Package 'cNORM'

March 15, 2019

Type Package

Title Continuous Norming

Version 1.1.8

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

Date 2019-03-15

Description Conventional methods for producing standard scores in psychometrics or biometrics are often plagued with ``jumps'' or ``gaps'' (i.e., discontinuities) in norm tables and low confidence for assessing extreme scores. The continuous norming method introduced by A. Lenhard et al. (2016), <doi:10.1177/1073191116656437>, generates continuous test norm scores on the basis of the raw data from standardization samples, without requiring assumptions about the distribution of the raw data: Norm scores are directly established from raw data by modeling the latter ones as a function of both percentile scores and an explanatory variable (e.g., age). The method minimizes bias arising from sampling and measurement error, while handling marked deviations from normality, addressing bottom or ceiling effects and capturing almost all of the variance in the original norm data sample.

Depends R (>= 3.1.0)

Imports lattice (>= 0.20), leaps (>= 3.0.0), latticeExtra (>= 0.6)

Suggests knitr, markdown, shiny, shinycssloaders, foreign, readxl, rmarkdown

License AGPL-3

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

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

Author Wolfgang Lenhard [cre, aut] (<https://orcid.org/0000-0002-8184-6889>),
Alexandra Lenhard [aut],
Sebastian Gary [ctb]

Date/Publication 2019-03-15 12:23:38 UTC

R topics documented:

`bestModel` *Retrieve the best fitting regression model based on powers of A, L and interactions*

Description

The function computes a series of regressions with an increasing number of predictors and takes the best fitting model per step. The aim is to find a model with as few predictors as possible, which at the same time manages to explain as much variance as possible from the original data. In psychometric test construction, this approach can be used to smooth the data and eliminate noise from norm sample stratification, while preserving the overall diagnostic information. Values around R2 = .99 usually show excellent results. The selection of the model can either be based on the number of terms in the regression functions or the share of explained variance of the model (R2). If both are specified, first the method tries to select the model based on the number of terms and in case, this does not work, use R2 instead. Pushing R2 by setting the number of terms, the R2 cut off and k to high values might lead to on over-fit, so be careful! These parameters depend on the distribution of the norm data. As a rule of thumb, terms = 5 or R2 = .99 and k = 4 is a good starting point for the analyses. `plotSubset(model)` can be used to weigh up R2 and information criteria (Cp, an AIC like measure) and fitted versus manifest scores can be plotted with 'plotRaw', 'plotNorm' and 'plotPercentiles'. Use `checkConsistency(model)` to check the model for violations. `cnorm.cv` can help in identifiying the ideal number of predictors.

Usage

```
bestModel(data, raw = NULL, R2 = 0.99, k = NULL, predictors = NULL,
  terms = 0, force.in = NULL)
```

Arguments

`data`	The preprocessed dataset, which should include the variables 'raw' and the powers and interactions of the norm score (L = Location; usually T scores) and an explanatory variably (usually age = A)
`raw`	the name of the raw score variable (default raw)
`R2`	Adjusted R square as a stopping criterion for the model building (default R2 = 0.99)
`k`	The power constant. Higher values result in more detailed approximations but have the danger of over-fit (default = 4, max = 6)
`predictors`	List of the names of predictor to use for the model selection. The parameter overrides the 'k' parameter and it can be used to preselect the variables entering the regression, or even to add variables like sex, that are not part of the original model building. Please note, that adding other variables than those based on L and A, plotting, prediction and normTable function will most likely not work, but at least the regression formula can be obtained that way. The parameter as well accepts a formula object, f. e. when applying a pre computed model to a new dataset. In this case, k is as well overridden. In order to include all

predictors in the regression, you might want to adjust the terms parameter to the number of predictors as well.

`terms` Selection criterion for model building. The best fitting model with this number of terms is used

`force.in` List of variable names forced into the regression function. This option can be used to force the regression to include covariates like sex or other background variables. This can be used to model separate norm scales for different groups in order the sample. Variables specified here, that are not part of the initial regression function resp. list of predictors, are ignored without further notice and thus do not show up in the final result. Additionally, all other functions like norm table generation and plotting are so far not yet prepared to handle covariates.

