Package: cNORM (via r-universe)

September 1, 2024

Type Package

Title Continuous Norming

Version 3.3.0

Maintainer Wolfgang Lenhard <`wolfgang.lenhard@uni-wuerzburg.de`>

Date 2024-08-25

Description A comprehensive toolkit for generating continuous test norms in psychometrics and biometrics, and analyzing model fit. The package offers both distribution-free modeling using Taylor polynomials and parametric modeling using the beta-binomial distribution. Originally developed for achievement tests, it is applicable to a wide range of mental, physical, or other test scores dependent on continuous or discrete explanatory variables. The package provides several advantages: It minimizes deviations from representativeness in subsamples, interpolates between discrete levels of explanatory variables, and significantly reduces the required sample size compared to conventional norming per age group. cNORM enables graphical and analytical evaluation of model fit, accommodates a wide range of scales including those with negative and descending values, and even supports conventional norming. It generates norm tables including confidence intervals. It also includes methods for addressing representativeness issues through Iterative Proportional Fitting.

Depends R (>= 4.0.0)

Imports leaps (>= 3.1), ggplot2 (>= 3.5.1)

Suggests knitr, shiny, foreign, readxl, rmarkdown, testthat

License AGPL-3

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation no

Encoding UTF-8

LazyData true

URL https://www.psychometrica.de/cNorm_en.html,
https://github.com/WLenhard/cNORM

BugReports https://github.com/WLenhard/cNORM/issues

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

RemoteUrl https://github.com/wlenhard/cnorm

RemoteRef HEAD

RemoteSha 184c4086a4f94e22f8192f67b554bb9e2a5acc2a

Contents

`bestModel` *Determine Regression Model*

Description

Computes Taylor polynomial regression models by evaluating a series of models with increasing predictors. It aims to find a consistent model that effectively captures the variance in the data. It draws on the regsubsets function from the leaps package and builds up to 20 models for each number of predictors, evaluates these models regarding model consistency and selects consistent model with the highest R^2. This automatic model selection should usually be accompanied with visual inspection of the percentile plots and assessment of fit statistics. Set R^2 or number of terms manually to retrieve a more parsimonious model, if desired.

Usage

```
bestModel(
  data,
  raw = NULL,
  R2 = NULL,
  k = NULL,
  t = NULL,
  predictors = NULL,
  terms = 0,
  weights = NULL,
  force.in = NULL,
  plot = TRUE,
  extensive = TRUE
)
```

Arguments

`data`	Preprocessed dataset with ’raw’ scores, powers, interactions, and usually an explanatory variable (like age).
`raw`	Name of the raw score variable (default: ’raw’).
`R2`	Adjusted R^2 stopping criterion for model building.
`k`	Power constant influencing model complexity (default: 4, max: 6).
`t`	Age power parameter. If unset, defaults to ‘k‘.
`predictors`	List of predictors or regression formula for model selection. Overrides ’k’ and can include additional variables.
`terms`	Desired number of terms in the model.
`weights`	Optional case weights. If set to FALSE, default weights (if any) are ignored.
`force.in`	Variables forcibly included in the regression.
`plot`	If TRUE (default), displays a percentile plot of the model and information about the regression object. FALSE turns off plotting and report.
`extensive`	If TRUE (default), screen models for consistency and - if possible, exclude inconsistent ones

Details

The functions `rankBySlidingWindow`, `rankByGroup`, `bestModel`, `computePowers` and `prepareData` are usually not called directly, but accessed through other functions like `cnorm`.

Additional functions like `plotSubset(model)` and `cnorm.cv` can aid in model evaluation.

Value

The model. Further exploration can be done using `plotSubset(model)` and `plotPercentiles(data, model)`.

See Also

plotSubset, plotPercentiles, plotPercentileSeries, checkConsistency

Other model: `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

Examples

```
# Example with sample data
## Not run:
# It is not recommende to use this function. Rather use 'cnorm' instead.
normData <- prepareData(elfe)
model <- bestModel(normData)
plotSubset(model)
plotPercentiles(buildCnormObject(normData, model))

# Specifying variables explicitly
preselectedModel <- bestModel(normData, predictors = c("L1", "L3", "L1A3", "A2", "A3"))
print(regressionFunction(preselectedModel))

## End(Not run)
```

`betaCoefficients` *Compute Parameters of a Beta Binomial Distribution*

Description

This function calculates the α (a) and β (b) parameters of a beta binomial distribution, along with the mean (m), variance (var) based on the input vector ‘x‘ and the maximum number ‘n‘.

Usage

```
betaCoefficients(x, n = NULL)
```

Arguments

`x`	A numeric vector of non-negative integers representing observed counts.
`n`	The maximum number or the maximum possible value of ‘x‘. If not specified, uses max(x) instead.

Details

The beta-binomial distribution is a discrete probability distribution that models the number of successes in a fixed number of trials, where the probability of success varies from trial to trial. This variability in success probability is modeled by a beta distribution. Such a calculation is particularly relevant in scenarios where there is heterogeneity in success probabilities across trials, which is common in real-world situations, as for example the number of correct solutions in a psychometric test, where the test has a fixed number of items.

Value

A numeric vector containing the calculated parameters in the following order: alpha (a), beta (b), mean (m), standard deviation (sd), and the maximum number (n).

Examples

```
x <- c(1, 2, 3, 4, 5)
n <- 5

betaCoefficients(x, n) # or, to set n to max(x)
betaCoefficients(x)
```

`buildCnormObject` *Build cnorm object from data and bestModel model object*

Description

Helper function to build a cnorm object from a data object and a model object from the bestModel function for compatibility reasons.

Usage

```
buildCnormObject(data, model)
```

Arguments

`data`	A data object from 'prepareData', or from 'rankByGroup' and 'computePower'
`model`	Object obtained from the bestModel function

Value

A cnorm object

Examples

```
## Not run:
  data <- prepareData(elfe)
  model <- bestModel(data, k = 4)
  model.cnorm <- buildCnormObject(data, model)

## End(Not run)
```

`buildFunction` *Build regression function for bestModel*

Description

Build regression function for bestModel

Usage

```
buildFunction(raw, k, t, age)
```

Arguments

`raw`	name of the raw score variable
`k`	the power degree for location
`t`	the power degree for age
`age`	use age

Value

regression function

`calcPolyInL` *Internal function for retrieving regression function coefficients at specific age*

Description

The function is an inline for searching zeros in the inverse regression function. It collapses the regression function at a specific age and simplifies the coefficients.

Usage

```
calcPolyInL(raw, age, model)
```

Arguments

`raw`	The raw value (subtracted from the intercept)
`age`	The age
`model`	The cNORM regression model

Value

The coefficients

`calcPolyInLBase` *Internal function for retrieving regression function coefficients at specific age*

Description

The function is an inline for searching zeros in the inverse regression function. It collapses the regression function at a specific age and simplifies the coefficients.

Usage

```
calcPolyInLBase(raw, age, coeff, k)
```

Arguments

`raw`	The raw value (subtracted from the intercept)
`age`	The age
`coeff`	The cNORM regression model coefficients
`k`	The cNORM regression model power parameter

Value

The coefficients

`calcPolyInLBase2` *Internal function for retrieving regression function coefficients at specific age (optimized)*

Description

The function is an inline for searching zeros in the inverse regression function. It collapses the regression function at a specific age and simplifies the coefficients. Optimized version of the prior ’calcPolyInLBase’

Usage

```
calcPolyInLBase2(raw, age, coeff, k)
```

Arguments

`raw`	The raw value (subtracted from the intercept)
`age`	The age
`coeff`	The cNORM regression model coefficients
`k`	The cNORM regression model power parameter

Value

The coefficients

`CDC` *BMI growth curves from age 2 to 25*

Description

By the courtesy of the Center of Disease Control (CDC), cNORM includes human growth data for children and adolescents age 2 to 25 that can be used to model trajectories of the body mass index and to estimate percentiles for clinical definitions of under- and overweight. The data stems from the NHANES surveys in the US and was published in 2012 as public domain. The data was cleaned by removing missing values and it includes the following variables from or based on the original dataset.

Usage

```
CDC
```

Format

A data frame with 45053 rows and 7 variables:

age continuous age in years, based on the month variable

group age group; chronological age in years at the time of examination

month chronological age in month at the time of examination

sex sex of the participant, 1 = male, 2 = female

height height of the participants in cm

weight weight of the participants in kg

bmi the body mass index, computed by (weight in kg)/(height in m)^2

A data frame with 45035 rows and 7 columns

Source

`https://www.cdc.gov/nchs/nhanes/index.htm`

References

CDC (2012). National Health and Nutrition Examination Survey: Questionnaires, Datasets and Related Documentation. available `https://www.cdc.gov/nchs/nhanes/index.htm` (date of retrieval: 25/08/2018)

`checkConsistency` *Check the consistency of the norm data model*

Description

While abilities increase and decline over age, within one age group, the norm scores always have to show a monotonic increase or decrease with increasing raw scores. Violations of this assumption are an indication for problems in modeling the relationship between raw and norm scores. There are several reasons, why this might occur:

1. Vertical extrapolation: Choosing extreme norm scores, e. g. values -3 <= x and x >= 3 In order to model these extreme values, a large sample dataset is necessary.
2. Horizontal extrapolation: Taylor polynomials converge in a certain radius. Using the model values outside the original dataset may lead to inconsistent results.
3. The data cannot be modeled with Taylor polynomials, or you need another power parameter (k) or R2 for the model.

Usage

```
checkConsistency(
  model,
  minAge = NULL,
  maxAge = NULL,
  minNorm = NULL,
  maxNorm = NULL,
  minRaw = NULL,
  maxRaw = NULL,
  stepAge = NULL,
  stepNorm = 1,
  warn = FALSE,
  silent = FALSE
)
```

Arguments

`model`	The model from the bestModel function or a cnorm object
`minAge`	Age to start with checking
`maxAge`	Upper end of the age check
`minNorm`	Lower end of the norm value range
`maxNorm`	Upper end of the norm value range
`minRaw`	clipping parameter for the lower bound of raw scores
`maxRaw`	clipping parameter for the upper bound of raw scores
`stepAge`	Stepping parameter for the age check. values indicate higher precision / closer checks

`stepNorm`	Stepping parameter for the norm table check within age with lower scores indicating a higher precision. The choice depends of the norm scale used. With T scores a stepping parameter of 1 is suitable
`warn`	If set to TRUE, already minor violations of the model assumptions are displayed (default = FALSE)
`silent`	turn off messages

Details

In general, extrapolation (point 1 and 2) can carefully be done to a certain degree outside the original sample, but it should in general be handled with caution. Please note that at extreme values, the models most likely become independent and it is thus recommended to restrict the norm score range to the relevant range of abilities, e.g. +/- 2.5 SD via the minNorm and maxNorm parameter.

Value

Boolean, indicating model violations (TRUE) or no problems (FALSE)

See Also

Other model: `bestModel()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

Examples

```
model <- cnorm(raw = elfe$raw, group = elfe$group, plot = FALSE)
modelViolations <- checkConsistency(model, minNorm = 25, maxNorm = 75)
plotDerivative(model, minNorm = 25, maxNorm = 75)
```

`checkWeights` *Check, if NA or values <= 0 occur and issue warning*

Description

Check, if NA or values <= 0 occur and issue warning

Usage

```
checkWeights(weights)
```

Arguments

`weights`	Raking weights

`check_monotonicity` *Check Monotonicity of Predicted Values*

Description

This function checks if the predicted values from a linear model are monotonically increasing or decreasing across a range of L values for multiple age points.

Usage

```
check_monotonicity(lm_model, pred_data, minRaw, maxRaw)
```

Arguments

`lm_model`	An object of class 'lm' representing the fitted linear model.
`pred_data`	Matrix with prediction values
`minRaw`	lowest raw score in prediction
`maxRaw`	highest raw score in prediction

Details

The function creates a prediction data frame using all combinations of the provided L values and age points. It then generates predictions using the provided linear model and checks if these predictions are monotonically increasing or decreasing for each age point across the range of L values.

Value

A named character vector where each element corresponds to an age point. Possible values for each element are 1 for "Monotonically increasing" -1 for "Monotonically decreasing", or 0 for "Not monotonic".

