

Package ‘MiscPsycho’

February 14, 2012

Type Package

Title Miscellaneous Psychometric Analyses

Version 1.6

Date 2010-04-19

Author Harold C. Doran

Depends statmod

Maintainer Harold C. Doran <hdoran@air.org>

Description Miscellaneous functions for psychometric problems

License GPL

LazyLoad yes

Repository CRAN

Date/Publication 2010-04-21 13:15:31

R topics documented:

MiscPsycho-package	2
alpha	3
cheat	4
choose.M	6
class.acc	7
classical	8
irt.ability	9
jml	11
mml	12
plaus.val	14
posterior	15
scoreCon	16
simRasch	17

SL	17
SLderivs	19
SSI	19
stringMatch	21
stringProbs	22
theta.max	23
wrongProb	24

Index	26
--------------	-----------

MiscPsycho-package	<i>Miscellaneous Psychometrics</i>
--------------------	------------------------------------

Description

MiscPsycho is a package containing functions that may be useful to applied psychometricians

Details

Package:	MiscPsycho
Type:	Package
Version:	1.6
Date:	2010-04-20
License:	GPL
LazyLoad:	yes

Author(s)

Maintainer: Harold C. Doran <hdoran@air.org>

Examples

```
## Not run:
params <- list("3pl" = list(a = c(1,1), b = c(0, 1), c = c(0,0)),
              "gpcm" = list(a = c(1,1), d = list(item1 = c(0,1,2,3,4), item2 = c(0, .5,1, 1.5))))
irt.ability(c(0,1,2,2), params, ind.dichot = c(1,2), method = "MLE")
irt.ability(c(0,1,2,2), params, ind.dichot = c(1,2), method = "MAP")
irt.ability(c(0,1,2,2), params, ind.dichot = c(1,2), method = "EAP")
## End(Not run)
```

alpha	<i>Cronbach's Coefficient Alpha</i>
-------	-------------------------------------

Description

Computes Cronbach's alpha. This reduces to KR-20 when the columns of the data matrix are dichotomous.

Usage

```
alpha(...)
## Default S3 method:
alpha(dat, ...)
## S3 method for class 'formula'
alpha(formula, data, na.action, subset, ...)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
data	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>alpha</code> is called.
dat	A data frame or matrix with item responses. Implemented only for the <code>alpha.default</code> method.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to <code>getOption("na.action")</code> .
subset	an optional vector specifying a subset of observations to be used.
...	Not implemented

Details

The conditional alpha is accesible via the summary method for objects of class alpha

Value

A list with class "alpha" containing the following components:

alpha	coefficient alpha
numItems	the number of test items used in the computation
condAlpha	The alpha that would be realized if the item were excluded

Author(s)

Harold C. Doran

Examples

```

set.seed(1234)
tmp <- data.frame(item1 = sample(c(0,1), 20, replace=TRUE), item2 = sample(c(0,1), 20, replace=TRUE),
item3 = sample(c(0,1), 20, replace=TRUE), item4 = sample(c(0,1), 20, replace=TRUE), item5 = sample(c(0,1), 20, repla

## Formula interface
fm1 <- alpha(~ item1 + item2 + item3 + item4 + item5, data = tmp)
summary(fm1)
coef(fm1)

## Default interface
fm1 <- alpha(tmp)
summary(fm1)

```

 cheat

Method to detect excessive similarity in student test responses

Description

Examines all pairwise comparisons within a group to assess the degree to which response patterns between individuals are too similar to have occurred from random chance alone.

Usage