Value

The model meeting the R2 criteria with coefficients and variable selection in model$coefficients. Use `plotSubset(model)` and `plotPercentiles(data, model)` to inspect model

See Also

plotSubset, plotPercentiles, plotPercentileSeries, checkConsistency

Examples

```
## Not run:
# Standard example with sample data
normData <- prepareData()
model <- bestModel(normData)
plotSubset(model)
plotPercentiles(normData, model)

# It is possible to specify the variables explicitly - useful to smuggle
# in variables like sex
preselectedModel <- bestModel(normData, predictors = c("L1", "L3", "L1A3", "A2", "A3"))
print(regressionFunction(preselectedModel))

# Example for modeling based on continuous age variable and raw variable,
# based on the CDC data. We use the default k=4 parameter; raw variable has
# to be set to "bmi".
bmi.data <- prepareData(CDC, raw = "bmi", group = "group", age = "age")
bmi.model <- bestModel(bmi.data, raw = "bmi")
printSubset(bmi.model)

# Use the formula of the pre calculated bmi data to compute models for girls and
# boys seperately
bmi.model.boys <- bestModel(bmi.data[bmi.data$sex == 1, ], predictors = bmi.model$terms)
bmi.model.girls <- bestModel(bmi.data[bmi.data$sex == 2, ], predictors = bmi.model$terms)


# Custom list of predictors (based on k = 3) and forcing in the sex variable
# While calculating the regression model works well, all other functions like
```

```
# plotting and norm table generation are not yet prepared to use covariates
bmi.sex <- bestModel(bmi.data, raw = "bmi", predictors = c(
  "L1", "L2", "L3",
  "A1", "A2", "A3", "L1A1", "L1A2", "L1A3", "L2A1", "L2A2",
  "L2A3", "L3A1", "L3A2", "L3A3", "sex"
), force.in = c("sex"))

## End(Not run)
```

---

boxcox *Generate box cox power function for regression model at specific age*

Description

Applies a curve fitting for the regression model with the Box-Cox power transformation via the LMS method of Cole and Green (1992) at a specific age. Therefore, it simulates a data set and applies the transformation. It iteratively determines the power transformation lambda parameter with a precision up to 10^E-5 with a lambda value of 1 indicating normal distribution, values between 0 and 1 representing negative skew and values above 1 positive skewness of the distribution. The function is an optional step following the non-parametric modeling in order to conduct a parametric fitting of the percentiles.

Usage

```
boxcox(model, age, n = 250, m = 50, sd = 10)
```

Arguments

model	The regression model
age	The specific age
n	Number of simulated observations, used to span a percentile range from .5/n to (n-.5)/n with equally distanced percentiles
m	Scale mean of norm scale (default 50)
sd	Scale sd of norm scale (default 10)

Value

a list including the data.frame with percentiles, norm scores, fitted raw scores of the regression model and the fitted scores of the box cox curve fitting (indicated by variable names 'BC'), as well as the parameters for the box cox function (mean, sd, lambda) for the specified the age:

median	The median of the raw value distribution, estimated by the regression model
meanBC	The mean of the box cox function
sdBC	The standard deviation of the box cox function
lambdaBC	The skewness parameter of the box cox function
age	The age for which the power function was modeled

data	The data frame including the generated percentiles, the according norm scores, the fitted raw scores according to the regression model, the retrieved norm scores by the box cox transformation, the according density and percentile

References

Cole, T. J., & Green, P. J. (1992). Smoothing reference percentile curves: the LMS method and penalized likelihood. Statistics in medicine, 11(10), 1305-1319.

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. Journal of the Royal Statistical Society. Series B (Methodological), 211-252.

See Also

predictNormBC, predictRawBC

Examples

```
# model sample data set
model <- bestModel(prepareData())

# fitting scores of regression model box cox power function at specific age and retrieving
# the parameters for the box cox power function
bcParameters <- boxcox(model, 3)
```

---

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

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

Source

https://wwwn.cdc.gov/nchs/nhanes/OtherNhanesData.aspx

References

CDC (2012). National Health and Nutrition Examination Survey: Questionnaires, Datasets and Related Documentation. available https://wwwn.cdc.gov/nchs/nhanes/OtherNhanesData.aspx (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 linear increase with increasing raw scores. Violations of this assumption are a strong 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.

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.

Usage

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

Arguments

model	The model from the bestModel function
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, usually 1 or 0.1; lower 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

Value

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

Examples

```
normData <- prepareData()
m <- bestModel(normData)
modelViolations <- checkConsistency(m,
  minAge = 2, maxAge = 5, stepAge = 0.1,
  minNorm = 25, maxNorm = 75, minRaw = 0, maxRaw = 28, stepNorm = 1
)
plotDerivative(m, , minAge = 2, maxAge = 5, minNorm = 25, maxNorm = 75)
```

---

cNORM *cNORM: Continuous Norming*

Description

The package provides methods for generating regression based continuous standard scores, as f. e. for psychometric test development, biometrics (e. g. physiological growth curves), and screenings in the medical domain. Contrary to parametric approaches, it does not rely on distribution assumptions of the initial norm data and is thus a very robust approach in generating norm tables.