`cnorm` *Continuous Norming*

Description

Conducts continuous norming in one step and returns an object including ranked raw data and the continuous norming model. Please consult the function description ' of 'rankByGroup', 'rankBySlidingWindow' and 'bestModel' for specifics of the steps in the data preparation and modeling process. In addition to the raw scores, either provide

- a numeric vector for the grouping information (group)
- a numeric age vector and the width of the sliding window (age, width)

for the ranking of the raw scores. You can adjust the grade of smoothing of the regression model by setting the k and terms parameter. In general, increasing k to more than 4 and the number of terms lead to a higher fit, while lower values lead to more smoothing. The power parameter for the age trajectory can be specified independently by 't'. If both parameters are missing, cnorm uses k = 5 and t = 3 by default.

Usage

```
cnorm(
  raw = NULL,
  group = NULL,
  age = NULL,
  width = NA,
  weights = NULL,
  scale = "T",
  method = 4,
  descend = FALSE,
  k = NULL,
  t = NULL,
  terms = 0,
  R2 = NULL,
  plot = TRUE,
  extensive = TRUE
)
```

Arguments

`raw`	Numeric vector of raw scores
`group`	Numeric vector of grouping variable, e. g. grade. If no group or age variable is provided, conventional norming is applied
`age`	Numeric vector with chronological age, please additionally specify width of window
`width`	Size of the moving window in case an age vector is used
`weights`	Vector or variable name in the dataset with weights for each individual case. It can be used to compensate for moderate imbalances due to insufficient norm data stratification. Weights should be numerical and positive.
`scale`	type of norm scale, either T (default), IQ, z or percentile (= no transformation); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index points
`method`	Ranking method in case of bindings, please provide an index, choosing from the following methods: 1 = Blom (1958), 2 = Tukey (1949), 3 = Van der Warden (1952), 4 = Rankit (default), 5 = Levenbach (1953), 6 = Filliben (1975), 7 = Yu & Huang (2001)
`descend`	ranking order (default descent = FALSE): inverses the ranking order with higher raw scores getting lower norm scores; relevant for example when norming error scores, where lower scores mean higher performance

`k`	The power constant. Higher values result in more detailed approximations but have the danger of over-fit (max = 6). If not set, it uses t and if both parameters are NULL, k is set to 5.
`t`	The age power parameter (max = 6). If not set, it uses k and if both parameters are NULL, k is set to 3, since age trajectories are most often well captured by cubic polynomials.
`terms`	Selection criterion for model building. The best fitting model with this number of terms is used
`R2`	Adjusted R square as a stopping criterion for the model building (default R2 = 0.99)
`plot`	Default TRUE; plots the regression model and prints report
`extensive`	If TRUE, screen models for consistency and - if possible, exclude inconsistent ones

Value

cnorm object including the ranked raw data and the regression model

References

1. Gary, S. & Lenhard, W. (2021). In norming we trust. Diagnostica.
2. Gary, S., Lenhard, W. & Lenhard, A. (2021). Modelling Norm Scores with the cNORM Package in R. Psych, 3(3), 501-521. https://doi.org/10.3390/psych3030033
3. Lenhard, A., Lenhard, W., Suggate, S. & Segerer, R. (2016). A continuous solution to the norming problem. Assessment, Online first, 1-14. doi:10.1177/1073191116656437
4. Lenhard, A., Lenhard, W., Gary, S. (2018). Continuous Norming (cNORM). The Comprehensive R Network, Package cNORM, available: https://CRAN.R-project.org/package=cNORM
5. Lenhard, A., Lenhard, W., Gary, S. (2019). Continuous norming of psychometric tests: A simulation study of parametric and semi-parametric approaches. PLoS ONE, 14(9), e0222279. doi:10.1371/journal.pone.0222279
6. Lenhard, W., & Lenhard, A. (2020). Improvement of Norm Score Quality via Regression-Based Continuous Norming. Educational and Psychological Measurement(Online First), 1-33. https://doi.org/10.1177/0013164420928457

See Also

rankByGroup, rankBySlidingWindow, computePowers, bestModel

Examples

```
## Not run:
# Using this function with the example dataset 'elfe'

# Conventional norming (no modelling over age)
cnorm(raw=elfe$raw)

# Continuous norming
```

```
# You can use the 'getGroups()' function to set up grouping variable in case,
# you have a continuous age variable.
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)

# return norm tables including 90% confidence intervals for a
# test with a reliability of r = .85; table are set to mean of quartal
# in grade 3 (children completed 2 years of schooling)
normTable(c(2.125, 2.375, 2.625), cnorm.elfe, CI = .90, reliability = .95)

# ... or instead of raw scores for norm scores, the other way round
rawTable(c(2.125, 2.375, 2.625), cnorm.elfe, CI = .90, reliability = .95)


# Using a continuous age variable instead of distinct groups, using a sliding
# window for percentile estimation. Please specify continuos variable for age
# and the sliding window size.
cnorm.ppvt.continuous <- cnorm(raw = ppvt$raw, age = ppvt$age, width=1)


# In case of unbalanced datasets, deviating from the census, the norm data
# can be weighted by the means of raking / post stratification. Please generate
# the weights with the computeWeights() function and pass them as the weights
# parameter. For computing the weights, please specify a data.frame with the
# population margins (further information is available in the computeWeights
# function). A demonstration based on sex and migration status in vocabulary
# development (ppvt dataset):
margins <- data.frame(variables = c("sex", "sex",
                                    "migration", "migration"),
                      levels = c(1, 2, 0, 1),
                      share = c(.52, .48, .7, .3))
weights <- computeWeights(ppvt, margins)
model <- cnorm(raw = ppvt$raw, group=ppvt$group, weights = weights)

## End(Not run)
```

`cnorm.betabinomial` *Fit a beta-binomial regression model for continuous norming*

Description

This function fits a beta-binomial regression model where both the /$alpha$ and /$beta$ parameters of the beta-binomial distribution are modeled as polynomial functions of the predictor variable (typically age). Setting mode to 1 fits a beta-binomial model on the basis of /mu and /$sigma$, setting it to 2 (default) fits a beta-binomial model directly on the basis of /$alpha$ and /$beta$.

Usage

```
cnorm.betabinomial(
  age,
  score,
```

```
  n = NULL,
  weights = NULL,
  mode = 2,
  alpha = 3,
  beta = 3,
  control = NULL,
  scale = "T",
  plot = T
)
```

Arguments

`age`	A numeric vector of predictor values (e.g., age).
`score`	A numeric vector of response values.
`n`	The maximum score (number of trials in the beta-binomial distribution). If NULL, max(score) is used.
`weights`	A numeric vector of weights for each observation. Default is NULL (equal weights).
`mode`	Integer specifying the mode of the model. Default is 2 (direct modelling of /\$alpha\$ and /\$beta\$). If set to 1, the model is fitted on the basis of /\$mu\$ and /\$sigma\$, the predicted mean and standard deviation over age.
`alpha`	Integer specifying the degree of the polynomial for the alpha model. Default is 3. If mode is set to 1, this parameter is used to specify the degree of the polynomial for the /\$mu\$ model.
`beta`	Integer specifying the degree of the polynomial for the beta model. Default is 3. If mode is set to 1, this parameter is used to specify the degree of the polynomial for the /\$sigma\$ model.
`control`	A list of control parameters to be passed to the ‘optim‘ function. If NULL, default values are used, namely control = list(reltol = 1e-8, maxit = 1000) for mode 1 and control = list(factr = 1e-8, maxit = 1000) for mode 2. and
`scale`	Type of norm scale, either "T" (default), "IQ", "z" or a double vector with the mean and standard deviation.
`plot`	Logical indicating whether to plot the model. Default is TRUE.

Details

The function standardizes the input variables, fits polynomial models for both the alpha and beta parameters, and uses maximum likelihood estimation to find the optimal parameters. The optimization is performed using the L-BFGS-B method.

Value

A list of class "cnormBetaBinomial" or "cnormBetaBinomial2". In case of mode 2 containing:

`alpha_est`	Estimated coefficients for the alpha model
`beta_est`	Estimated coefficients for the beta model

`se`	Standard errors of the estimated coefficients
`alpha_degree`	Degree of the polynomial for the alpha model
`beta_degree`	Degree of the polynomial for the beta model
`result`	Full result from the optimization procedure

Examples

```
## Not run:
# Fit a beta-binomial regression model to the PPVT data
model <- cnorm.betabinomial(ppvt$age, ppvt$raw, n = 228)
summary(model)

# Use weights for post-stratification
marginals <- data.frame(var = c("sex", "sex", "migration", "migration"),
                        level = c(1,2,0,1),
                        prop = c(0.51, 0.49, 0.65, 0.35))
weights <- computeWeights(ppvt, marginals)
model <- cnorm.betabinomial(ppvt$age, ppvt$raw, n = 228, weights = weights)

## End(Not run)
```

`cnorm.cv` *Cross-validation for Term Selection in cNORM*

Description

Assists in determining the optimal number of terms for the regression model using repeated Monte Carlo cross-validation. It leverages an 80-20 split between training and validation data, with stratification by norm group or random sample in case of using sliding window ranking.

Usage

```
cnorm.cv(
  data,
  formula = NULL,
  repetitions = 5,
  norms = TRUE,
  min = 1,
  max = 12,
  cv = "full",
  pCutoff = NULL,
  width = NA,
  raw = NULL,
  group = NULL,
  age = NULL,
  weights = NULL
)
```

Arguments

`data`	Data frame of norm sample or a cnorm object. Should have ranking, powers, and interaction of L and A.
`formula`	Formula from an existing regression model; min/max functions ignored. If using a cnorm object, this is automatically fetched.
`repetitions`	Number of repetitions for cross-validation.
`norms`	If TRUE, computes norm score crossfit and R^2. Note: Computationally intensive.
`min`	Start with a minimum number of terms (default = 1).
`max`	Maximum terms in model, up to (k + 1) * (t + 1) - 1.
`cv`	"full" (default) splits data into training/validation, then ranks. Otherwise, expects a pre-ranked dataset.
`pCutoff`	Checks stratification for unbalanced data. Performs a t-test per group. Default set to 0.2 to minimize beta error.
`width`	If provided, ranking done via ‘rankBySlidingWindow‘. Otherwise, by group.
`raw`	Name of the raw score variable.
`group`	Name of the grouping variable.
`age`	Name of the age variable.
`weights`	Name of the weighting parameter.

Details

Successive models, with an increasing number of terms, are evaluated, and the RMSE for raw scores plotted. This encompasses the training, validation, and entire dataset. If ‘norms‘ is set to TRUE (default), the function will also calculate the mean norm score reliability and crossfit measures. Note that due to the computational requirements of norm score calculations, execution can be slow, especially with numerous repetitions or terms.

When ‘cv‘ is set to "full" (default), both test and validation datasets are ranked separately, providing comprehensive cross-validation. For a more streamlined validation process focused only on modeling, a pre-ranked dataset can be used. The output comprises RMSE for raw score models, norm score R^2, delta R^2, crossfit, and the norm score SE according to Oosterhuis, van der Ark, & Sijtsma (2016).

This function is not yet prepared for the ’extensive’ search strategy, introduced in version 3.3, but instead relies on the first model per number of terms, without consistency check.

For assessing overfitting:

$$CROSSFIT = R(Training; Model)^2 / R(Validation; Model)^2$$

A CROSSFIT > 1 suggests overfitting, < 1 suggests potential underfitting, and values around 1 are optimal, given a low raw score RMSE and high norm score validation R^2.

Suggestions for ideal model selection:

- Visual inspection of percentiles with ‘plotPercentiles‘ or ‘plotPercentileSeries‘.
- Pair visual inspection with repeated cross-validation (e.g., 10 repetitions).
- Aim for low raw score RMSE and high norm score R^2, avoiding terms with significant overfit (e.g., crossfit > 1.1).

Value

Table with results per term number: RMSE for raw scores, R^2 for norm scores, and crossfit measure.