```

cheat(...)
## Default S3 method:
cheat(dat, key, wrongChoice, alpha = .01, rfa = c('nr', 'uni', 'bsct'), bonf = c('yes', 'no'), con = 1e-12)
## S3 method for class 'formula'
cheat(formula, data, na.action, subset, key, wrongChoice, ...)

```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
data	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>cheat</code> is called.
dat	A data frame or matrix with item responses. Implemented only for the default method.

na.action	a function which indicates what should happen when the data contain NAs. Defaults to <code>getOption("na.action")</code> .
subset	an optional vector specifying a subset of observations to be used.
key	a numeric vector containing the correct responses to each test item
rfa	the Root Finding Algorithm used. Options include <code>nr</code> for newton-raphson, <code>uni</code> to use R's internal uniroot function, or <code>bsct</code> for the bisection method.
alpha	Level of significance
bonf	Option to choose bonferonni adjustment to the alpha
con	Tolerance for root finding algorithms
lower	the lower end points of the interval to be searched for the bisection and uniroot functions
upper	the upper end points of the interval to be searched for the bisection and uniroot functions
wrongChoice	a list containing the probability of choosing each incorrect response option. See wrongProb for details
...	Not implemented

Details

The dataframe must be organized with examinees as rows and their responses in columns

Value

A list with class "cheat" containing the following components:

Number of Possible Cheating Pairs	the number of individuals with similar response patterns)
Possible Cheating Pairs	the identified individuals. <code>S28:32</code> denotes that individuals in rows 28 and 32 have similar response patterns
Number of Exact Matches	Number of observed exact matches between the two individuals compared
Observed Z Values	the statistical result comparing number of observed exact matches to the expected
Critical Z	the z value used as the threshold
Expected Number of Matches	the expected number of matches between the two individuals compared

Author(s)

Harold C. Doran

References

Wesolowsky, G.E (2000). Detecting excessive similarity in answers on multiple choice exams. *Journal of Applied Statistics* (27)7

Examples

```

## Simulate data
NumStu <- 30
NumItems <- 50
dat <- matrix(0, nrow=NumStu, ncol=NumItems)
set.seed(1234)
for(i in 1:NumStu){
  dat[i,] <- sample(1:4, NumItems, replace=TRUE)
}
dat <- data.frame(dat)

## Add in explicit answer copying
dat[(NumStu+1),] <- dat[NumStu,]
dat[(NumStu+2),] <- c(dat[(NumStu-1), 1:25], dat[(NumStu-2), 26:50 ])

## Answer Key
set.seed(1234)
key <- sample(1:4, NumItems, replace=TRUE)

## Formula interface
ff <- as.formula(paste('~', paste( names(dat), collapse= "+")))
## See wrongProb help page
mm <- wrongProb(ff, data = dat, key = key)
(result <- cheat(ff, data = dat, key = key, wrongChoice = mm))
summary(result)

## Default interface
(result <- cheat(dat, key = key, wrongChoice = mm))

```

choose.M

Constant Tuning

Description

Find a Constant M that Improves the Acceptance Rate

Usage

```
choose.M(x, theta, params, ind.dichot, ...)
```

Arguments

x	A vector of observed responses to all items
theta	An arbitrarily chosen value of theta
params	Item parameters organized as a list of lists
ind.dichot	An indicator denoting the position the multiple choice items in x
...	Additional arguments passed to posterior

Details

This is a function that can be used to supplement the `plaus.val` function. See the MP vignette for details

Value

Returns a numeric value that can be the constant M in the `plaus.val` function

Author(s)

Harold C. Doran

Examples

```
params <- list("3pl" = list(a = c(1,1), b = c(0, 1), c = c(0,0)),
"GPCM" = list(a = c(1,1), d = list(item1 = c(0,1,2,3,4), item2 = c(0,.5,1, 1.5))))
choose.M(x = c(0,1,2,2), theta = -5, params = params, ind.dichot=c(1,2))
```

class.acc

Classification Accuracy Statistic: Integration over the Posterior

Description

Computes the probability that individual i has a true score above (or below) cutscore m . In other words, computes the proportion of the posterior distribution that falls above (or below) a cutpoint.

Usage

```
class.acc(x, prof_cut, params, ind.dichot, aboveC = FALSE, control=list())
```

Arguments

<code>x</code>	A vector of item responses
<code>prof_cut</code>	Cut score
<code>params</code>	Item parameters organized as a list of lists
<code>ind.dichot</code>	Indicator denoting which items in the vector <code>x</code> are dichotomous
<code>aboveC</code>	Test for above or below a cut score
<code>control</code>	A list of control parameters, <ul style="list-style-type: none"> D A constant usually fixed at 1.7 to bring the logistic function into coincidence with the probit mu Mean of the prior distribution sigma Standard deviation of the prior distribution Q Number of quadrature points used in the Gauss-Hermite approximation

Value

prob Returns the probability that individual i has a true score above (or below) the cut score specified

Author(s)

Harold C. Doran

Examples