Details

Conventional methods for producing test norm score tables are often plagued with "jumps" or "gaps" (i.e., discontinuities) in norm tables and low confidence for assessing extreme scores. The continuous norming method introduced by A. Lenhard et al. (2016), <doi:10.1177/1073191116656437> addresses these problems and also has the added advantage of not requiring assumptions about the distribution of the raw data: The norm scores are established from raw data by modeling the latter ones as a function of both percentile scores and an explanatory variable (e.g., age). The method minimizes bias arising from sampling and measurement error, while handling marked deviations from normality - such as are commonplace in clinical samples.

Conducting the analysis consists of four steps and cNORM offers all according functions for preparing data, conducting the regression, selecting the best model and generating norm tables (according functions in brackets):

1. Data preparation (rankByGroup, rankBySlidingWindow, computePowers)
2. Establishing the regression model and selecting the parameters (bestModel, printSubset, plotSubset, regressionFunction, derive)
3. Validating the model (checkConsistency, plotPercentiles, plotPercentileSeries, plotRaw, plotNorm, derivationTable, plotDerivative)
4. Generating norm tables and predicting scores (predictNorm, predictRaw, normTable, getNormCurve, plotNormCurves)

For an easy start, you can use the graphical user interface by typing `cNORM.GUI()` on the console. Example datasets with large cohorts are available for demonstration purposes ('elfe', 'ppvt', 'CDC', 'life' and 'mortality' sample data from the references). Use `data <- prepareData(elfe)` or `data <- prepareData(ppvt)` to load and prepare example data for the modeling. Use `vignette(cNORM-Demo)` for a walk through on conducting the modeling and https://www.psychometrica.de/cNorm_en.html for a comprehensive tutorial.

Author(s)

Wolfgang Lenhard, Alexandra Lenhard and Sebastian Gary

References

1. CDC (2012). National Health and Nutrition Examination Survey: Questionnaires, Datasets and Related Documentation. available: https://wwwn.cdc.gov/nchs/nhanes/OtherNhanesData.aspx. date of retrieval: 25/08/2018
2. 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
3. Lenhard, A., Lenhard, W., Segerer, R. & Suggate, S. (2015). Peabody Picture Vocabulary Test - Revision IV (German Adaption). Frankfurt a. M.: Pearson Assessment.
4. Lenhard, W. & Schneider, W. (2006). ELFE 1-6 - Ein Leseverstaendnistest fuer Erst- bis Sechstklässler. Goettingen: Hogrefe.
5. The World Bank (2018). Mortality rate, infant (per 1,000 live births). Data Source available https://data.worldbank.org/indicator/SP.DYN.IMRT.IN (date of retrieval: 02/09/2018)
6. The World Bank (2018). Life expectancy at birth, total (years). Data Source World Development Indicators available https://data.worldbank.org/indicator/sp.dyn.le00.in (date of retrieval: 01/09/2018)

See Also

cNORM.GUI

Examples

```
# Model internal 'elfe' dataset with the default k = 4 regression on T scores
data.elfe <- prepareData(elfe)
model.elfe <- bestModel(data.elfe)
plotPercentiles(data.elfe, model.elfe)

# Show model fit of models with progressing number of predictors
printSubset(model.elfe)
plotSubset(model.elfe)

# Plot manifest and predicted values, plot series of percentile charts
plotRaw(data.elfe, model.elfe)
## Not run:
plotPercentileSeries(data.elfe, model.elfe)

## End(Not run)
```

```
# Additional tests: Check model assumptions
checkConsistency(model.elfe)
plotDerivative(model.elfe)

# Generate norm tables; predict values, here: grade 3.75 from T score 25
# to 75 and within the raw value range of this specific test (0 to 28)
normTable <- normTable(3.75, model.elfe, minNorm=25, maxNorm=75, step=0.5)
rawTable <- rawTable(3.75, model.elfe, minRaw = 0, maxRaw = 28, minNorm=25,
                     maxNorm=75)

# Predict a specific norm score
score <- predictNorm(raw = 21, A = 3.75,
                           model = model.elfe, minNorm=25, maxNorm=75)

# Semi-parametric modeling with Box Cox power transformation for grade 3.75
bcParameters <- boxcox(model.elfe, 3.75)
# Print L, M and S
bcParameters$lambdaBC
bcParameters$meanBC
bcParameters$sdBC
# Plot density function of box cox versus regression model
plotBoxCox(model.elfe, bcParameters, type=2)
```

