Package 'normref'

May 9, 2026

Title Continuous Norming

Version 0.1.1

Description A toolbox for continuous norming of psychological and educational tests, supporting regression-based norming where norms can vary as a continuous function of age or another norm predictor. Norms are estimated using Generalized Additive Models for Location, Scale, and Shape (GAMLSS), enabling flexible modelling of the full score distribution in a normative sample. The package supports applications in psychometrics and psychological testing, and includes functions for model selection, reliability estimation, norm calculation, including confidence intervals, and sample size planning. For more details, see Timmerman et al. (2021) <doi:10.1037/met0000348>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports gamlss, gamlss.dist, ggplot2, lpSolve, openxlsx2, rlang, Rdpack, stats, utils, withr

RdMacros Rdpack

Suggests knitr, rmarkdown, testthat (>= 3.0.0), gamlss.tr, future, future.apply

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Klazien de Vries [aut] (ORCID: <https://orcid.org/0009-0007-9302-1562>), Hannah Heister [aut] (ORCID: <https://orcid.org/0009-0001-1512-5549>), Julian Urban [aut] (ORCID: <https://orcid.org/0000-0001-8886-4724>), Lieke Voncken [ctb] (ORCID: <https://orcid.org/0000-0002-6710-271X>), Marieke Timmerman [aut, cre] (ORCID: <https://orcid.org/0000-0003-3480-5918>)

Maintainer Marieke Timmerman <m.e.timmerman@rug.nl>

Repository CRAN

Date/Publication 2026-04-10 13:30:03 UTC

Contents

`centiles_bin` *Plot centiles of a fitted GAMLSS model (binomial-type)*

Description

`centiles_bin()` plots centile curves and the sample data for binomial-type distributions (see gamlss::.gamlss.bi.list) based on a fitted `GAMLSS` object.

Usage

```
centiles_bin(
  model,
  xvar,
  cent = c(0.4, 2, 10, 25, 50, 75, 90, 98, 99.6),
  legend = TRUE,
  ylab = "y",
  xlab = "x",
  main = NULL,
  main.gsub = "@",
  xleg = min(xvar),
  yleg = max(model$y),
  xlim = range(xvar),
  ylim = range(model$y),
  save = FALSE,
  plot = TRUE,
  points = TRUE,
  pch = 15,
  cex = 0.5,
  col = "grey",
```

```
  col.centiles = seq_along(cent) + 2,
  lty.centiles = 1,
  lwd.centiles = 1,
  colors = "rainbow",
  ...
)
```

Arguments

model	a GAMLSS fitted model, for example the result of fb_select().
xvar	the unique explanatory variable
cent	a vector with elements the % centile values for which the centile curves have to be evaluated
legend	whether a legend is required in the plot or not, the default is legent=TRUE
ylab	the y-variable label
xlab	the x-variable label
main	the main title here as character. If NULL the default title "centile curves using NO" (or the relevant distributions name) is shown
main.gsub	if the main.gsub (with default "@") appears in the main title then it is substituted with the default title.
xleg	position of the legend in the x-axis
yleg	position of the legend in the y-axis
xlim	the limits of the x-axis
ylim	the limits of the y-axis
save	whether to save the sample percentages or not with default equal to FALSE. In this case the sample percentages are printed but are not saved
plot	whether to plot the centiles. This option is useful for centile.split
points	whether the data points should be plotted, default is TRUE for centiles() and FALSE for centiles.fan()
pch	the character to be used as the default in plotting points see par
cex	size of character see par
col	plotting colour see par
col.centiles	Plotting colours for the centile curves
lty.centiles	line type for the centile curves
lwd.centiles	The line width for the centile curves
colors	the different colour schemes to be used for the fan-chart. The following are available c("cm","gray", "rainbow", "heat", "terrain", "topo"),
...	for extra arguments

Value

No return value, only graphical output.