References

Oosterhuis, H. E. M., van der Ark, L. A., & Sijtsma, K. (2016). Sample Size Requirements for Traditional and Regression-Based Norms. Assessment, 23(2), 191–202. https://doi.org/10.1177/1073191115580638

See Also

Other model: `bestModel()`, `checkConsistency()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

Examples

```
## Not run:
# Example: Plot cross-validation RMSE by number of terms (up to 9) with three repetitions.
result <- cnorm(raw = elfe$raw, group = elfe$group)
cnorm.cv(result$data, min = 2, max = 9, repetitions = 3)

# Using a cnorm object examines the predefined formula.
cnorm.cv(result, repetitions = 1)

## End(Not run)
```

`cNORM.GUI` *Launcher for the graphical user interface of cNORM*

Description

Launcher for the graphical user interface of cNORM

Usage

```
cNORM.GUI(launch.browser = TRUE)
```

Arguments

`launch.browser`	Default TRUE; automatically open browser for GUI

Examples

```
## Not run:
# Launch graphical user interface
cNORM.GUI()

## End(Not run)
```

`computePowers` *Compute powers of the explanatory variable a as well as of the person location l (data preparation)*

Description

The function computes powers of the norm variable e. g. T scores (location, L), an explanatory variable, e. g. age or grade of a data frame (age, A) and the interactions of both (L X A). The k variable indicates the degree up to which powers and interactions are build. These predictors can be used later on in the `bestModel` function to model the norm sample. Higher values of k allow for modeling the norm sample closer, but might lead to over-fit. In general k = 3 or k = 4 (default) is sufficient to model human performance data. For example, k = 2 results in the variables L1, L2, A1, A2, and their interactions L1A1, L2A1, L1A2 and L2A2 (but k = 2 is usually not sufficient for the modeling). Please note, that you do not need to use a normal rank transformed scale like T r IQ, but you can as well use the percentiles for the 'normValue' as well.

Usage

```
computePowers(data, k = 5, norm = NULL, age = NULL, t = 3, silent = FALSE)
```

Arguments

`data`	data.frame with the norm data
`k`	degree
`norm`	the variable containing the norm data in the data.frame; might be T scores, IQ scores, percentiles ...
`age`	Explanatory variable like age or grade, which was as well used for the grouping. Can be either the grouping variable itself or a finer grained variable like the exact age. Other explanatory variables can be used here instead an age variable as well, as long as the variable is at least ordered metric, e. g. language or development levels ... The label 'age' is used, as this is the most common field of application.
`t`	the age power parameter (default NULL). If not set, cNORM automatically uses k. The age power parameter can be used to specify the k to produce rectangular matrices and specify the course of scores per independently from k
`silent`	set to TRUE to suppress messages

Details

The functions `rankBySlidingWindow`, `rankByGroup`, `bestModel`, `computePowers` and `prepareData` are usually not called directly, but accessed through other functions like `cnorm`.

Value

data.frame with the powers and interactions of location and explanatory variable / age

See Also

bestModel

Other prepare: `prepareData()`, `rankByGroup()`, `rankBySlidingWindow()`

Examples

```
# Dataset with grade levels as grouping
data.elfe <- rankByGroup(elfe)
data.elfe <- computePowers(data.elfe)

# Dataset with continuous age variable and k = 5
data.ppvt <- rankByGroup(ppvt)
data.ppvt <- computePowers(data.ppvt, age = "age", k = 5)
```

`computeWeights` *Weighting of cases through iterative proportional fitting (Raking)*

Description

Computes and standardizes weights via raking to compensate for non-stratified samples. It is based on the implementation in the survey R package. It reduces data collection #' biases in the norm data by the means of post stratification, thus reducing the effect of unbalanced data in percentile estimation and norm data modeling.

Usage

```
computeWeights(data, population.margins, standardized = TRUE)
```

Arguments

`data`	data.frame with norm sample data.
`population.margins`	A data.frame including three columns, specifying the variable name in the original dataset used for data stratification, the factor level of the variable and the according population share. Please ensure, the original data does not include factor levels, not present in the population.margins. Additionally, summing up the shares of the different levels of a variable should result in a value near 1.0. The first column must specify the name of the stratification variable, the second the level and the third the proportion
`standardized`	If TRUE (default), the raking weights are scaled to weights/min(weights)

Details

This function computes standardized raking weights to overcome biases in norm samples. It generates weights, by drawing on the information of population shares (e. g. for sex, ethnic group, region ...) and subsequently reduces the influence of over-represented groups or increases under-represented cases. The returned weights are either raw or standardized and scaled to be larger than 0.

Raking in general has a number of advantages over post stratification and it additionally allows cNORM to draw on larger datasets, since less cases have to be removed during stratification. To use this function, additionally to the data, a data frame with stratification variables has to be specified. The data frame should include a row with (a) the variable name, (b) the level of the variable and (c) the according population proportion.

Value

a vector with the standardized weights

Examples

```
# cNORM features a dataset on vocabulary development (ppvt)
# that includes variables like sex or migration. In order
# to weight the data, we have to specify the population shares.
# According to census, the population includes 52% boys
# (factor level 1 in the ppvt dataset) and 70% / 30% of persons
# without / with a a history of migration (= 0 / 1 in the dataset).
# First we set up the popolation margins with all shares of the
# different levels:

margins <- data.frame(variables = c("sex", "sex",
                                    "migration", "migration"),
                      levels = c(1, 2, 0, 1),
                      share = c(.52, .48, .7, .3))
head(margins)

# Now we use the population margins to generate weights
# through raking

weights <- computeWeights(ppvt, margins)


# There are as many different weights as combinations of
# factor levels, thus only four in this specific case

unique(weights)


# To include the weights in the cNORM modelling, we have
# to pass them as weights. They are then used to set up
# weighted quantiles and as weights in the regession.

model <- cnorm(raw = ppvt$raw,
               group=ppvt$group,
```

```
                 weights = weights)
```

`derivationTable` *Create a table based on first order derivative of the regression model for specific age*

Description

In order to check model assumptions, a table of the first order derivative of the model coefficients is created.

Usage

```
derivationTable(A, model, minNorm = NULL, maxNorm = NULL, step = 0.1)
```

Arguments

`A`	the age
`model`	The regression model or a cnorm object
`minNorm`	The lower bound of the norm value range
`maxNorm`	The upper bound of the norm value range
`step`	Stepping parameter with lower values indicating higher precision

Value

data.frame with norm scores and the predicted scores based on the derived regression function

See Also

plotDerivative, derive

Other predict: `getNormCurve()`, `normTable()`, `predict.cnormBetaBinomial()`, `predict.cnormBetaBinomial2()`, `predictNorm()`, `predictRaw()`, `rawTable()`

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)

# retrieve function for time point 6
d <- derivationTable(6, cnorm.elfe, step = 0.5)
```

`derive` *Derivative of regression model*

Description

Calculates the derivative of the location / norm value from the regression model with the first derivative as the default. This is useful for finding violations of model assumptions and problematic distribution features as f. e. bottom and ceiling effects, non-progressive norm scores within an age group or in general #' intersecting percentile curves.

Usage

```
derive(model, order = 1)
```

Arguments

`model`	The regression model or a cnorm object
`order`	The degree of the derivate, default: 1

Value

The derived coefficients

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

Examples

```
m <- cnorm(raw = elfe$raw, group = elfe$group)
derivedCoefficients <- derive(m)
```

`diagnostics.betabinomial`
Diagnostic Information for Beta-Binomial Model

Description

This function provides diagnostic information for a fitted beta-binomial model from the cnorm.betabinomial function. It returns various metrics related to model convergence, fit, and complexity. In case, age and raw scores are provided, the function as well computes R2, rmse and bias for the norm scores based on the manifest and predicted norm scores.

Usage

```
diagnostics.betabinomial(model, age = NULL, score = NULL, weights = NULL)
```

Arguments

`model`	An object of class "cnormBetaBinomial", typically the result of a call to cnorm.betabinomial().
`age`	An optional vector with age values
`score`	An optional vector with raw values
`weights`	An optional vector with weights

Details

The AIC and BIC are calculated as: AIC = 2k - 2ln(L) BIC = ln(n)k - 2ln(L) where k is the number of parameters, L is the maximum likelihood, and n is the number of observations.

Value

A list containing the following diagnostic information:

- converged: Logical indicating whether the optimization algorithm converged.
- n_evaluations: Number of function evaluations performed during optimization.
- n_gradient: Number of gradient evaluations performed during optimization.
- final_value: Final value of the objective function (negative log-likelihood).
- message: Any message returned by the optimization algorithm.
- AIC: Akaike Information Criterion.
- BIC: Bayesian Information Criterion.
- max_gradient: Maximum absolute gradient at the solution (if available).

Examples

```
## Not run:
# Fit a beta-binomial model
model <- cnorm.betabinomial(ppvt$age, ppvt$raw)

# Get diagnostic information
diag_info <- diagnostics.betabinomial(model)

# Print the diagnostic information
print(diag_info)

# Summary the diagnostic information
summary(diag_info)

# Check if the model converged
if(diag_info$converged) {
  cat("Model converged successfully.\n")
} else {
  cat("Warning: Model did not converge.\n")
```

```
}

# Compare AIC and BIC
cat("AIC:", diag_info$AIC, "\n")
cat("BIC:", diag_info$BIC, "\n")

## End(Not run)
```

`elfe` *Sentence completion test from ELFE 1-6*

Description

A dataset containing the raw data of 1400 students from grade 2 to 5 in the sentence comprehension test from ELFE 1-6 (Lenhard & Schneider, 2006). In this test, students are presented lists of sentences with one gap. The student has to fill in the correct solution by selecting from a list of 5 alternatives per sentence. The alternatives include verbs, adjectives, nouns, pronouns and conjunctives. Each item stems from the same word type. The text is speeded, with a time cutoff of 180 seconds. The variables are as follows:

Usage

```
elfe
```

Format

A data frame with 1400 rows and 3 variables:

personID ID of the student

group grade level, with x.5 indicating the end of the school year and x.0 indicating the middle of the school year

raw the raw score of the student, spanning values from 0 to 28

A data frame with 1400 rows and 3 columns

Source

https://www.psychometrica.de/elfe2.html

References

Lenhard, W. & Schneider, W.(2006). Ein Leseverstaendnistest fuer Erst- bis Sechstklaesser. Goettingen/Germany: Hogrefe.

Examples

```
# prepare data, retrieve model and plot percentiles
model <- cnorm(elfe$group, elfe$raw)
```

`getGroups` *Determine groups and group means*

Description

Helps to split the continuous explanatory variable into groups and assigns the group mean. The groups can be split either into groups of equal size (default) or equal number of observations.

Usage

```
getGroups(x, n = NULL, equidistant = FALSE)
```

Arguments

`x`	The continuous variable to be split
`n`	The number of groups; if NULL then the function determines a number of groups with usually 100 cases or 3 <= n <= 20.
`equidistant`	If set to TRUE, builds equidistant interval, otherwise (default) with equal number of observations

Value

vector with group means for each observation

Examples

```
x <- rnorm(1000, m = 50, sd = 10)
m <- getGroups(x, n = 10)
```

`getNormCurve` *Computes the curve for a specific T value*

Description

As with this continuous norming regression approach, raw scores are modeled as a function of age and norm score (location), getNormCurve is a straightforward approach to show the raw score development over age, while keeping the norm value constant. This way, e. g. academic performance or intelligence development of a specific ability is shown.

Usage

```
getNormCurve(
  norm,
  model,
  minAge = NULL,
  maxAge = NULL,
  step = 0.1,
  minRaw = NULL,
  maxRaw = NULL
)
```

Arguments

`norm`	The specific norm score, e. g. T value
`model`	The model from the regression modeling obtained with the cnorm function
`minAge`	Age to start from
`maxAge`	Age to stop at
`step`	Stepping parameter for the precision when retrieving of the values, lower values indicate higher precision (default 0.1).
`minRaw`	lower bound of the range of raw scores (default = 0)
`maxRaw`	upper bound of raw scores

Value

data.frame of the variables raw, age and norm

See Also

Other predict: `derivationTable()`, `normTable()`, `predict.cnormBetaBinomial()`, `predict.cnormBetaBinomial2()`, `predictNorm()`, `predictRaw()`, `rawTable()`

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)
getNormCurve(35, cnorm.elfe)
```

`getNormScoreSE` *Calculates the standard error (SE) or root mean square error (RMSE) of the norm scores In case of large datasets, both results should be almost identical*

Description

Calculates the standard error (SE) or root mean square error (RMSE) of the norm scores In case of large datasets, both results should be almost identical

Usage

```
getNormScoreSE(model, type = 2)
```

Arguments

`model`	a cnorm object
`type`	either '1' for the standard error senso Oosterhuis et al. (2016) or '2' for the RMSE (default)

Value

The standard error (SE) of the norm scores sensu Oosterhuis et al. (2016) or the RMSE

References

Oosterhuis, H. E. M., van der Ark, L. A., & Sijtsma, K. (2016). Sample Size Requirements for Traditional and Regression-Based Norms. Assessment, 23(2), 191–202. https://doi.org/10.1177/1073191115580638

`modelSummary` *Prints the results and regression function of a cnorm model*

Description

Prints the results and regression function of a cnorm model

Usage

```
modelSummary(object, ...)
```

Arguments

`object`	A regression model or cnorm object
`...`	additional parameters

Value

A report on the regression function, weights, R2 and RMSE

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

`normTable` *Create a norm table based on model for specific age*

Description

This function generates a norm table for a specific age based on the regression model by assigning raw scores to norm scores. Please specify the range of norm scores, you want to cover. A T value of 25 corresponds to a percentile of .6. As a consequence, specifying a range of T = 25 to T = 75 would cover 98.4 the population. Please be careful when extrapolating vertically (at the lower and upper end of the age specific distribution). Depending on the size of your standardization sample, extreme values with T < 20 or T > 80 might lead to inconsistent results. In case a confidence coefficient (CI, default .9) and the reliability is specified, confidence intervals are computed for the true score estimates, including a correction for regression to the mean (Eid & Schmidt, 2012, p. 272).