`cnorm.cv` *Cross validation for term selection*

Description

This function helps in selecting the number of terms for the model by doing repeated cross validation with 80 percent of the data as training data and 20 percent as the validation data. The cases are drawn randomly but stratified by norm group. Successive models are retrieved with increasing number of terms and the RMSE of raw scores (fitted by the regression model) is plotted for the training, validation and the complete dataset. Additionally to this analysis on the raw score level, it is possible (default) to estimate the mean norm score reliability and crossfit measures. For this, please set the norms parameter to TRUE. Due to the high computational load when computing norm scores, it takes time to finish when doing repeated cv or comparing models up to the maximum number of terms. When using the cv = "full" option, the ranking is done for the test and validation dataset separately (always based on T scores), resulting in a complete cross validation. In order to only validate the modeling, you as well can use a pre-ranked data set with prepareData() already applied. In this case, the training and validation data is drawn from the already ranked data and the scores for the validation set should improve. It is however no independent test, as the ranking between both samples is interlinked. In the output, you will get RMSE for the raw score models, norm score R2 and delta R2 and the crossfit. For assessing, if a model over-fits the data and to what extent, we need cross-validation. We assumed that an overfitting occurred when a model captures more variance of the observed norm scores of the training sample compared to the captured variance of the norm scores of the validation sample. The overfit can therefore be described as:

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

A CROSSFIT higher than 1 is a sign of overfitting. Value lower than 1 indicate an underfit due to a suboptimal modeling procedure, i. e. the method may not have captured all the variance of the observed data it could possibly capture. Values around 1 are ideal, as long as the raw score RMSE is low and the norm score validation R2 reaches high levels. As a suggestion for real tests:

- Use visual inspection of the percentiles with plotPercentiles or plotPercentileSeries
- Combine the visual inspection of the percentiles with a repeated cross validation (e. g. 10 repetitions)
- Focus on low raw score RMSE, high norm score R2 in the validation dataset and avoid a number of terms with a high overfit (e. g. crossfit > 1.1).

Usage

```
cnorm.cv(data, repetitions = 1, norms = TRUE, min = 1, max = 12,
  cv = "full", pCutoff = 0.2, width = NA, raw = NA, group = NA,
  age = NA)
```

Arguments

data	data frame of norm sample with ranking, powers and interaction of L and A
repetitions	number of repetitions for cross validation
norms	determine norm score crossfit and R2 (if set to TRUE). The option is computationally intensive and duration increases with sample size, number of repetitions and maximum number of terms (max option).
min	Minimum number of terms to start from, default = 1
max	Maximum number of terms in model up to 2*k + k^2
cv	If set to full (default), the data is split into training and validation data and ranked afterwards, otherwise, a pre ranked dataset has to be provided, which is then split into train and validation (and thus only the modelling, but not the ranking is independent)
pCutoff	The function checks the stratification for unbalanced data sampling. It performs a t-test per group . pCutoff specifies the p-value per group that the test result has to reach at least. To minimize beta error, the value is set to .2 per default
width	If provided, ranking is done via rankBySlidingWindow, otherwise by group
raw	Name of the raw variable
group	Name of the grouping variable
age	Name of the age variable

Value

table with results per term number, including RMSE for raw scores in training, validationand complete sample, R2 for the norm scores and the crossfit measure (1 = ideal, <1 = underfit, >1 = overfit)

Examples

```
# plot cross validation RMSE by number of terms up to 9 with three repetitions
data <- prepareData()
cnorm.cv(data, 3, max = 7, norms = FALSE)
```

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 = 4, norm = NULL, age = NULL)
```

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.

Value

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

See Also

bestModel

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)
```

---

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
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

Examples

```
normData <- prepareData()
m <- bestModel(data = normData)
d <- derivationTable(6, m, 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
order	The degree of the derivate, default: 1

Value

The derived coefficients

Examples

```
normData <- prepareData()
m <- bestModel(normData)
derivedCoefficients <- derive(m)
```

`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

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
data.elfe <- prepareData(elfe)
model.elfe <- bestModel(data.elfe)
plotPercentiles(data.elfe, model.elfe)
```

`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
`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

Examples

```
normData <- prepareData()
m <- bestModel(data = normData)
getNormCurve(35, m)
```

life *Life expectancy at birth from 1960 to 2017*

Description

The data is available by the courtesy of the World Bank under Creative Commons Attribution 4.0 (CC-BY 4.0). It includes the life expectancy at birth on nation level from 1960 to 2017. The data has been converted to long data format, aggregates for groups of nations and missings have been deleted and a grouping variable with a broader scope spanning 4 years each has been added. It shows, that it can be better to reduce predictors. The model does not converge anymore after using 8 predictors and the optimal solution is achieved with four predictors, equaling R2=.9825.