See Also

fb_select()

Examples

```
data("ids_data")

mydata_BB_y14 <- shape_data(
  data       = ids_data,
  age_name   = "age",
  score_name = "y14",
  family     = "BB"
)

mod_BB_y14 <- fb_select(
  data       = mydata_BB_y14,
  age_name   = "age",
  score_name = "shaped_score",
  family     = "BB",
  selcrit    = "BIC"
)

centiles_bin(mod_BB_y14, xvar = age)
```

---

composite_shape *Shape data for a composite scale based on normalized Z-scores*

Description

composite_shape() creates a data.frame with age values and the sum of normalized z-scores from multiple NormTable objects, suitable for use as input to fb_select().

Usage

```
composite_shape(normtables)
```

Arguments

normtables	list of NormTable objects created by normtable_create(). Each must contain znorm_sample and norm_sample.

Value

A data.frame with:

- age: Age values from the first NormTable
- z_sum: Unweighted sum of normalized z-scores across all objects

See Also

shape_data(), fb_select(), normtable_create()

Examples

```
invisible(data("ids_data"))

# Example with two normtables
mydata1 <- shape_data(ids_data, age_name = "age", score_name = "y7", family = "BCPE")
mod1 <- fb_select(mydata1, age_name = "age", score_name = "shaped_score",
                  family = "BCPE", selcrit = "BIC")
norm1 <- normtable_create(mod1, mydata1, age_name = "age", score_name = "shaped_score")

mydata2 <- shape_data(ids_data, age_name = "age", score_name = "y14", family = "BCPE")
mod2 <- fb_select(mydata2, age_name = "age", score_name = "shaped_score",
                  family = "BCPE", selcrit = "BIC")
norm2 <- normtable_create(mod2, mydata2, age_name = "age", score_name = "shaped_score")

composite_data <- composite_shape(list(norm1, norm2))
```

---

cotapp_data *cotapp data*

Description

The data are perturbed variants of the scores on the raw speed of block 1 (`rt`) and the raw number of errors of block 7 (`error`) of the normative sample of the cotapp test (Rommelse et al., 2018). More information on the cotapp test: https://www.boom.nl/productgroep/101-45_COTAPP

Usage

```
data(cotapp_data)
```

Format

A dataframe with three columns:

`age` age in years

`rt` reaction time: scores on the raw speed of block 1

`error` number of errors of block 7

References

Rommelse N, Brinkman A, Slaats-Willemse D, Timmerman ME, Voncken L, de Zeeuw P, Luman M, Hartman C (2020). "De Cognitieve Test Applicatie (COTAPP): geavanceerde computertest voor het meten van aandacht, informatieverwerking en executieve functies bij kinderen." *Kind en Adolescent*, **41**, 50–80.

`different_rel` *Estimate reliability across multiple window widths and age steps*

Description

Estimates reliability curves across various combinations of window widths and age step sizes, with optional per-individual estimation.

Usage

```
different_rel(
  data,
  item_variables,
  age_name,
  step_window,
  min_agegroup = NULL,
  max_agegroup = NULL,
  step_agegroup,
  include_window_per_person = FALSE,
  complete.obs = TRUE
)
```

Arguments

`data`	data.frame containing item scores and age variable.
`item_variables`	character vector. Names of the columns with item scores.
`age_name`	string. Name of the age variable. Default is `"age_years"`.
`step_window`	numeric vector. Window widths to evaluate.
`min_agegroup`	numeric. Minimum age to include. Defaults to the floor of the minimum age in the data.
`max_agegroup`	numeric. Maximum age to include. Defaults to the ceiling of the maximum age in the data.
`step_agegroup`	numeric vector. Step sizes between evaluated age points.
`include_window_per_person`	logical. If `TRUE`, also estimates reliability for each individual. Default is `FALSE`.
`complete.obs`	logical. If `TRUE` (default), uses listwise deletion; if `FALSE`, uses pairwise deletion.

Value

An object of class `Drel` (a data.frame) with:

- `rel`: Reliability estimates
- `age`: Corresponding evaluated ages
- `window_width`: Width of the window used
- `age_group_width`: Step size between evaluated age groups
- `version`: Type of estimation (`"step"` or `"window_per_person"`)

See Also

plot_drel()

Examples

```
invisible(data("ids_kn_data"))
rel_int <- different_rel(
  data = ids_kn_data,
  item_variables = colnames(ids_kn_data),
  age_name = "age_years",
  step_window = c(0.5, 1, 2, 5, 10, 20),
  min_agegroup = 5,
  max_agegroup = 20,
  step_agegroup = c(0.5, 1, 1.5, 2)
)
```

---

fb_select *Free order model selection procedure*

Description

fb_select() applies the free order model selection procedure, using forward–backward selection (Voncken et al. 2019). For a given GAMLSS distribution and model selection criterion, it selects the optimal polynomial degrees for all distribution parameters.