Usage

```
normTable(
  A,
  model,
  minNorm = NULL,
  maxNorm = NULL,
  minRaw = NULL,
  maxRaw = NULL,
  step = NULL,
  monotonuous = TRUE,
  CI = 0.9,
  reliability = NULL,
  pretty = T
)
```

Arguments

`A`	the age as single value or a vector of age values
`model`	The regression model from the cnorm function
`minNorm`	The lower bound of the norm score range
`maxNorm`	The upper bound of the norm score range
`minRaw`	clipping parameter for the lower bound of raw scores
`maxRaw`	clipping parameter for the upper bound of raw scores
`step`	Stepping parameter with lower values indicating higher precision
`monotonuous`	corrects for decreasing norm scores in case of model inconsistencies (default)
`CI`	confidence coefficient, ranging from 0 to 1, default .9
`reliability`	coefficient, ranging between 0 to 1
`pretty`	Format table by collapsing intervals and rounding to meaningful precision

Value

either data.frame with norm scores, predicted raw scores and percentiles in case of simple A value or a list #' of norm tables if vector of A values was provided

References

Eid, M. & Schmidt, K. (2012). Testtheorie und Testkonstruktion. Hogrefe.

See Also

rawTable

Other predict: `derivationTable()`, `getNormCurve()`, `predict.cnormBetaBinomial()`, `predict.cnormBetaBinomial2(`
`predictNorm()`, `predictRaw()`, `rawTable()`

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)

# create single norm table
norms <- normTable(3.5, cnorm.elfe, minNorm = 25, maxNorm = 75, step = 0.5)

# create list of norm tables
norms <- normTable(c(2.5, 3.5, 4.5), cnorm.elfe,
  minNorm = 25, maxNorm = 75,
  step = 1, minRaw = 0, maxRaw = 26
)

# conventional norming, set age to arbitrary value
model <- cnorm(raw=elfe$raw)
normTable(0, model)
```

---

normTable.betabinomial

Calculate Cumulative Probabilities, Density, Percentiles, and Z-Scores for Beta-Binomial Distribution

Description

This function generates a norm table for a specific ages based on the beta binomial regression model. In case a confidence coefficient (CI, default .9) and the reliability is specified, confidence intervals are computed for the true score estimates, including a correction for regression to the mean (Eid & Schmidt, 2012, p. 272).

Usage

```
normTable.betabinomial(
  model,
  ages,
  n = NULL,
  m = NULL,
  range = 3,
  CI = 0.9,
  reliability = NULL
)
```

Arguments

`model`	The model, which was fitted using the ‘optimized.model‘ function.
`ages`	A numeric vector of age points at which to make predictions.
`n`	The number of items resp. the maximum score.
`m`	An optional stop criterion in table generation. Positive integer lower than n.
`range`	The range of the norm scores in standard deviations. Default is 3. Thus, scores in the range of +/- 3 standard deviations are considered.
`CI`	confidence coefficient, ranging from 0 to 1, default .9
`reliability`	coefficient, ranging between 0 to 1

Value

A list of data frames with columns: x, Px, Pcum, Percentile, z, norm score and possibly confidence interval

---

`plot.cnorm` *S3 function for plotting cnorm objects*

Description

S3 function for plotting cnorm objects

Usage

```
## S3 method for class 'cnorm'
plot(x, y, ...)
```

Arguments

`x`	the cnorm object
`y`	the type of plot as a string, can be one of ’raw’ (1), ’norm’ (2), ’curves’ (3), ’percentiles’ (4), ’series’ (5), ’subset’ (6), or ’derivative’ (7), either as a string or the according index
`...`	additional parameters for the specific plotting function

See Also

Other plot: `plot.cnormBetaBinomial()`, `plot.cnormBetaBinomial2()`, `plotDensity()`, `plotDerivative()`, `plotNorm()`, `plotNormCurves()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

`plot.cnormBetaBinomial` *Plot cnormBetaBinomial Model with Data and Percentile Lines*

Description

This function creates a visualization of a fitted cnormBetaBinomial model, including the original data points manifest percentiles and specified percentile lines.

Usage

```
## S3 method for class 'cnormBetaBinomial'
plot(x, ...)
```

Arguments

`x`	A fitted model object of class "cnormBetaBinomial" or "cnormBetaBinomial2".
`...`	Additional arguments passed to the plot method.

- age A vector the age data.
- A vector of the score data.
- weights An optional numeric vector of weights for each observation.
- percentiles An optional vector with the percentiles to plot.
- points Logical indicating whether to plot the data points. Default is TRUE.

Value

A ggplot object.

See Also

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial2()`, `plotDensity()`, `plotDerivative()`, `plotNorm()`, `plotNormCurves()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

Examples

```
## Not run:
# Computing beta binomial models already displays plot
model.bb <- cnorm.betabinomial(elfe$group, elfe$raw)

# Without data points
plot(model.bb, age = elfe$group, score = elfe$raw, weights=NULL, points=FALSE)


## End(Not run)
```

`plot.cnormBetaBinomial2`

Plot cnormBetaBinomial Model with Data and Percentile Lines

Description

This function creates a visualization of a fitted cnormBetaBinomial model, including the original data points manifest percentiles and specified percentile lines.

Usage

```
## S3 method for class 'cnormBetaBinomial2'
plot(x, ...)
```

Arguments

`x`	A fitted model object of class "cnormBetaBinomial" or "cnormBetaBinomial2".
`...`	Additional arguments passed to the plot method. • age A vector the age data. • A vector of the score data. • weights An optional numeric vector of weights for each observation. • percentiles An optional vector with the percentiles to plot. • points Logical indicating whether to plot the data points. Default is TRUE.

Value

A ggplot object.

See Also

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial()`, `plotDensity()`, `plotDerivative()`, `plotNorm()`, `plotNormCurves()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

`plotCnorm` *General convenience plotting function*

Description

General convenience plotting function

Usage

```
plotCnorm(x, y, ...)
```

Arguments

`x`	a cnorm object
`y`	the type of plot as a string, can be one of 'raw' (1), 'norm' (2), 'curves' (3), 'percentiles' (4), 'series' (5), 'subset' (6), or 'derivative' (7), either as a string or the according index
`...`	additional parameters for the specific plotting function

`plotDensity` *Plot the density function per group by raw score*

Description

This function plots density curves based on the regression model against the raw scores. It supports both traditional continuous norming models and beta-binomial models. The function allows for customization of the plot range and groups to be displayed.

Usage

```
plotDensity(
  model,
  minRaw = NULL,
  maxRaw = NULL,
  minNorm = NULL,
  maxNorm = NULL,
  group = NULL
)
```

Arguments

`model`	The model from the bestModel function, a cnorm object, or a cnormBetaBinomial or cnormBetaBinomial2 object.
`minRaw`	Lower bound of the raw score. If NULL, it's automatically determined based on the model type.
`maxRaw`	Upper bound of the raw score. If NULL, it's automatically determined based on the model type.
`minNorm`	Lower bound of the norm score. If NULL, it's automatically determined based on the model type.
`maxNorm`	Upper bound of the norm score. If NULL, it's automatically determined based on the model type.
`group`	Numeric vector specifying the age groups to plot. If NULL, groups are automatically selected.

Details

The function generates density curves for specified age groups, allowing for easy comparison of score distributions across different ages.

For beta-binomial models, the density is based on the probability mass function, while for traditional models, it uses a normal distribution based on the norm scores.

Value

A ggplot object representing the density functions.

Note

Please check for inconsistent curves, especially those showing implausible shapes such as violations of biuniqueness in the cnorm models.

See Also

`plotNormCurves`, `plotPercentiles`

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial()`, `plot.cnormBetaBinomial2()`, `plotDerivative()`, `plotNorm()`, `plotNormCurves()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

Examples

```
## Not run:
# For traditional continuous norming model
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotDensity(result, group = c(2, 4, 6))

# For beta-binomial model
bb_model <- cnorm.betabinomial(age = ppvt$age, score = ppvt$raw, n = 228)
plotDensity(bb_model)
```

```
## End(Not run)
```

`plotDerivative` *Plot first order derivative of regression model*

Description

This function plots the scores obtained via the first order derivative of the regression model in dependence of the norm score.

Usage

```
plotDerivative(
  model,
  minAge = NULL,
  maxAge = NULL,
  minNorm = NULL,
  maxNorm = NULL,
  stepAge = NULL,
  stepNorm = NULL,
  order = 1
)
```

Arguments

`model`	The model from the bestModel function, a cnorm object.
`minAge`	Minimum age to start checking. If NULL, it's automatically determined from the model.
`maxAge`	Maximum age for checking. If NULL, it's automatically determined from the model.
`minNorm`	Lower end of the norm score range. If NULL, it's automatically determined from the model.
`maxNorm`	Upper end of the norm score range. If NULL, it's automatically determined from the model.
`stepAge`	Stepping parameter for the age check, usually 1 or 0.1; lower values indicate higher precision.
`stepNorm`	Stepping parameter for norm scores.
`order`	Degree of the derivative (default = 1).

Details

The results indicate the progression of the norm scores within each age group. The regression-based modeling approach relies on the assumption of a linear progression of the norm scores. Negative scores in the first order derivative indicate a violation of this assumption. Scores near zero are typical for bottom and ceiling effects in the raw data.

The regression models usually converge within the range of the original values. In case of vertical and horizontal extrapolation, with increasing distance to the original data, the risk of assumption violation increases as well.

Value

A ggplot object representing the derivative of the regression function.

Note

This function is currently incompatible with reversed raw score scales ('descent' option).

See Also

`checkConsistency`, `bestModel`, `derive`

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial()`, `plot.cnormBetaBinomial2()`, `plotDensity()`, `plotNorm()`, `plotNormCurves()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

Examples

```
# For traditional continuous norming model
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotDerivative(result, minAge=2, maxAge=5, stepAge=.2, minNorm=25, maxNorm=75, stepNorm=1)
```

`plotNorm` *Plot manifest and fitted norm scores*

Description

This function plots the manifest norm score against the fitted norm score from the inverse regression model per group. This helps to inspect the precision of the modeling process. The scores should not deviate too far from the regression line. Applicable for Taylor polynomial models.

Usage

```
plotNorm(
  model,
  age = NULL,
  score = NULL,
  width = NULL,
  weights = NULL,
  group = FALSE,
  minNorm = NULL,
  maxNorm = NULL,
  type = 0
)
```

Arguments

`model`	The regression model, usually from the 'cnorm' or 'cnorm.betabinomial' function
`age`	In case of beta binomial model, please provide the age vector
`score`	In case of beta binomial model, please provide the score vector
`width`	In case of beta binomial model, please provide the width for the sliding window. If null, the function tries to determine a sensible setting.
`weights`	Vector or variable name in the dataset with weights for each individual case. If NULL, no weights are used.
`group`	On optional grouping variable, use empty string for no group, the variable name for Taylor polynomial models or a vector with the groups for beta binomial models
`minNorm`	lower bound of fitted norm scores
`maxNorm`	upper bound of fitted norm scores
`type`	Type of display: 0 = plot manifest against fitted values, 1 = plot manifest against difference values

Value

A ggplot object representing the norm scores plot.

See Also

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial()`, `plot.cnormBetaBinomial2()`, `plotDensity()`, `plotDerivative()`, `plotNormCurves()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

Examples