Usage

```
life
```

Format

A data frame with 11182 rows and 4 variables:

Country The name of the country

year reference year of data collection

life the life expectancy at birth

group a grouping variable based on 'year' but with a lower resolution; spans intervals of 4 years each

Source

https://data.worldbank.org/indicator/sp.dyn.le00.in

References

The World Bank (2018). Life expectancy at birth, total (years). Data Source World Development Indicators available https://data.worldbank.org/indicator/sp.dyn.le00.in (date of retrieval: 01/09/2018)

Examples

```
## Not run:
# data preparation
data.life <- rankByGroup(life, raw="life")
data.life <- computePowers(data.life, age="year")

#determining best suiting model by plotting series
model.life <- bestModel(data.life, raw="life")
plotPercentileSeries(data.life, model.life, end=10)

# model with four predictors seems to work best
```

```
model2.life <- bestModel(data.life, raw="life", terms=4)

## End(Not run)
```

`mortality` *Mortality of infants per 1000 life birth from 1960 to 2017*

Description

The data is available by the courtesy of the World Bank under Creative Commons Attribution 4.0 (CC-BY 4.0). It includes the mortality rate of life birth per country from 1960 to 2017. The data has been converted to long data format, aggregates for groups of nations and missings have been deleted and a grouping variable with a broader scope spanning 4 years each has been added. It can be used for demonstrating intersecting percentile curves at bottom effects.

Usage

```
mortality
```

Format

A data frame with 9547 rows and 4 variables:

Country The name of the country

year reference year of data collection

mortality the mortality per 1000 life born children

group grouping variable based on 'year' with a lower resolution; spans intervals of 4 years each

Source

https://data.worldbank.org/indicator/SP.DYN.IMRT.IN

References

The World Bank (2018). Mortality rate, infant (per 1,000 live births). Data Source available https://data.worldbank.org/indicator/SP.DYN.IMRT.IN (date of retrieval: 02/09/2018)

Examples

```
## Not run:
# data preparation
data.mortality <- rankByGroup(mortality, raw="mortality")
data.mortality <- computePowers(data.mortality, age="year")

# modeling
model.mortality <- bestModel(data.mortality, raw="mortality")
plotSubset(model.mortality, type = 0)
plotPercentileSeries(data.mortality, model.mortality, end=9, percentiles = c(.1, .25, .5, .75, .9))

## End(Not run)
```

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 rang 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.

Usage

```
normTable(A, model, minNorm = NULL, maxNorm = NULL, minRaw = NULL,
  maxRaw = NULL, step = NULL)
```

Arguments

A	the age as single value or a vector of age values
model	The regression model
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

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

See Also

rawTable

Examples

```
normData <- prepareData()
m <- bestModel(data = normData)

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

# create list of norm tables
norms <- normTable(c(2.5, 3.5, 4.5), m,
```

```
  minNorm = 25, maxNorm = 75,
  step = 1, minRaw = 0, maxRaw = 26
)
```

---

plotBoxCox *Plot regression model versus box cox for a specific age*

Description

This plot can be used to compare, how well the regression data can be modeled via a Box Cox power transformation.

Usage

```
plotBoxCox(regressionModel, boxcoxParameters, minRaw = NULL,
  maxRaw = NULL, type = 0)
```

Arguments

regressionModel	The regression model from the 'bestModel' function
boxcoxParameters	The parameters from the box cox power transformation
minRaw	The lower bound of raw scores; must not fall below 0 due to restrictions of the box cox function
maxRaw	The upper bound of raw scores
type	Type of plot; 0 = Show percentiles as function of raw scores, 1 = Show raw scores as function of norm scores, 2 = Density plot

Value

data frame with fitted box cox and regression scores

See Also

boxcox

Examples

```
# Calculate model based on PPVT4 data
data <- prepareData(ppvt)
model <- bestModel(data)

# compute power function for a specific age, e. g. 9.2
bc <- boxcox(model, 9.2)

# plot results as a function of norm scores
```

```
plotBoxCox(model, bc, minRaw=0, maxRaw=228, type=1)

# plot density
plotBoxCox(model, bc, minRaw=0, maxRaw=228, type=2)
```

---

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

Description

The function plots the density curves based on the regression model against the actual percentiles from the raw data. As in 'plotNormCurves', please check for inconsistent curves, especially curves showing implausible shapes.

Usage

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

Arguments

model	The model from the bestModel function
minRaw	Lower bound of the raw score
maxRaw	Upper bound of the raw score
minNorm	Lower bound of the norm score
maxNorm	Upper bound of the norm score
group	Column of groups to plot

See Also

plotNormCurves, plotPercentiles

Examples

```
# Load example data set, compute model and plot results for age values 2, 4 and 6
normData <- prepareData()
m <- bestModel(data = normData)
plotDensity(m, group = c (2, 4, 6))
```

plotDerivative *Plot first order derivative of regression model*

Description

Plots the scores obtained via the first order derivative of the regression model in dependence of the norm score. 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. ATTENTION: plotDerivative is currently still incompatible with reversed raw score scales ('descent' option)

Usage

```
plotDerivative(model, minAge = NULL, maxAge = NULL, minNorm = NULL,
  maxNorm = NULL, stepAge = 0.2, stepNorm = 1, order = 1)
```