Usage

```
fb_select(
  data,
  age_name,
  score_name,
  family,
  selcrit = "BIC",
  spline = FALSE,
  method = "RS(10000)",
  max_poly = c(5, 5, 2, 2),
  min_poly = c(0, 0, 0, 0),
  start_poly = c(2, 1, 0, 0),
  trace = TRUE,
  seed = 123,
  parallel = FALSE
)
```

Arguments

`data`	data.frame. Sample on which to fit the distribution; contains the scores and ages.
`age_name`	string. Name of the age variable.
`score_name`	string. Name of the score variable.
`family`	string. For example, `"BB"`, `"BCPE"`, `"NO"`, etc. See gamlss.dist::gamlss.family for more information.
`selcrit`	string. Model selection criterion: `"AIC"`, `"BIC"` (default), `"GAIC(3)"`, or `"CV"` (cross-validation with 10 folds).
`spline`	logical. If `FALSE` (default), estimate polynomial(s) for μ; if `TRUE`, estimate a p-spline for μ.
`method`	string. Estimation method for `gamlss::gamlss()`. Either `"RS()"`, `"CG()"`, or `"mixed()"`, with iteration count. Default is `"RS(10000)"`.
`max_poly`	vector. Maximum polynomial degrees for each parameter.
`min_poly`	vector. Minimum polynomial degrees for each parameter.
`start_poly`	vector. Starting polynomial degrees for each parameter.
`trace`	logical. If `TRUE`, prints progress during selection.
`seed`	integer. Random seed for cross-validation folds.
`parallel`	logical. If `TRUE`, candidate models are evaluated in parallel using **future.apply**. This can reduce elapsed time for computationally heavy settings (e.g., large datasets, distributions with many parameters, or when using cross-validation as the selection criterion). For light models or small datasets, the overhead of parallelization may make it slower than sequential evaluation. Parallelization is not supported for user-defined distribution families; use built-in **gamlss.dist** families instead. Default is `FALSE`.

Details

If `parallel = TRUE`, candidate models are evaluated in parallel using the **future** and **future.apply** packages. If these packages are not installed, a message is printed and the function continues with sequential evaluation. Parallelization can reduce elapsed time for large datasets, complex models and cross-validation, but may be slower than sequential evaluation for smaller problems.

Value

A selected GAMLSS model with the chosen polynomial degrees and the final criterion value.

References

Voncken L, Albers CJ, Timmerman ME (2019). "Model selection in continuous test norming with GAMLSS." *Assessment*, **26**(7), 1329–1346. doi:10.1177/1073191117715113.

See Also

`shape_data()`, `fb_select()`, `normtable_create()`

Examples

```
invisible(data("ids_data"))
mydata <- shape_data(ids_data, age_name = "age", score_name = "y14", family = "BB")
mod <- fb_select(mydata, age_name = "age", score_name = "shaped_score",
                 family = "BB", selcrit = "BIC")
```

`ids_data` *ids data*

Description

The data are perturbed data, based on scores on Test 14 ("naming antonyms") and Test 7 ("naming categories") of the intelligence test IDS-2 (Grob & Hagmann-von Arx, 2018a; Grob et al., 2018b). The data are provided as supplementary material to Timmerman et al. (2021).

Usage

```
data(ids_data)
```

Format

A dataframe with three columns:

`age` age in years

`y7` raw test score on Test 7

`y14` raw test score on Test 14

Source

https://osf.io/p75a6

References

Grob A, Hagmann-von Arx P (2018). *IDS 2: Intelligence and Development Scales-2*. Hogrefe.

Grob A, Hagmann-von Arx P, Ruiter S, Timmerman M, Visser L (2018). *IDS-2: Intelligentie-en ontwikkelingsschalen voor kinderen en jongeren*. Hogrefe Publishing.