```
## Not run:
# Load example data set, compute model and plot results

# Taylor polynomial model
```

```
model <- cnorm(raw = elfe$raw, group = elfe$group)
plot(model, "norm")

# Beta binomial models; maximum number of items in elfe is n = 28
model.bb <- cnorm.betabinomial(elfe$group, elfe$raw, n = 28)
plotNorm(model.bb, age = elfe$group, score = elfe$raw)

## End(Not run)
```

`plotNormCurves` *Plot norm curves*

Description

This function plots the norm curves based on the regression model. It supports both Taylor polynomial models and beta-binomial models.

Usage

```
plotNormCurves(
  model,
  normList = NULL,
  minAge = NULL,
  maxAge = NULL,
  step = 0.1,
  minRaw = NULL,
  maxRaw = NULL
)
```

Arguments

`model`	The model from the bestModel function, a cnorm object, or a cnormBetaBinomial / cnormBetaBinomial2 object.
`normList`	Vector with norm scores to display. If NULL, default values are used.
`minAge`	Age to start with checking. If NULL, it's automatically determined from the model.
`maxAge`	Upper end of the age check. If NULL, it's automatically determined from the model.
`step`	Stepping parameter for the age check, usually 1 or 0.1; lower scores indicate higher precision.
`minRaw`	Lower end of the raw score range, used for clipping implausible results. If NULL, it's automatically determined from the model.
`maxRaw`	Upper end of the raw score range, used for clipping implausible results. If NULL, it's automatically determined from the model.

Details

Please check the function for inconsistent curves: The different curves should not intersect. Violations of this assumption are a strong indication of violations of model assumptions in modeling the relationship between raw and norm scores.

Common reasons for inconsistencies include: 1. Vertical extrapolation: Choosing extreme norm scores (e.g., scores <= -3 or >= 3). 2. Horizontal extrapolation: Using the model scores outside the original dataset. 3. The data cannot be modeled with the current approach, or you need another power parameter (k) or R2 for the model.

Value

A ggplot object representing the norm curves.

See Also

`checkConsistency`, `plotDerivative`, `plotPercentiles`

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial()`, `plot.cnormBetaBinomial2()`, `plotDensity()`, `plotDerivative()`, `plotNorm()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

Examples

```
## Not run:
# For Taylor continuous norming model
m <- cnorm(raw = ppvt$raw, group = ppvt$group)
plotNormCurves(m, minAge=2, maxAge=5)

# For beta-binomial model
bb_model <- cnorm.betabinomial(age = ppvt$age, score = ppvt$raw, n = 228)
plotNormCurves(bb_model)

## End(Not run)
```

`plotPercentiles` *Plot norm curves against actual percentiles*

Description

The function plots the norm curves based on the regression model against the actual percentiles from the raw data. As in 'plotNormCurves', please check for inconsistent curves, especially intersections. Violations of this assumption are a strong indication for problems in modeling the relationship between raw and norm scores. In general, extrapolation (point 1 and 2) can carefully be done to a certain degree outside the original sample, but it should in general be handled with caution. The original percentiles are displayed as distinct points in the according color, the model based projection of percentiles are drawn as lines. Please note, that the estimation of the percentiles of the raw data is done with the quantile function with the default settings. In case, you get 'jagged' or disorganized percentile curve, try to reduce the 'k' and/or 't' parameter in modeling.

Usage

```
plotPercentiles(
  model,
  minRaw = NULL,
  maxRaw = NULL,
  minAge = NULL,
  maxAge = NULL,
  raw = NULL,
  group = NULL,
  percentiles = c(0.025, 0.1, 0.25, 0.5, 0.75, 0.9, 0.975),
  scale = NULL,
  title = NULL,
  subtitle = NULL,
  points = F
)
```

Arguments

`model`	The Taylor polynomial regression model object from the cNORM
`minRaw`	Lower bound of the raw score (default = 0)
`maxRaw`	Upper bound of the raw score
`minAge`	Variable to restrict the lower bound of the plot to a specific age
`maxAge`	Variable to restrict the upper bound of the plot to a specific age
`raw`	The name of the raw variable
`group`	The name of the grouping variable; the distinct groups are automatically determined
`percentiles`	Vector with percentile scores, ranging from 0 to 1 (exclusive)
`scale`	The norm scale, either 'T', 'IQ', 'z', 'percentile' or self defined with a double vector with the mean and standard deviation, f. e. c(10, 3) for Wechsler scale index points; if NULL, scale information from the data preparation is used (default)
`title`	custom title for plot
`subtitle`	custom title for plot
`points`	Logical indicating whether to plot the data points. Default is TRUE.

See Also

plotNormCurves, plotPercentileSeries

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial()`, `plot.cnormBetaBinomial2()`, `plotDensity()`, `plotDerivative()`, `plotNorm()`, `plotNormCurves()`, `plotPercentileSeries()`, `plotRaw()`, `plotSubset()`

Examples

```
# Load example data set, compute model and plot results
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotPercentiles(result)
```

`plotPercentileSeries` *Generates a series of plots with number curves by percentile for different models*

Description

This functions makes use of 'plotPercentiles' to generate a series of plots with different number of predictors. It draws on the information provided by the model object to determine the bounds of the modeling (age and standard score range). It can be used as an additional model check to determine the best fitting model. Please have a look at the ' plotPercentiles' function for further information.

Usage

```
plotPercentileSeries(
  model,
  start = 1,
  end = NULL,
  group = NULL,
  percentiles = c(0.025, 0.1, 0.25, 0.5, 0.75, 0.9, 0.975),
  filename = NULL
)
```

Arguments

`model`	The Taylor polynomial regression model object from the cNORM
`start`	Number of predictors to start with
`end`	Number of predictors to end with
`group`	The name of the grouping variable; the distinct groups are automatically determined
`percentiles`	Vector with percentile scores, ranging from 0 to 1 (exclusive)
`filename`	Prefix of the filename. If specified, the plots are saves as png files in the directory of the workspace, instead of displaying them

Value

the complete list of plots

See Also

plotPercentiles

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial()`, `plot.cnormBetaBinomial2()`, `plotDensity()`, `plotDerivative()`, `plotNorm()`, `plotNormCurves()`, `plotPercentiles()`, `plotRaw()`, `plotSubset()`

Examples

```
# Load example data set, compute model and plot results
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotPercentileSeries(result, start=4, end=6)
```

`plotRaw` *Plot manifest and fitted raw scores*

Description

The function plots the raw data against the fitted scores from the regression model per group. This helps to inspect the precision of the modeling process. The scores should not deviate too far from regression line.

Usage

```
plotRaw(model, group = FALSE, raw = NULL, type = 0)
```

Arguments

`model`	The regression model from the 'cnorm' function
`group`	Should the fit be displayed by group?
`raw`	Vector of the observed raw data
`type`	Type of display: 0 = plot manifest against fitted values, 1 = plot manifest against difference values

See Also

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial()`, `plot.cnormBetaBinomial2()`, `plotDensity()`, `plotDerivative()`, `plotNorm()`, `plotNormCurves()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotSubset()`

Examples

```
# Compute model with example dataset and plot results
result <- cnorm(raw = elfe$raw, group = elfe$group)
plotRaw(result)
```

`plotSubset` *Evaluate information criteria for regression model*

Description

This function plots various information criteria and model fit statistics against the number of predictors or adjusted R-squared, depending on the type of plot selected. It helps in model selection by visualizing different aspects of model performance. Models, which did not pass the initial consistency check are depicted with an empty circle.

Usage

```
plotSubset(model, type = 0)
```

Arguments

`model`	The regression model from the bestModel function or a cnorm object.
`type`	Integer specifying the type of plot to generate: • 0: Adjusted R2 by number of predictors (default) • 1: Log-transformed Mallow's Cp by adjusted R2 • 2: Bayesian Information Criterion (BIC) by adjusted R2 • 3: Root Mean Square Error (RMSE) by number of predictors • 4: Residual Sum of Squares (RSS) by number of predictors • 5: F-test statistic for consecutive models by number of predictors • 6: p-value for model tests by number of predictors

Details

The function generates different plots to help in model selection:

- For types 1 and 2 (Mallow's Cp and BIC), look for the "elbow" in the curve where the information criterion begins to drop. This often indicates a good balance between model fit and complexity. - For type 0 (Adjusted R2), higher values indicate better fit, but be cautious of overfitting with values approaching 1. - For types 3 and 4 (RMSE and RSS), lower values indicate better fit. - For type 5 (F-test), higher values suggest significant improvement with added predictors. - For type 6 (p-values), values below the significance level (typically 0.05) suggest significant improvement with added predictors.

Value

A ggplot object representing the selected information criterion plot.

Note

It's important to balance statistical measures with practical considerations and to visually inspect the model fit using functions like `plotPercentiles`.

See Also

`bestModel`, `plotPercentiles`, `printSubset`

Other plot: `plot.cnorm()`, `plot.cnormBetaBinomial()`, `plot.cnormBetaBinomial2()`, `plotDensity()`, `plotDerivative()`, `plotNorm()`, `plotNormCurves()`, `plotPercentileSeries()`, `plotPercentiles()`, `plotRaw()`

Examples

```
# Compute model with example data and plot information function
cnorm.model <- cnorm(raw = elfe$raw, group = elfe$group)
plotSubset(cnorm.model)

# Plot BIC against adjusted R-squared
plotSubset(cnorm.model, type = 2)

# Plot RMSE against number of predictors
plotSubset(cnorm.model, type = 3)
```

`ppvt` *Vocabulary development from 2.5 to 17*

Description

A dataset based on an unstratified sample of PPVT4 data (German adaption). The PPVT4 consists of blocks of items with 12 items each. Each item consists of 4 pictures. The test taker is given a word orally and he or she has to point out the picture matching the oral word. Bottom and ceiling blocks of items are determined according to age and performance. For instance, when a student knows less than 4 word from a block of 12 items, the testing stops. The sample is not identical with the norm sample and includes doublets of cases in order to align the sample size per age group. It is primarily intended for running the cNORM analyses with regard to modeling and stratification.

Usage

```
ppvt
```

Format

A data frame with 4542 rows and 6 variables:

age the chronological age of the child

sex the sex of the test taker, 1=male, 2=female

migration migration status of the family, 0=no, 1=yes

region factor specifying the region, the data were collected; grouped into south, north, east and west

raw the raw score of the student, spanning values from 0 to 228

group age group of the child, determined by the getGroups()-function with 12 equidistant age groups

A data frame with 5600 rows and 9 columns

Source

https://www.psychometrica.de/ppvt4.html

References

Lenhard, A., Lenhard, W., Segerer, R. & Suggate, S. (2015). Peabody Picture Vocabulary Test - Revision IV (Deutsche Adaption). Frankfurt a. M./Germany: Pearson Assessment.

Examples

```
## Not run:
# Example with continuous age variable, ranked with sliding window
model.ppvt.sliding <- cnorm(age=ppvt$age, raw=ppvt$raw, width=1)

# Example with age groups; you might first want to experiment with
# the granularity of the groups via the 'getGroups()' function
model.ppvt.group <- cnorm(group=ppvt$group, raw=ppvt$raw) # with predefined groups
model.ppvt.group <- cnorm(group=getGroups(ppvt$age, n=15, equidistant = T),
                          raw=ppvt$raw) # groups built 'on the fly'


# plot information function
plot(model.ppvt.group, "subset")

# check model consistency
checkConsistency(model.ppvt.group)

# plot percentiles
plot(model.ppvt.group, "percentiles")

## End(Not run)
```

---

predict.cnormBetaBinomial

Predict Norm Scores from Raw Scores

Description

This function calculates norm scores based on raw scores, age, and a fitted cnormBetaBinomial model.

Usage