Arguments

model	The model from the bestModel function
minAge	Age to start with checking
maxAge	Upper end of the age check
minNorm	Lower end of the norm score range, in case of T scores, 25 might be good
maxNorm	Upper end of the norm score range, in case of T scores, 25 might be good
stepAge	Stepping parameter for the age check, usually 1 or 0.1; lower values indicate higher precision / closer checks
stepNorm	Stepping parameter for norm scores
order	Degree of the derivative (default = 1)

See Also

checkConsistency, bestModel, derive

Examples

```
# Load example data set, compute model and plot results
normData <- prepareData()
m <- bestModel(data = normData)
plotDerivative(m, minAge=2, maxAge=5, step=.2, minNorm=25, maxNorm=75, stepNorm=1)
```

plotNorm *Plot manifest and fitted norm scores*

Description

The 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 regression line. The computation of the standard error is based on Ooserhuis, van der Ark and Sijtsma (2016).

Usage

```
plotNorm(data, model, group = "", minNorm = NULL, maxNorm = NULL,
  type = 0)
```

Arguments

data	The raw data within a data.frame
model	The regression model
group	The grouping variable, use empty string "" for no group
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

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

Examples

```
# Load example data set, compute model and plot results
## Not run:
normData <- prepareData()
m <- bestModel(data = normData)
plotNorm(normData, m, group="group", minNorm=25, maxNorm=75)

## End(Not run)
```

plotNormCurves *Plot norm curves*

Description

The function plots the norm curves based on the regression model. Please check the function for inconsistent curves: The different curves should not intersect. Violations of this assumption are a strong 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. scores -3 <= x and x >= 3 In order to model these extreme scores, a large sample dataset is necessary.
2. Horizontal extrapolation: Taylor polynomials converge in a certain radius. Using the model scores 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.

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. checkConsistency and derivationPlot can be used to further inspect the model.

Usage

```
plotNormCurves(model, normList = c(30, 40, 50, 60, 70), minAge = NULL,
  maxAge = NULL, step = 0.1, minRaw = NULL, maxRaw = NULL)
```

Arguments

model	The model from the bestModel function
normList	Vector with norm scores to display
minAge	Age to start with checking
maxAge	Upper end of the age check
step	Stepping parameter for the age check, usually 1 or 0.1; lower scores indicate higher precision / closer checks
minRaw	Lower end of the raw score range, used for clipping implausible results (default = 0)
maxRaw	Upper end of the raw score range, used for clipping implausible results

See Also

checkConsistency, derivationPlot, plotPercentiles

Examples

```
# Load example data set, compute model and plot results
normData <- prepareData()
m <- bestModel(data = normData)
plotNormCurves(m, minAge=2, maxAge=5)
```

`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. Please consult help(quantile) and change the 'type' parameter accordingly.

Usage

```
plotPercentiles(data, 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, type = 7, title = NULL)
```

Arguments

`data`	The raw data including the percentiles and norm scores
`model`	The model from the bestModel function
`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)
`type`	The type parameter of the quantile function to estimate the percentiles of the raw data (default 7)
`title`	custom title for plot

See Also

plotNormCurves, plotPercentileSeries

Examples

```
# Load example data set, compute model and plot results
normData <- prepareData()
m <- bestModel(data = normData)
plotPercentiles(normData, m)
```

`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(data, model, start = 1, end = NULL,
  group = NULL, percentiles = c(0.025, 0.1, 0.25, 0.5, 0.75, 0.9,
  0.975), type = 7, filename = NULL)
```

Arguments

`data`	The raw data including the percentiles and norm scores
`model`	The model from the bestModel function
`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)
`type`	The type parameter of the quantile function to estimate the percentiles of the raw data (default 7)
`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

Examples

```
# Load example data set, compute model and plot results
normData <- prepareData(elfe)
model <- bestModel(data = normData)
plotPercentileSeries(normData, model, start=1, end=5, group="group")
```

---

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(data, model, group = NULL, raw = NULL, type = 0)
```

Arguments

data	The raw data within a data.frame
model	The regression model
group	The grouping variable
raw	The raw score variable
type	Type of display: 0 = plot manifest against fitted values, 1 = plot manifest against difference values

Examples

```
# Load example data set, compute model and plot results
normData <- prepareData()
m <- bestModel(data = normData)
plotRaw(normData, m, group="group")
```

plotSubset *Evaluate information criteria for regression model*

Description

Plots the information criterion - either Cp (default) or BIC - against the adjusted R square of the feature selection in the modeling process. Both BIC and Mallow's Cp are measures to avoid overfitting. Please choose the model that has a high information criterion, while modeling the original data as close as possible. R2 adjusted values of ~ .99 might work well, depending on your scenario. In other words: Look out for the elbow in the curve and choose th model where the information criterion begins to drop. Nonetheless, inspect the according model with `plotPercentiles(data, group)` to visually inspect the course of the percentiles. In the plot, Mallow's Cp is log transformed and the BIC is always highly negative. The R2 cutoff that was specified in the bestModel function is displayed as a dashed line.