Timmerman ME, Voncken L, Albers CJ (2021). "A tutorial on regression-based norming of psychological tests with GAMLSS." *Psychological methods*, **26**(3), 357. doi:10.1037/met0000348.

ids_kn_data *The ids_kn_data are simlulated data for demonstration purposes*

Description

The data are simulated data for demonstration purposes, akin to Test 7 ("naming categories") of the intelligence test IDS-2 (Grob & Hagmann-von Arx, 2018). It consists of the binary scores on 34 items (KN_1,...,KN_34). The raw test score is the sum of the 34 item scores. The data are provided as supplementary material to Heister et al. (2024).

Usage

```
data(ids_kn_data)
```

Format

A dataframe with 36 columns:

`KN_1` binary score on item 1

`KN_2` binary score on item 2 ...

`KN_34` binary score on item 34

`rawscore` raw test score as the unweighted sum of the scores on item 1 to item 34

`age_years` age in year

Source

https://osf.io/dc5k9/files/osfstorage

References

Grob A, Hagmann-von Arx P (2018). *IDS 2: Intelligence and Development Scales-2*. Hogrefe.

Heister HM, Albers CJ, Wiberg M, Timmerman ME (2024). "Item response theory-based continuous test norming." *Psychological methods*. doi:10.1037/met0000686.

ids_rel_data *These fictional reliability data are for demonstration purposes.*

Description

Dataframe with the vectors age and rel, with the ages evaluated, and rel the (fictional) test reliability per age.

Usage

```
data(ids_rel_data)
```

Format

A dataframe with two columns:

`age` age in years

`rel` reliability

Source

constructed by authors

`normtable_create` *Create a norm table based on a GAMLSS fitted model*

Description

`normtable_create()` creates a norm table based on a fitted `GAMLSS` model.

Usage

```
normtable_create(
  model,
  data,
  age_name,
  score_name,
  datarel = NULL,
  normtype = "Z",
  min_age = NULL,
  max_age = NULL,
  min_score = NULL,
  max_score = NULL,
  step_size_score = 1,
  step_size_age = NULL,
  cont_cor = FALSE,
  ci_level = 0.95,
  trim = 3,
  excel = FALSE,
  excel_name = tempfile("norms", fileext = ".xlsx"),
  new_data = FALSE
)
```

Arguments

`model`	a GAMLSS fitted model, for example the result of `fb_select()`.
`data`	data.frame. The sample on which the model has been fitted, or new data; must contain the score variable (with name given in `score_name`) and age variable (with name given in `age_name`).

age_name	string. Name of the age variable.
score_name	string. Name of the score variable.
datarel	data.frame or numeric. If a data.frame, must contain columns age and rel, with estimated test reliability per age. If numeric, a constant reliability is assumed for all ages (optional, only needed for confidence intervals).
normtype	string. Norm score type: "Z" (N(0,1); default), "T" (N(50,10)), or "IQ" (N(100,15)).
min_age	numeric. Lowest age value in the norm table; default is the first integer below the minimum observed age.
max_age	numeric. Highest age value in the norm table; default is the first integer above the maximum observed age.
min_score	numeric. Lowest score value in the norm table; default is the minimum observed score.
max_score	numeric. Highest score value in the norm table; default is the maximum observed score.
step_size_score	numeric. Increment of the scores in the norm table; default is 1.
step_size_age	numeric. Increment of the ages in the norm table; defaults to approximately 100 ages in total.
cont_cor	logical. If TRUE, apply continuity correction for discrete test scores. Default is FALSE.
ci_level	numeric. Confidence interval level (if datarel is provided). Default is 0.95.
trim	numeric. Trim norm scores at ± trim standard deviations. Default is 3.
excel	logical. If TRUE, attempt to write results to an Excel file. Default is FALSE.
excel_name	character. Path to the Excel file. Defaults to a temporary file. Ignored if excel = FALSE.
new_data	logical. If FALSE (default), create a full norm table and norm scores. If TRUE, only return norm scores for the given data.

Details

If excel = TRUE, results are written to an Excel file via the **openxlsx2** package. If the package is not installed, a message is printed and the function continues without writing an Excel file. By default, the file is written to a temporary path (see tempfile()); if you want to keep the file permanently, provide your own file name via the excel_name argument (e.g., "norms.xlsx").

