation, 5
create_lags, 6

describe_countries, 6

gfs_countries, 7
gfs_design, 8
gfs_dictionary, 9
gfs_prepare, 10
gfs_weights, 11

haven::as_factor(), *13*

join_country_data, 12

label_to_factor, 13
loop_countries, 14
loop_longitudinal, 15

meta_countries, 17

plot_country_effects, 18
plot_cross_national, 18
plot_longitudinal, 20
plot_missingness, 21
plot_retention, 22

recode_missing, 22

score_tipi, 23

within_between, 24