Usage

```
plotSubset(model, type = 1)
```

Arguments

`model`	The regression model from the bestModel function
`type`	Type of chart with 0 = adjusted R2 by number of predictors, 1 = log transformed Mallow's Cp by adjusted R2, 2 = Bayesian Information Criterion (BIC) by adjusted R2 and 3 = Root Mean Square Error (RMSE) by number of predictors

See Also

bestModel, plotPercentiles, printSubset

Examples

```
normData <- prepareData()
m <- bestModel(data = normData)
plotSubset(m)
```

ppvt *Vocabulary development from 4 to 16*

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. The cleaned and stratified data is available on request.

Usage

```
ppvt
```

Format

A data frame with 5600 rows and 4 variables:

age the chronological age of the child

group the according age group, e.g. age group 4 consists of children age 3.5 to 4.5

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

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

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
data.ppvt <- rankBySlidingWindow(ppvt, age="age", width=1.5)
data.ppvt <- computePowers(data.ppvt, age="age")
model.ppvt <- bestModel(data.ppvt, R2 = .994)

# plot information function
plotSubset(model.ppvt, type=2)

# check model consistency
checkConsistency(model.ppvt)

# plot percentiles
plotPercentiles(data.ppvt, model.ppvt)

## End(Not run)
```

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

Description

In real test scenarios, usually the results are available as raw values, for which norm scores have to be looked up. This function conducts this reverse transformation via a numerical solution: A precise norm table is generated and the closest fitting norm score for a raw score is returned.

Usage

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

Arguments

raw	The raw value, either single numeric or list of values
A	the age, either single numeric or list of values
model	The regression model
minNorm	The lower bound of the norm score range
maxNorm	The upper bound of the norm score range

Value

The predicted norm score for a raw score, either single value or list of results

Examples

```
normData <- prepareData()
m <- bestModel(data = normData)

# return norm value for raw value 21 for grade 2, month 9
# Use 'as.list(normData$raw)' and 'as.list(normData$group)' for raw scores
# and age to calculate predicted norm values for original data.
specificNormValue <- predictNorm(raw = 21, A = 2.75, model = m, minNorm = 25, maxNorm = 75)
```

predictNormBC *Calculate the norm value for a given raw value based on the parametric box cox distribution*

Description

In addition to the numeric solution of the inversion of the regression function applied in 'predictNorm', this function can be used retrieving the norm scores at a specific age via the parametric box cox power transformation. Please provide the box cox parameters retrieved via the 'boxcox'-function and a raw value.

Usage

```
predictNormBC(boxcoxParameters, raw, scale = "percentile")
```

Arguments

boxcoxParameters	The parameters of the box cox power function, calculated via 'boxcox'
raw	The raw value (>0)
scale	type of norm scale, either T, IQ, z or percentile (= no transformation; default); a double vector with the mean and standard deviation can as well, be provided f. e. c(10, 3) for Wechsler scale index points

Value

the predicted raw value

References

Cole, T. J., & Green, P. J. (1992). Smoothing reference centile curves: the LMS method and penalized likelihood. Statistics in medicine, 11(10), 1305-1319.

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. Journal of the Royal Statistical Society. Series B (Methodological), 211-252.

See Also

boxcox, predictNorm, predictRawBC

Examples

```
# model sample data set
model <- bestModel(prepareData())

# fitting scores of regression model box cox power function at specific age and retrieving
# the parameters for the box cox power function
bcParameters <- boxcox(model, 3)

# predict norm value for raw value 15 at age 3 based on the regression model and via box cox
predictNormBC(bcParameters, 15, scale="T")
predictNorm(15, 3, model, minNorm=25, maxNorm=75)
```

predictRaw *Predict single raw value*

Description

Most elementary function to predict raw score based on Location (L, T score), Age (grouping variable) and the coefficients from a regression model. WARNING! This function, and all functions depending on it, only works with regression functions including L, A and interactions. Manually adding predictors to bestModel via the predictors parameter is currently incompatible. In that case, and if you are primarily interested on fitting a complete data set, rather user the predict function of the stats:lm package on the ideal model solution. You than have to provide a prepared data frame with the according input variables.

Usage

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

Arguments

norm	The norm score, e. g. a specific T score
age	The age value
coefficients	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

Examples

```
# Prediction of single scores
normData <- prepareData()
m <- bestModel(data = normData)
predictRaw(35, 3.5, m$coefficients)