Value

A list of class NormTable containing:

- norm_sample: Estimated norm scores (normtype) in the sample, trimmed at trim.
- norm_sample_lower, norm_sample_upper: Lower and upper ci_level confidence bounds of norm_sample.
- norm_matrix: Norm scores (normtype) by age (only if new_data = FALSE).
- norm_matrix_lower, norm_matrix_upper: Lower and upper ci_level bounds of norm_matrix.

- `znorm_sample`: Estimated Z scores in the sample.
- `cdf_sample`: Estimated percentiles in the sample.
- `cdf_matrix`: Percentile table by age (only if `new_data = FALSE`).
- `data`, `age_name`, `score_name`: Copies of respective function arguments.
- `pop_age`: Evaluated ages in the norm table (only if `new_data = FALSE`).

References

Timmerman ME, Voncken L, Albers CJ (2021). "A tutorial on regression-based norming of psychological tests with GAMLSS." *Psychological methods*, **26**(3), 357. doi:10.1037/met0000348.

See Also

`fb_select()`, `plot_normtable()`

Examples

```
# Load example data
invisible(data("ids_data"))

# Prepare data for modeling
mydata_BB_y14 <- shape_data(
  data = ids_data,
  age_name = "age",
  score_name = "y14",
  family = "BB"
)

# Fit model using BIC as selection criterion
mod_BB_y14 <- fb_select(
  data = mydata_BB_y14,
  age_name = "age",
  score_name = "shaped_score",
  family = "BB",
  selcrit = "BIC"
)

# Create norm table from fitted model
norm_mod_BB_y14 <- normtable_create(
  model = mod_BB_y14,
  data = mydata_BB_y14,
  age_name = "age",
  score_name = "shaped_score"
)

# Calculate norms for a new sample using reliability data
invisible(data("ids_rel_data"))
newdata <- ids_data[1:5, c("age", "y14")]

norm_mod_BB_newdata <- normtable_create(
  model = mod_BB_y14,
```

```
  data = newdata,
  age_name = "age",
  score_name = "y14",
  new_data = TRUE,
  datarel = ids_rel_data
)
```

`plot_drel` *Plot reliability estimates over age*

Description

`plot_drel()` plots reliability estimates as a function of age, based on different window widths, using a `Drel` object.

Usage

```
plot_drel(drel, ncol = 3, nrow = 2, ...)
```

Arguments

`drel`	a `Drel` object (created with `different_rel()`).
`ncol`	number of plots per row (default: 3).
`nrow`	number of plots per column (default: 2).
`...`	additional arguments passed to plotting functions.

Value

graphical output and the `ggplot` object used to create it.

See Also

`different_rel()`

Examples

```
data("ids_kn_data")

rel_int <- different_rel(
  data           = ids_kn_data,
  item_variables = colnames(ids_kn_data),
  age_name       = "age_years",
  step_window    = c(0.5, 1, 2, 5, 10, 20),
  min_agegroup   = 5,
  max_agegroup   = 20,
  step_agegroup  = c(0.5, 1, 1.5, 2)
)
```

```
plot_drel(rel_int, ncol = 2)
```

`plot_normtable` *Plot norm curves from a NormTable object*

Description

`plot_normtable()` plots norm curves as a function of the predictor, along with the sample data, based on a `NormTable` object.

Usage

```
plot_normtable(
  normtable,
  lty = 1,
  lwd = 3,
  pch = 1,
  cex = 0.5,
  col = "aquamarine4",
  xlab = "Age",
  ylab = "Percentile",
  ...
)
```

Arguments

`normtable`	a `NormTable` object (created by `normtable_create()` with `new_data = FALSE`).
`lty`	line type(s) for curves.
`lwd`	line width(s) for curves.
`pch`	symbol for sample points.
`cex`	point size (default: `0.5`).
`col`	point colour (default: `"aquamarine4"`).
`xlab`	x-axis label (default: `"Age"`).
`ylab`	y-axis label (default: `"Percentile"`).
`...`	additional graphical parameters passed to `graphics::plot()`, `graphics::lines()`, or `graphics::points()`.