```
## S3 method for class 'cnormBetaBinomial'
predict(object, ...)
```

Arguments

`object`	A fitted model object of class 'cnormBetaBinomial' or 'cnormBetaBinomial2'.
`...`	Additional arguments passed to the prediction method: • age A numeric vector of ages, same length as raw. • score A numeric vector of raw scores. • range The range of the norm scores in standard deviations. Default is 3. Thus, scores in the range of +/- 3 standard deviations are considered.

Details

The function first predicts the alpha and beta parameters of the beta-binomial distribution for each age using the provided model. It then calculates the cumulative probability for each raw score given these parameters. Finally, it converts these probabilities to the norm scale specified in the model.

Value

A numeric vector of norm scores.

See Also

Other predict: `derivationTable()`, `getNormCurve()`, `normTable()`, `predict.cnormBetaBinomial2()`, `predictNorm()`, `predictRaw()`, `rawTable()`

Examples

```
## Not run:
# Assuming you have a fitted model named 'bb_model':
model <- cnorm.betabinomial(ppvt$age, ppvt$raw)
raw <- c(100, 121, 97, 180)
ages <- c(7, 8, 9, 10)
norm_scores <- predict(model, ages, raw)

## End(Not run)
```

`predict.cnormBetaBinomial2`

Predict Norm Scores from Raw Scores

Description

This function calculates norm scores based on raw scores, age, and a fitted cnormBetaBinomial model.

Usage

```
## S3 method for class 'cnormBetaBinomial2'
predict(object, ...)
```

Arguments

`object`	A fitted model object of class 'cnormBetaBinomial' or 'cnormBetaBinomial2'.
`...`	Additional arguments passed to the prediction method: • age A numeric vector of ages, same length as raw. • score A numeric vector of raw scores. • range The range of the norm scores in standard deviations. Default is 3. Thus, scores in the range of +/- 3 standard deviations are considered.

Details

The function first predicts the alpha and beta parameters of the beta-binomial distribution for each age using the provided model. It then calculates the cumulative probability for each raw score given these parameters. Finally, it converts these probabilities to the norm scale specified in the model.

Value

A numeric vector of norm scores.

See Also

Other predict: `derivationTable()`, `getNormCurve()`, `normTable()`, `predict.cnormBetaBinomial()`, `predictNorm()`, `predictRaw()`, `rawTable()`

Examples

```
## Not run:
# Assuming you have a fitted model named 'bb_model':
model <- cnorm.betabinomial(ppvt$age, ppvt$raw)
raw <- c(100, 121, 97, 180)
ages <- c(7, 8, 9, 10)
norm_scores <- predict(model, ages, raw)

## End(Not run)
```

`predictNorm` *Retrieve norm value for raw score at a specific age*

Description

This functions numerically determines the norm score for raw scores depending on the level of the explanatory variable A, e. g. norm scores for raw scores at given ages.

Usage

```
predictNorm(
  raw,
  A,
  model,
  minNorm = NULL,
  maxNorm = NULL,
  force = FALSE,
  silent = FALSE
)
```

Arguments

`raw`	The raw value, either single numeric or numeric vector
`A`	the explanatory variable (e. g. age), either single numeric or numeric vector
`model`	The regression model or a cnorm object
`minNorm`	The lower bound of the norm score range
`maxNorm`	The upper bound of the norm score range
`force`	Try to resolve missing norm scores in case of inconsistent models
`silent`	set to TRUE to suppress messages

Value

The predicted norm score for a raw score, either single value or vector

See Also

Other predict: `derivationTable()`, `getNormCurve()`, `normTable()`, `predict.cnormBetaBinomial()`, `predict.cnormBetaBinomial2()`, `predictRaw()`, `rawTable()`

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)

# return norm value for raw value 21 for grade 2, month 9
specificNormValue <- predictNorm(raw = 21, A = 2.75, cnorm.elfe)

# predicted norm scores for the elfe dataset
# predictNorm(elfe$raw, elfe$group, cnorm.elfe)
```

`predictRaw` *Predict raw values*

Description

Most elementary function to predict raw score based on Location (L, T score), Age (grouping variable) and the coefficients from a regression model.

Usage

```
predictRaw(norm, age, coefficients, minRaw = -Inf, maxRaw = Inf)
```

Arguments

`norm`	The norm score, e. g. a specific T score or a vector of scores
`age`	The age value or a vector of scores
`coefficients`	The a cnorm object or the coefficients from the regression model
`minRaw`	Minimum score for the results; can be used for clipping unrealistic outcomes, usually set to the lower bound of the range of values of the test (default: 0)
`maxRaw`	Maximum score for the results; can be used for clipping unrealistic outcomes usually set to the upper bound of the range of values of the test

Value

the predicted raw score or a data.frame of scores in case, lists of norm scores or age is used

See Also

Other predict: `derivationTable()`, `getNormCurve()`, `normTable()`, `predict.cnormBetaBinomial()`, `predict.cnormBetaBinomial2()`, `predictNorm()`, `rawTable()`

Examples

```
# Prediction of single scores
model <- cnorm(raw = elfe$raw, group = elfe$group)
predictRaw(35, 3.5, model)
```

`prepareData` *Prepare data for modeling in one step (convenience method)*

Description

This is a convenience method to either load the inbuilt sample dataset, or to provide a data frame with the variables "raw" (for the raw scores) and "group" The function ranks the data within groups, computes norm values, powers of the norm scores and interactions. Afterwards, you can use these preprocessed data to determine the best fitting model.

Usage

```
prepareData(
  data = NULL,
  group = "group",
  raw = "raw",
  age = "group",
  k = 4,
  t = NULL,
  width = NA,
  weights = NULL,
  scale = "T",
  descend = FALSE,
  silent = FALSE
)
```

Arguments

`data`	data.frame with a grouping variable named 'group' and a raw score variable named 'raw'.
`group`	grouping variable in the data, e. g. age groups, grades ... Setting group = FALSE deactivates modeling in dependence of age. Use this in case you do want conventional norm tables.
`raw`	the raw scores
`age`	the continuous explanatory variable; by default set to "group"
`k`	The power parameter, default = 4
`t`	the age power parameter (default NULL). If not set, cNORM automatically uses k. The age power parameter can be used to specify the k to produce rectangular matrices and specify the course of scores per independently from k
`width`	if a width is provided, the function switches to rankBySlidingWindow to determine the observed raw scores, otherwise, ranking is done by group (default)
`weights`	Vector or variable name in the dataset with weights for each individual case. It can be used to compensate for moderate imbalances due to insufficient norm data stratification. Weights should be numerical and positive. Please use the 'computeWeights' function for this purpose.

`scale`	type of norm scale, either T (default), IQ, z or percentile (= no transformation); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index point
`descend`	ranking order (default descent = FALSE): inverses the ranking order with higher raw scores getting lower norm scores; relevant for example when norming error scores, where lower scores mean higher performance
`silent`	set to TRUE to suppress messages

Details

The functions `rankBySlidingWindow`, `rankByGroup`, `bestModel`, `computePowers` and `prepareData` are usually not called directly, but accessed through other functions like `cnorm`.

Value

data frame including the norm scores, powers and interactions of the norm score and grouping variable

See Also

Other prepare: `computePowers()`, `rankByGroup()`, `rankBySlidingWindow()`

Examples

```
# conducts ranking and computation of powers and interactions with the 'elfe' dataset
data.elfe <- prepareData(elfe)

# use vectors instead of data frame
data.elfe <- prepareData(raw=elfe$raw, group=elfe$group)

# variable names can be specified as well, here with the BMI data included in the package
## Not run:
data.bmi <- prepareData(CDC, group = "group", raw = "bmi", age = "age")

## End(Not run)

# modeling with only one group with the 'elfe' dataset as an example
# this results in conventional norming
data.elfe2 <- prepareData(data = elfe, group = FALSE)
m <- bestModel(data.elfe2)
```

`prettyPrint` *Format raw and norm tables The function takes a raw or norm table, condenses intervals at the bottom and top and round the numbers to meaningful interval.*

Description

Format raw and norm tables The function takes a raw or norm table, condenses intervals at the bottom and top and round the numbers to meaningful interval.

Usage

```
prettyPrint(table)
```

Arguments

`table`	The table to format

Value

formatted table

`print.cnorm` *S3 method for printing model selection information*

Description

After conducting the model fitting procedure on the data set, the best fitting model has to be chosen. The print function shows the R2 and other information on the different best fitting models with increasing number of predictors.

Usage

```
## S3 method for class 'cnorm'
print(x, ...)
```

Arguments

`x`	The model from the 'bestModel' function or a cnorm object
`...`	additional parameters

Value

A table with information criteria

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

`printSubset` *Print Model Selection Information*

Description

Displays R^2 and other metrics for models with varying predictors, aiding in choosing the best-fitting model after model fitting.

Usage

```
printSubset(x, ...)
```

Arguments

`x`	Model output from 'bestModel' or a cnorm object.
`...`	Additional parameters.

Value

Table with model information criteria.

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `print.cnorm()`, `rangeCheck()`, `regressionFunction()`, `summary.cnorm()`

Examples

```
# Using cnorm object from sample data
result <- cnorm(raw = elfe$raw, group = elfe$group)
printSubset(result)
```

`rangeCheck` *Check for horizontal and vertical extrapolation*

Description

Regression model only work in a specific range and extrapolation horizontally (outside the original range) or vertically (extreme norm scores) might lead to inconsistent results. The function generates a message, indicating extrapolation and the range of the original data.

Usage

```
rangeCheck(
  object,
  minAge = NULL,
  maxAge = NULL,
  minNorm = NULL,
  maxNorm = NULL,
  digits = 3,
  ...
)
```

Arguments

`object`	The regression model or a cnorm object
`minAge`	The lower age bound
`maxAge`	The upper age bound
`minNorm`	The lower norm value bound
`maxNorm`	The upper norm value bound
`digits`	The precision for rounding the norm and age data
`...`	additional parameters

Value

the report

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `regressionFunction()`, `summary.cnorm()`

Examples

```
m <- cnorm(raw = elfe$raw, group = elfe$group)
rangeCheck(m)
```

`rankByGroup` *Determine the norm scores of the participants in each subsample*

Description

This is the initial step, usually done in all kinds of test norming projects, after the scale is constructed and the norm sample is established. First, the data is grouped according to a grouping variable and afterwards, the percentile for each raw value is retrieved. The percentile can be used for the modeling procedure, but in case, the samples to not deviate too much from normality, T, IQ or z scores can be computed via a normal rank procedure based on the inverse cumulative normal distribution. In case of bindings, we use the medium rank and there are different methods for estimating the percentiles (default RankIt).

Usage

```
rankByGroup(
  data = NULL,
  group = "group",
  raw = "raw",
  weights = NULL,
  method = 4,
  scale = "T",
  descend = FALSE,
  descriptives = TRUE,
  na.rm = TRUE,
  silent = FALSE
)
```

Arguments

`data`	data.frame with norm sample data. If no data.frame is provided, the raw score and group vectors are directly used
`group`	name of the grouping variable (default ’group’) or numeric vector, e. g. grade, setting group to FALSE cancels grouping (data is treated as one group)
`raw`	name of the raw value variable (default ’raw’) or numeric vector
`weights`	Vector or variable name in the dataset with weights for each individual case. It can be used to compensate for moderate imbalances due to insufficient norm data stratification. Weights should be numerical and positive. Please use the ’computeWeights’ function for this purpose.
`method`	Ranking method in case of bindings, please provide an index, choosing from the following methods: 1 = Blom (1958), 2 = Tukey (1949), 3 = Van der Warden (1952), 4 = Rankit (default), 5 = Levenbach (1953), 6 = Filliben (1975), 7 = Yu & Huang (2001)
`scale`	type of norm scale, either T (default), IQ, z or percentile (= no transformation); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index points
`descend`	ranking order (default descent = FALSE): inverses the ranking order with higher raw scores getting lower norm scores; relevant for example when norming error scores, where lower scores mean higher performance
`descriptives`	If set to TRUE (default), information in n, mean, median and standard deviation per group is added to each observation
`na.rm`	remove values, where the percentiles could not be estimated, most likely happens in the context of weighting
`silent`	set to TRUE to suppress messages

Value

the dataset with the percentiles and norm scales per group

Remarks on using covariates

So far the inclusion of a binary covariate is experimental and far from optimized. The according variable name has to be specified in the ranking procedure and the modeling includes this in the further process. At the moment, during ranking the data are split into the according cells group x covariate, which leads to small sample sizes. Please take care to have enough cases in each combination. Additionally, covariates can lead to unstable modeling solutions. The question, if it is really reasonable to include covariates when norming a test is a decision beyond the pure data modeling. Please use with care or alternatively split the dataset into the two groups beforehand and model them separately.