# Fitting complete data sets
fitted.values <- predict(m)

# break up contribution of each predictor variable
fitted.partial <- predict(m, type = "terms")
```

predictRawBC *Calculate the raw score for a given percentile based on the parametric box cox distribution*

Description

In addition of the numeric solution to the regression function on 'predictRaw', this function can be used retrieving the raw values at a specific age via the parametric box cox power transformation. Please provide the box cox parameters retrieved via the 'boxcox'-function and a percentile.

Usage

```
predictRawBC(boxcoxParameters, percentile)
```

Arguments

boxcoxParameters	The parameters of the box cox power function, calculated via 'boxcox'
percentile	The percentile (ranging from >0 to <1)

Value

the predicted raw value

References

Cole, T. J., & Green, P. J. (1992). Smoothing reference percentile curves: the LMS method and penalized likelihood. Statistics in medicine, 11(10), 1305-1319.

Box, G. E., & Cox, D. R. (1964). An analysis of transformations. Journal of the Royal Statistical Society. Series B (Methodological), 211-252.

See Also

boxcox, predictRaw

Examples

```
## Not run:
# model sample data set
model <- bestModel(prepareData())

# fitting scores of regression model box cox power function at specific age and retrieving
# the parameters for the box cox power function
bcParameters <- boxcox(model, 3)

# define percentile and according t value
percentile <- .4
tValue <- qnorm(percentile)*10 + 50
```

```
# predict raw value based on the regression model and via box cox
predictRawBC(bcParameters, percentile)
predictRaw(tValue, 3, model$coefficients)

## End(Not run)
```

`prepareData` *Set up example dataset and compute model*

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 the best fitting model is determined, based on all default parameters.

Usage

```
prepareData(data = NULL, group = "group", raw = "raw",
  age = "group", width = NA, scale = "T")
```

Arguments

`data`	data.frame with a grouping variable named 'group' and a raw score variable named 'raw'. In case no object is provided, cNORM uses the inbuilt sample data to demonstrate the procedure.
`group`	grouping variable in the data, e. g. age groups, grades ... Setting group = FALSE deactivates modelling 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"
`width`	if a width is provided, the function switches to rankBySlidingWindow to determine the observed raw scores, otherwise, ranking is done by group (default)
`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

Value

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

Examples

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

# variable names can be specified as well, here with the BMI data included in the package
data.bmi <- prepareData(CDC, group="group", raw="bmi", age="age")

# modelling 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)
```

---

printSubset *Convenience 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

```
printSubset(model)
```

Arguments

model	The model from the 'bestModel' function

Value

A table with information criteria

Examples

```
model <- bestModel(prepareData())
printSubset(model)
```

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(model, minAge = NULL, maxAge = NULL, minNorm = NULL,
  maxNorm = NULL, digits = 3)
```

Arguments

model	The regression model
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

Value

the report

Examples

```
normData <- prepareData()
m <- bestModel(normData)
print(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, group = "group", raw = "raw", method = 4,
  scale = "T", descend = FALSE, descriptives = TRUE)
```

Arguments

data	data.frame with norm sample data
group	name of the grouping variable (default 'group'), e. g. grade, setting group to FALSE cancels grouping (data is treated as one group)
raw	name of the raw value variable (default 'raw')
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

Value

the dataset with the percentiles and norm scales per group

See Also

rankBySlidingWindow, computePowers

Examples

```
# Transformation with default parameters: RandIt and converting to T scores
data.elfe <- rankByGroup(elfe, group = "group")

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

# cNORM can as well be used for conventional norming
# The group variable has to be set to NULL when ranking the group in this case
d <- rankByGroup(elfe, raw="raw", group=FALSE)
d <- computePowers(d)
m <- bestModel(d)
rawTable(0, model = m) # please use an arbitrary value for age when generating the tables
```

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

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, age = "age", raw = "raw", width,
  method = 4, scale = "T", descend = FALSE, descriptives = TRUE,
  nGroup = 0, group = NA)
```

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')
`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)

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

See Also

rankByGroup, computePowers

Examples

```
## Not run:
# Transformation using a sliding window
data.elfe2 <- rankBySlidingWindow(elfe, 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.

Usage

```
rawTable(A, model, minRaw = NULL, maxRaw = NULL, minNorm = NULL,
  maxNorm = NULL, step = 1)
```

Arguments

A	the age, either single value or vector with age values
model	The regression model
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)

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

See Also

normTable

Examples

```
normData <- prepareData()
m <- bestModel(data = normData)
# generate a norm table for the raw value range from 0 to 28 for month 7 of grade 3
table <- rawTable(3 + 7 / 12, m, minRaw = 0, maxRaw = 28)

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

---

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
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

Examples

```
normData <- prepareData()
model <- bestModel(normData)
regressionFunction(model)
```

`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)
```

Index