Value

graphical output and the `ggplot` object used to create it.

See Also

normtable_create()

Examples

```
data("ids_data")

mydata_BB_y14 <- shape_data(
  data       = ids_data,
  age_name   = "age",
  score_name = "y14",
  family     = "BB"
)

mod_BB_y14 <- fb_select(
  data       = mydata_BB_y14,
  age_name   = "age",
  score_name = "shaped_score",
  family     = "BB",
  selcrit    = "BIC"
)

norm_mod_BB_y14 <- normtable_create(
  model      = mod_BB_y14,
  data       = mydata_BB_y14,
  age_name   = "age",
  score_name = "shaped_score"
)

# default plot
plot_normtable(norm_mod_BB_y14)
```

---

reliability_window *Estimate test reliability by age using a sliding window*

Description

Estimates reliability across age using a sliding window approach, either at fixed age points or per individual.

Usage

```
reliability_window(
  data,
  age_name,
  item_variables,
  window_width,
```

```
  window_version = "step",
  min_agegroup = NULL,
  max_agegroup = NULL,
  step_agegroup = 1,
  complete.obs = TRUE
)
```

Arguments

`data`	data.frame containing the item scores and age variable.
`age_name`	string. Name of the age variable.
`item_variables`	numeric or character vector. Column indices or names of the item variables.
`window_width`	numeric. Width of the sliding window used to group individuals by age.
`window_version`	string. Type of windowing: • `"step"` (default): Estimate reliability at fixed age intervals. • `"window_per_person"`: Estimate reliability for each individual.
`min_agegroup`	numeric. Minimum age to include. Defaults to the floor of the minimum age in the data.
`max_agegroup`	numeric. Maximum age to include. Defaults to the ceiling of the maximum age in the data.
`step_agegroup`	numeric. Step size between evaluated ages. Used only when `window_version = "step"`.
`complete.obs`	logical. If `TRUE` (default), uses listwise deletion; if `FALSE`, uses pairwise deletion.

Value

A data.frame with:

- `rel`: Reliability estimates
- `age`: Corresponding age values
- `window_width`: The width of the sliding window
- `window_per`: Description of age step or observation unit

This output can be used as the `datarel` argument in `normtable_create()`.

References

Heister HM, Albers CJ, Wiberg M, Timmerman ME (2024). "Item response theory-based continuous test norming." *Psychological methods*. doi:10.1037/met0000686.

See Also

`normtable_create()`

Examples

```
invisible(data("ids_kn_data"))
rel_est <- reliability_window(
  data = ids_kn_data,
  age_name = "age_years",
  item_variables = colnames(ids_kn_data),
  window_width = 2
)
```

`sample_size_poly` *Sample size planning for continuous norming using polynomial regression*

Description

Computes optimal sample sizes per group under a distribution free polynomial regression model for group means, following Hessen (2026).

Usage

```
sample_size_poly(
  n_groups,
  poly_degree,
  n0,
  variances = NULL,
  solution_type = c("balanced", "all", "range")
)
```

Arguments

`n_groups`	Integer. Number of groups (e.g., age groups).
`poly_degree`	Integer scalar or vector. Polynomial degree(s) of polynomial regression model.
`n0`	Integer scalar or vector. Reference sample size per group under traditional norming. Typical values are 200, 300, or 400 (see e.g., Egberink et al. (2026)).
`variances`	Optional numeric vector of length `n_groups`. Estimated variances of the means in each group. If not provided, homoscedastic variances are assumed (`rep(1, n_groups)`).
`solution_type`	Character string indicating which solution(s) to return. Must be one of:
	"balanced" (default) Returns a single optimal solution with the most balanced allocation across groups.
	"range" Returns, for each group, the range of sample sizes across all optimal solutions.
	"all" Returns all optimal solutions.

Details

In the assumed continuous norming model, group means are first estimated and then smoothed using a polynomial model that regresses group means on age. This function determines group sample sizes such that the precision of the estimated means (not higher-order moments) is at least as high as under traditional norming.

This function implements the linear programming (LP) approach described in Hessen (2026). Multiple optimal solutions may exist: different combinations of n_{min} and n_{max} can yield the same minimal total sample size while satisfying the precision constraints.