The functions `rankBySlidingWindow`, `rankByGroup`, `bestModel`, `computePowers` and `prepareData` are usually not called directly, but accessed through other functions like `cnorm`.

See Also

rankBySlidingWindow, computePowers, computeWeights, weighted.rank

Other prepare: `computePowers()`, `prepareData()`, `rankBySlidingWindow()`

Examples

```
# Transformation with default parameters: RankIt and converting to T scores
data.elfe <- rankByGroup(elfe, group = "group") # using a data frame with vector names
data.elfe2 <- rankByGroup(raw=elfe$raw, group=elfe$group) # use vectors for raw score and group

# Transformation into Wechsler scores with Yu & Huang (2001) ranking procedure
data.elfe <- rankByGroup(raw = elfe$raw, group = elfe$group, method = 7, scale = c(10, 3))

# cNORM can as well be used for conventional norming, in case no group is given
d <- rankByGroup(raw = elfe$raw)
d <- computePowers(d)
m <- bestModel(d)
rawTable(0, m) # please use an arbitrary value for age when generating the tables
```

`rankBySlidingWindow` *Determine the norm scores of the participants by sliding window*

Description

The function retrieves all individuals in the predefined age range (x +/- width/2) around each case and ranks that individual based on this individually drawn sample. This function can be directly used with a continuous age variable in order to avoid grouping. When collecting data on the basis of a continuous age variable, cases located far from the mean age of the group receive distorted percentiles when building discrete groups and generating percentiles with the traditional approach. The distortion increases with distance from the group mean and this effect can be avoided by the sliding window. Nonetheless, please ensure, that the optional grouping variable in fact represents the correct mean age of the respective age groups, as this variable is later on used for displaying the manifest data in the percentile plots.

Usage

```
rankBySlidingWindow(
  data = NULL,
  age = "age",
  raw = "raw",
  weights = NULL,
  width,
  method = 4,
  scale = "T",
  descend = FALSE,
  descriptives = TRUE,
  nGroup = 0,
  group = NA,
  na.rm = TRUE,
  silent = FALSE
)
```

Arguments

`data`	data.frame with norm sample data
`age`	the continuous age variable. Setting 'age' to FALSE inhibits computation of powers of age and the interactions
`raw`	name of the raw value variable (default 'raw')
`weights`	Vector or variable name in the dataset with weights for each individual case. It can be used to compensate for moderate imbalances due to insufficient norm data stratification. Weights should be numerical and positive. It can be resource intense when applied to the sliding window. Please use the 'computeWeights' function for this purpose.
`width`	the width of the sliding window
`method`	Ranking method in case of bindings, please provide an index, choosing from the following methods: 1 = Blom (1958), 2 = Tukey (1949), 3 = Van der Warden (1952), 4 = Rankit (default), 5 = Levenbach (1953), 6 = Filliben (1975), 7 = Yu & Huang (2001)
`scale`	type of norm scale, either T (default), IQ, z or percentile (= no transformation); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index points
`descend`	ranking order (default descent = FALSE): inverses the ranking order with higher raw scores getting lower norm scores; relevant for example when norming error scores, where lower scores mean higher performance
`descriptives`	If set to TRUE (default), information in n, mean, median and standard deviation per group is added to each observation
`nGroup`	If set to a positive value, a grouping variable is created with the desired number of equi distant groups, named by the group mean age of each group. It creates the column 'group' in the data.frame and in case, there is already one with that name, overwrites it.

`group`	Optional parameter for providing the name of the grouping variable (if present; overwritten if ngroups is used)
`na.rm`	remove values, where the percentiles could not be estimated, most likely happens in the context of weighting
`silent`	set to TRUE to suppress messages

Details

In case of bindings, the function uses the medium rank and applies the algorithms already described in the `rankByGroup` function. At the upper and lower end of the data sample, the sliding stops and the sample is drawn from the interval min + width and max - width, respectively.

Value

the dataset with the individual percentiles and norm scores

Remarks on using covariates

So far the inclusion of a binary covariate is experimental and far from optimized. The according variable name has to be specified in the ranking procedure and the modeling includes this in the further process. At the moment, during ranking the data are split into the according degrees of the covariate and the ranking is done separately. This may lead to small sample sizes. Please take care to have enough cases in each combination. Additionally, covariates can lead to unstable modeling solutions. The question, if it is really reasonable to include covariates when norming a test is a decision beyond the pure data modeling. Please use with care or alternatively split the dataset into the two groups beforehand and model them separately.

The functions `rankBySlidingWindow`, `rankByGroup`, `bestModel`, `computePowers` and `prepareData` are usually not called directly, but accessed through other functions like `cnorm`.

See Also

rankByGroup, computePowers, computeWeights, weighted.rank, weighted.quantile

Other prepare: `computePowers()`, `prepareData()`, `rankByGroup()`

Examples

```
## Not run:
# Transformation using a sliding window
data.elfe2 <- rankBySlidingWindow(relfe, raw = "raw", age = "group", width = 0.5)

# Comparing this to the traditional approach should give us exactly the same
# values, since the sample dataset only has a grouping variable for age
data.elfe <- rankByGroup(elfe, group = "group")
mean(data.elfe$normValue - data.elfe2$normValue)

## End(Not run)
```

`rawTable` *Create a table with norm scores assigned to raw scores for a specific age based on the regression model*

Description

This function is comparable to 'normTable', despite it reverses the assignment: A table with raw scores and the according norm scores for a specific age based on the regression model is generated. This way, the inverse function of the regression model is solved numerically with brute force. Please specify the range of raw values, you want to cover. With higher precision and smaller stepping, this function becomes computational intensive. In case a confidence coefficient (CI, default .9) and the reliability is specified, confidence intervals are computed for the true score estimates, including a correction for regression to the mean (Eid & Schmidt, 2012, p. 272).

Usage

```
rawTable(
  A,
  model,
  minRaw = NULL,
  maxRaw = NULL,
  minNorm = NULL,
  maxNorm = NULL,
  step = 1,
  monotonuous = TRUE,
  CI = 0.9,
  reliability = NULL,
  pretty = TRUE
)
```

Arguments

`A`	the age, either single value or vector with age values
`model`	The regression model or a cnorm object
`minRaw`	The lower bound of the raw score range
`maxRaw`	The upper bound of the raw score range
`minNorm`	Clipping parameter for the lower bound of norm scores (default 25)
`maxNorm`	Clipping parameter for the upper bound of norm scores (default 25)
`step`	Stepping parameter for the raw scores (default 1)
`monotonuous`	corrects for decreasing norm scores in case of model inconsistencies (default)
`CI`	confidence coefficient, ranging from 0 to 1, default .9
`reliability`	coefficient, ranging between 0 to 1
`pretty`	Format table by collapsing intervals and rounding to meaningful precision

Value

either data.frame with raw scores and the predicted norm scores in case of simple A value or a list of norm tables if vector of A values was provided

References

Eid, M. & Schmidt, K. (2012). Testtheorie und Testkonstruktion. Hogrefe.

See Also

normTable

Other predict: `derivationTable()`, `getNormCurve()`, `normTable()`, `predict.cnormBetaBinomial()`, `predict.cnormBetaBinomial2()`, `predictNorm()`, `predictRaw()`

Examples

```
# Generate cnorm object from example data
cnorm.elfe <- cnorm(raw = elfe$raw, group = elfe$group)
# generate a norm table for the raw value range from 0 to 28 for the time point month 7 of grade 3
table <- rawTable(3 + 7 / 12, cnorm.elfe, minRaw = 0, maxRaw = 28)

# generate several raw tables
table <- rawTable(c(2.5, 3.5, 4.5), cnorm.elfe, minRaw = 0, maxRaw = 28)

# additionally compute confidence intervals
table <- rawTable(c(2.5, 3.5, 4.5), cnorm.elfe, minRaw = 0, maxRaw = 28, CI = .9, reliability = .94)

# conventional norming, set age to arbitrary value
model <- cnorm(raw=elfe$raw)
rawTable(0, model)
```

`regressionFunction` *Regression function*

Description

The method builds the regression function for the regression model, including the beta weights. It can be used to predict the raw scores based on age and location.

Usage

```
regressionFunction(model, raw = NULL, digits = NULL)
```

Arguments

`model`	The regression model from the bestModel function or a cnorm object
`raw`	The name of the raw value variable (default ’raw’)
`digits`	Number of digits for formatting the coefficients

Value

The regression formula as a string

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `summary.cnorm()`

Examples

```
result <- cnorm(raw = elfe$raw, group = elfe$group)
regressionFunction(result)
```

`simMean` *Simulate mean per age*

Description

Simulate mean per age

Usage

```
simMean(age)
```

Arguments

`age`	the age variable

Value

return predicted means

Examples

```
## Not run:
x <- simMean(a)

## End(Not run)
```

`simSD` *Simulate sd per age*

Description

Simulate sd per age

Usage

```
simSD(age)
```

Arguments

`age`	the age variable

Value

return predicted sd

Examples

```
## Not run:
x <- simSD(a)

## End(Not run)
```

`simulateRasch` *Simulate raw test scores based on Rasch model*

Description

For testing purposes only: The function simulates raw test scores based on a virtual Rasch based test with n results per age group, an evenly distributed age variable, items.n test items with a simulated difficulty and standard deviation. The development trajectories over age group are modeled by a curve linear function of age, with at first fast progression, which slows down over age, and a slightly increasing standard deviation in order to model a scissor effects. The item difficulties can be accessed via $theta and the raw data via $data of the returned object.

Usage

```
simulateRasch(
  data = NULL,
  n = 100,
  minAge = 1,
  maxAge = 7,
  items.n = 21,
```

```
  items.m = 0,
  items.sd = 1,
  Theta = "random",
  width = 1
)
```

Arguments

`data`	data.frame from previous simulations for recomputation (overrides n, minAge, maxAge)
`n`	The sample size per age group
`minAge`	The minimum age (default 1)
`maxAge`	The maximum age (default 7)
`items.n`	The number of items of the test
`items.m`	The mean difficulty of the items
`items.sd`	The standard deviation of the item difficulty
`Theta`	irt scales difficulty parameters, either "random" for drawing a random sample, "even" for evenly distributed or a set of predefined values, which then overrides the item.n parameters
`width`	The width of the window size for the continuous age per group; +- 1/2 width around group center on items.m and item.sd; if set to FALSE, the distribution is not drawn randomly but normally nonetheless

Value

a list containing the simulated data and thetas

data the data.frame with only age, group and raw

sim the complete simulated data with item level results

theta the difficulty of the items

Examples

```
# simulate data for a rather easy test (m = -1.0)
sim <- simulateRasch(n=150, minAge=1,
                     maxAge=7, items.n = 30, items.m = -1.0,
                     items.sd = 1, Theta = "random", width = 1.0)

# Show item difficulties
mean(sim$theta)
sd(sim$theta)
hist(sim$theta)

# Plot raw scores
boxplot(raw~group, data=sim$data)

# Model data
data <- prepareData(sim$data, age="age")
```

```
model <- bestModel(data, k = 4)
printSubset(model)
plotSubset(model, type=0)
```

`standardize` *Standardize a numeric vector*

Description

This function standardizes a numeric vector by subtracting the mean and dividing by the standard deviation. The resulting vector will have a mean of 0 and a standard deviation of 1.

Usage

```
standardize(x)
```

Arguments

`x`	A numeric vector to be standardized.

Value

A numeric vector of the same length as x, containing the standardized values.

Examples

```
data <- c(1, 2, 3, 4, 5)
standardized_data <- standardize(data)
print(standardized_data)
```

`standardizeRakingWeights` *Function for standardizing raking weights Raking weights get divided by the smallest weight. Thereby, all weights become larger or equal to 1 without changing the ratio of the weights to each other.*

Description

Function for standardizing raking weights Raking weights get divided by the smallest weight. Thereby, all weights become larger or equal to 1 without changing the ratio of the weights to each other.