By default (`solution_type = "balanced"`), a single solution is returned, defined as the solution minimizing the difference between the largest and smallest group sample sizes. This yields a practically balanced design. Alternatively, users can inspect all optimal solutions (`solution_type = "all"`) or examine the range of optimal sample sizes per group (`solution_type = "range"`).

How to use:

- Before data collection: do not provide `variances` (assumes homoscedasticity).
- During/after data collection: provide estimated `variances` per group.

Value

A data.frame. The structure depends on `solution_type`:

solution_type = "balanced" Returns one optimal solution with:

n0 Reference sample size per group
group Group index
poly_degree Polynomial degree
n Sample size per group
variance Expected variance of estimated mean
variance_traditional Variance under traditional norming ($\sigma_j^2/n0$)
total_n Total sample size across all groups
traditional_total_n Total sample size under traditional norming
n_min Lower bound used in LP
n_max Upper bound used in LP

solution_type = "all" Returns all optimal solutions with the same columns as above, plus:

solution_id Identifier for solution number

solution_type = "range" Returns one row per group with:

n0 Reference sample size per group
group Group index
degree Polynomial degree
n_range Range of sample sizes over optimal solutions

References

Egberink I, De Leng W, Evers A, Hemker B, Lucassen W (2026). *COTAN Beoordelingssysteem voor de Kwaliteit van Tests. Geheel herziene versie 2026*. Nederlands Instituut van Psychologen. Hessen DJ (2026). "Richtlijnen voor steekproefgroottes bij continu normeren." Utrecht University, https://github.com/djhessen/COTAN.

Examples

```
# Example 1: Planning before data collection (homoscedastic)
## Not run:
res1 <- sample_size_poly(
  n_groups = 5,
  poly_degree = 2,
  n0 = 400
)

# Example 2: Midway planning with variance estimates (heteroscedastic)
ids_data$age_group <- cut(ids_data$age, breaks = seq(6, 18, by = 1))
v <- tapply(ids_data$y7, ids_data$age_group, var, na.rm = TRUE)

res2 <- sample_size_poly(
 n_groups = length(v),
 poly_degree = c(1,2,3),
 n0 = c(300,400),
 variances = v
 )

## End(Not run)
```

---

shape_data *Shape data as input for* fb_select()

Description

shape_data() reshapes the response variable into the right format for the specified distribution and removes all cases with missing data on the score or age variable. The result is suitable for use as input to fb_select().

Usage

```
shape_data(
  data,
  age_name,
  score_name,
  family,
  max_score = NULL,
  verbose = TRUE
)
```

Arguments

data	data.frame. Sample on which to fit the distribution; contains the scores and ages.
age_name	string. Name of the age variable.
score_name	string. Name of the score variable.

`family`	string. For example, `"BB"`, `"BCPE"`, `"NO"`, etc. See gamlss.dist::gamlss.family for more information.
`max_score`	numeric. Highest possible score in the norm table. Defaults to the maximum observed score in the sample.
`verbose`	logical. If `TRUE`, messages are printed whenever a transformation is applied.

Details

The function checks whether the response values are valid for the specified GAMLSS distribution family. If not, transformations are applied to ensure compatibility. Messages are printed (if `verbose = TRUE`) to describe each transformation.

Unexpected transformations should prompt inspection of the original data. Note that the function does **not** assess whether the chosen family is appropriate for the data—it only ensures compatibility.

Compatible with all gamlss distributions, with the exception of distributions in the multinomial family (gamlss::.gamlss.multin.list). This includes user-defined distributions, such as truncated distributions.

Value

A data.frame containing the original variables and a new column `shaped_score`, with the response variable in the correct format for GAMLSS modeling.

References

Voncken L, Albers CJ, Timmerman ME (2019). "Model selection in continuous test norming with GAMLSS." *Assessment*, **26**(7), 1329–1346. doi:10.1177/1073191117715113.

See Also

`fb_select()`

Examples

```
invisible(data("ids_data"))
mydata_BB <- shape_data(ids_data, age_name = "age", score_name = "y14", family = "BB")
mydata_BCPE <- shape_data(ids_data, age_name = "age", score_name = "y14", family = "BCPE")
```

Index