Usage

```
standardizeRakingWeights(weights)
```

Arguments

`weights`	Raking weights computed by computeWeights()

Value

the standardized weights

`summary.cnorm` *S3 method for printing the results and regression function of a cnorm model*

Description

S3 method for printing the results and regression function of a cnorm model

Usage

```
## S3 method for class 'cnorm'
summary(object, ...)
```

Arguments

`object`	A regression model or cnorm object
`...`	additional parameters

Value

A report on the regression function, weights, R2 and RMSE

See Also

Other model: `bestModel()`, `checkConsistency()`, `cnorm.cv()`, `derive()`, `modelSummary()`, `print.cnorm()`, `printSubset()`, `rangeCheck()`, `regressionFunction()`

`summary.cnormBetaBinomial`

Summarize a Beta-Binomial Continuous Norming Model

Description

This function provides a summary of a fitted beta-binomial continuous norming model, including model fit statistics, convergence information, and parameter estimates.

Usage

```
## S3 method for class 'cnormBetaBinomial'
summary(object, ...)
```

Arguments

`object`	An object of class "cnormBetaBinomial" or "cnormBetaBinomial2", typically the result of a call to `cnorm.betabinomial`.
`...`	Additional arguments passed to the summary method: • age An optional numeric vector of age values corresponding to the raw scores. If provided along with `raw`, additional fit statistics (R-squared, RMSE, bias) will be calculated. • score An optional numeric vector of raw scores. Must be provided if age is given. • weights An optional numeric vector of weights for each observation.

Details

The summary includes:

- Basic model information (type, number of observations, number of parameters)
- Model fit statistics (log-likelihood, AIC, BIC)
- R-squared, RMSE, and bias (if age and raw scores are provided) in comparison to manifest norm scores
- Convergence information
- Parameter estimates with standard errors, z-values, and p-values

Value

Invisibly returns a list containing detailed diagnostic information about the model. The function primarily produces printed output summarizing the model.

See Also

`cnorm.betabinomial`, `diagnostics.betabinomial`

Examples

```
## Not run:
model <- cnorm.betabinomial(ppvt$age, ppvt$raw, n = 228)
summary(model)

# Including R-squared, RMSE, and bias in the summary:
summary(model, age = ppvt$age, score = ppvt$raw)

## End(Not run)
```

`summary.cnormBetaBinomial2`

Summarize a Beta-Binomial Continuous Norming Model

Description

This function provides a summary of a fitted beta-binomial continuous norming model, including model fit statistics, convergence information, and parameter estimates.

Usage

```
## S3 method for class 'cnormBetaBinomial2'
summary(object, ...)
```

Arguments

`object`	An object of class "cnormBetaBinomial" or "cnormBetaBinomial2", typically the result of a call to `cnorm.betabinomial`.
`...`	Additional arguments passed to the summary method: • age An optional numeric vector of age values corresponding to the raw scores. If provided along with `raw`, additional fit statistics (R-squared, RMSE, bias) will be calculated. • score An optional numeric vector of raw scores. Must be provided if age is given. • weights An optional numeric vector of weights for each observation.

Details

The summary includes:

- Basic model information (type, number of observations, number of parameters)
- Model fit statistics (log-likelihood, AIC, BIC)
- R-squared, RMSE, and bias (if age and raw scores are provided) in comparison to manifest norm scores
- Convergence information
- Parameter estimates with standard errors, z-values, and p-values

Value

Invisibly returns a list containing detailed diagnostic information about the model. The function primarily produces printed output summarizing the model.

See Also

`cnorm.betabinomial`, `diagnostics.betabinomial`

Examples

```
## Not run:
model <- cnorm.betabinomial(ppvt$age, ppvt$raw, n = 228)
summary(model)

# Including R-squared, RMSE, and bias in the summary:
summary(model, age = ppvt$age, raw = ppvt$raw)

## End(Not run)
```

`taylorSwift` *Swiftly compute Taylor regression models for distribution free continuous norming*

Description

Conducts distribution free continuous norming and aims to find a fitting model. Raw data are modelled as a Taylor polynomial of powers of age and location and their interactions. In addition to the raw scores, either provide a numeric vector for the grouping information (group) for the ranking of the raw scores. You can adjust the grade of smoothing of the regression model by setting the k, t and terms parameter. In general, increasing k and t leads to a higher fit, while lower values lead to more smoothing. If both parameters are missing, taylorSwift uses k = 5 and t = 3 by default.

Usage

```
taylorSwift(
  raw = NULL,
  group = NULL,
  age = NULL,
  width = NA,
  weights = NULL,
  scale = "T",
  method = 4,
  descend = FALSE,
  k = NULL,
  t = NULL,
  terms = 0,
  R2 = NULL,
```

```
  plot = TRUE,
  extensive = TRUE
)
```

Arguments

`raw`	Numeric vector of raw scores
`group`	Numeric vector of grouping variable, e. g. grade. If no group or age variable is provided, conventional norming is applied
`age`	Numeric vector with chronological age, please additionally specify width of window
`width`	Size of the moving window in case an age vector is used
`weights`	Vector or variable name in the dataset with weights for each individual case. It can be used to compensate for moderate imbalances due to insufficient norm data stratification. Weights should be numerical and positive.
`scale`	type of norm scale, either T (default), IQ, z or percentile (= no transformation); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index points
`method`	Ranking method in case of bindings, please provide an index, choosing from the following methods: 1 = Blom (1958), 2 = Tukey (1949), 3 = Van der Warden (1952), 4 = Rankit (default), 5 = Levenbach (1953), 6 = Filliben (1975), 7 = Yu & Huang (2001)
`descend`	ranking order (default descent = FALSE): inverses the ranking order with higher raw scores getting lower norm scores; relevant for example when norming error scores, where lower scores mean higher performance
`k`	The power constant. Higher values result in more detailed approximations but have the danger of over-fit (max = 6). If not set, it uses t and if both parameters are NULL, k is set to 5.
`t`	The age power parameter (max = 6). If not set, it uses k and if both parameters are NULL, k is set to 3, since age trajectories are most often well captured by cubic polynomials.
`terms`	Selection criterion for model building. The best fitting model with this number of terms is used
`R2`	Adjusted R square as a stopping criterion for the model building (default R2 = 0.99)
`plot`	Default TRUE; plots the regression model and prints report
`extensive`	If TRUE, screen models for consistency and - if possible, exclude inconsistent ones

Value

cnorm object including the ranked raw data and the regression model

References

1. Gary, S. & Lenhard, W. (2021). In norming we trust. Diagnostica.
2. Gary, S., Lenhard, W. & Lenhard, A. (2021). Modelling Norm Scores with the cNORM Package in R. Psych, 3(3), 501-521. https://doi.org/10.3390/psych3030033
3. Lenhard, A., Lenhard, W., Suggate, S. & Segerer, R. (2016). A continuous solution to the norming problem. Assessment, Online first, 1-14. doi:10.1177/1073191116656437
4. Lenhard, A., Lenhard, W., Gary, S. (2018). Continuous Norming (cNORM). The Comprehensive R Network, Package cNORM, available: https://CRAN.R-project.org/package=cNORM
5. Lenhard, A., Lenhard, W., Gary, S. (2019). Continuous norming of psychometric tests: A simulation study of parametric and semi-parametric approaches. PLoS ONE, 14(9), e0222279. doi:10.1371/journal.pone.0222279
6. Lenhard, W., & Lenhard, A. (2020). Improvement of Norm Score Quality via Regression-Based Continuous Norming. Educational and Psychological Measurement(Online First), 1-33. https://doi.org/10.1177/0013164420928457

See Also

rankByGroup, rankBySlidingWindow, computePowers, bestModel

Examples

```
## Not run:
# Using this function with the example dataset 'ppvt'
# You can use the 'getGroups()' function to set up grouping variable in case,
# you have a continuous age variable.
model <- taylorSwift(raw = ppvt$raw, group = ppvt$group)

# return norm tables including 90% confidence intervals for a
# test with a reliability of r = .85; table are set to mean of quartal
# in grade 3 (children completed 2 years of schooling)
normTable(c(5, 15), model, CI = .90, reliability = .95)

# ... or instead of raw scores for norm scores, the other way round
rawTable(c(8, 12), model, CI = .90, reliability = .95)

## End(Not run)
```

`weighted.quantile` *Weighted quantile estimator*

Description

Computes weighted quantiles (code from Andrey Akinshin (2023) "Weighted quantile estimators" arXiv:2304.07265 [stat.ME] Code made available via the CC BY-NC-SA 4.0 license) on the basis of either the weighted Harrell-Davis quantile estimator or an adaption of the type 7 quantile estimator of the generic quantile function in the base package. Please provide a vector with raw values, the probabilities for the quantiles and an additional vector with the weight of each observation. In case the weight vector is NULL, a normal quantile estimation is done. The vectors may not include NAs and the weights should be positive non-zero values. Please draw on the computeWeights() function for retrieving weights in post stratification.

Usage

```
weighted.quantile(x, probs, weights = NULL, type = "Harrell-Davis")
```

Arguments

`x`	A numerical vector
`probs`	Numerical vector of quantiles
`weights`	A numerical vector with weights; should have the same length as x
`type`	Type of estimator, can either be "inflation", "Harrell-Davis" using a beta function to approximate the weighted percentiles (Harrell & Davis, 1982) or "Type7" (default; Hyndman & Fan, 1996), an adaption of the generic quantile function in R, including weighting. The inflation procedure is essentially a numerical, non-parametric solution that gives the same results as Harrel-Davis. It requires less ressources with small datasets and always finds a solution (e. g. 1000 cases with weights between 1 and 10). If it becomes too resource intense, it switches to Harrell-Davis automatically. Harrel-Davis and Type7 code is based on the work of Akinshin (2023).

Value

the weighted quantiles

References

1. Harrell, F.E. & Davis, C.E. (1982). A new distribution-free quantile estimator. Biometrika, 69(3), 635-640.
2. Hyndman, R. J. & Fan, Y. (1996). Sample quantiles in statistical packages, American Statistician 50, 361–365.
3. Akinshin, A. (2023). Weighted quantile estimators arXiv:2304.07265 [stat.ME]

See Also

weighted.quantile.inflation, weighted.quantile.harrell.davis, weighted.quantile.type7

`weighted.quantile.harrell.davis`

Weighted Harrell-Davis quantile estimator

Description

Computes weighted quantiles; code from Andrey Akinshin (2023) "Weighted quantile estimators" arXiv:2304.07265 [stat.ME] Code made available via the CC BY-NC-SA 4.0 license

Usage

```
weighted.quantile.harrell.davis(x, probs, weights = NULL)
```

Arguments

`x`	A numerical vector
`probs`	Numerical vector of quantiles
`weights`	A numerical vector with weights; should have the same length as x. If no weights are provided (NULL), it falls back to the base quantile function, type 7

Value

the quantiles

`weighted.quantile.inflation`

Weighted quantile estimator through case inflation

Description

Applies weighted ranking numerically by inflating cases according to weight. This function will be resource intensive, if inflated cases get too high and in this cases, it switches to the parametric Harrell-Davis estimator.

Usage

```
weighted.quantile.inflation(
  x,
  probs,
  weights = NULL,
  degree = 3,
  cutoff = 1e+07
)
```

Arguments

`x`	A numerical vector
`probs`	Numerical vector of quantiles
`weights`	A numerical vector with weights; should have the same length as x.
`degree`	power parameter for case inflation (default = 3, equaling factor 1000) If no weights are provided (NULL), it falls back to the base quantile function, type 7
`cutoff`	stop criterion for the sum of standardized weights to switch to Harrell-Davis, default = 1000000

Value

the quantiles

`weighted.quantile.type7`

Weighted type7 quantile estimator

Description

Computes weighted quantiles; code from Andrey Akinshin (2023) "Weighted quantile estimators" arXiv:2304.07265 [stat.ME] Code made available via the CC BY-NC-SA 4.0 license

Usage

```
weighted.quantile.type7(x, probs, weights = NULL)
```

Arguments

`x`	A numerical vector
`probs`	Numerical vector of quantiles
`weights`	A numerical vector with weights; should have the same length as x. If no weights are provided (NULL), it falls back to the base quantile function, type 7

Value

the quantiles

`weighted.rank` *Weighted rank estimation*

Description

Conducts weighted ranking on the basis of sums of weights per unique raw score. Please provide a vector with raw values and an additional vector with the weight of each observation. In case the weight vector is NULL, a normal ranking is done. The vectors may not include NAs and the weights should be positive non-zero values.

Usage

```
weighted.rank(x, weights = NULL)
```

Arguments

`x`	A numerical vector
`weights`	A numerical vector with weights; should have the same length as x

Value

the weighted absolute ranks

Index