```
a <- c(1.45, 1.84, 2.55, 2.27, 3.68, 4.07, 2.26, 1.87, 2.19, 1.33)
b <- c(-.6, -.82, -1.6, -.87, -1.41, -1.33, -1.16, -.11, -.64, -1.23)
params <- list("3pl" = list(a = a, b = b, c = rep(0, 10)),
              "gpcm" = NULL)
x <- c(rep(0,9),1)
class.acc(x, prof_cut = 0, params, ind.dichot = c(1:10), aboveC=TRUE)
```

classical

Classical Item Analysis

Description

Provides p-values, standard errors of p-values, and point-biserial correlations for multiple choice test items. The standard errors can be design-consistent to reflect the sampling design

Usage

```
classical(...)
## Default S3 method:
classical(data, designSE = FALSE, group, na.rm = TRUE, use = 'everything', ...)
## S3 method for class 'formula'
classical(formula, data, na.action, subset, designSE = FALSE, group, na.rm = TRUE, use = 'everything', .
```

Arguments

formula an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.

data an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `classical` is called.

na.action a function which indicates what should happen when the data contain NAs. Defaults to `getOption("na.action")`.

na.rm logical. Should missing values be removed?

use See `cor` function

subset	an optional vector specifying a subset of observations to be used.
designSE	logical. If TRUE the function returns design-consistent standard errors to capture clustering in the population
group	the grouping variable. Used only if designSE = TRUE
...	not implemented

Author(s)

Harold C. Doran

Examples

```
xx <- simRasch(200,10)
itemDat <- xx$dat
itemDat$group <- gl(10,20)
(aa <- classical(~V1+V2+V3+V4+V5+V6+V7+V8+V9+V10, data = itemDat, design = TRUE, group = group))
summary(aa)
classical(itemDat[,1:10])
```

irt.ability	<i>IRT Ability Estimates</i>
-------------	------------------------------

Description

Returns the MLE, MAP, or the EAP given a set of item parameters

Usage

```
irt.ability(x, params, ind.dichot = NULL, std.err = FALSE, method = c("MLE", "MAP", "EAP"), control = li
```

Arguments

x	a numeric vector of observed item responses.
params	item parameters organized as a list of lists. For more info see the Details section and the Examples below.
ind.dichot	a numeric vector that denotes which items in x are multiple choice items. See the Examples below.
std.err	logical; if TRUE it returns the standard error.
method	a character string indicating which method to use; available options are Maximum Likelihood Estimation, Maximum A Posteriori and Expected A Posteriori.
control	a list of control parameters, D a constant usually fixed at 1.7 to bring the logistic function into coincidence with the probit. mu mean of the normal prior distribution. sigma standard deviation of the normal prior distribution. Q number of quadrature points used in the Gauss-Hermite approximation. start_val a starting value for the optimization routine.

Details

The function can be used for a mixture of dichotomously and polytomously scored items. For dichotomous items the Birnbaum's three parameter model is assumed for which a denotes the discrimination parameter, b the difficulty parameter and c the guessing parameter. For polytomous items the generalized partial credit model is assumed for which a denotes the discrimination parameter and d the threshold parameters. See the **Examples** below.

Value

a numeric value. If `std.err = TRUE`, it has also the attribute `'std.err'` that contains the estimated standard error for the estimated ability.

Author(s)

Harold C. Doran and Dimitris Rizopoulos

References

Thissen, D and Wainer, H. (2001). Test Scoring. Lawrence Erlbaum.

Examples

```
## all mix
params <- list("3pl" = list(a = c(1,1), b = c(0, 1), c = c(0,0)),
              "gpcm" = list(a = c(1,1), d = list(item1 = c(0,1,2,3,4), item2 = c(0, .5,1, 1.5))))
irt.ability(c(0,1,2,2), params, ind.dichot = c(1,2), method = "MLE")
irt.ability(c(0,1,2,2), params, ind.dichot = c(1,2), method = "MAP")
irt.ability(c(0,1,2,2), params, ind.dichot = c(1,2), method = "EAP")

## under the logit link
irt.ability(c(0,1,2,2), params, ind.dichot = c(1,2), method = "MLE", control = list(D = 1))
irt.ability(c(0,1,2,2), params, ind.dichot = c(1,2), method = "MAP", control = list(D = 1))
irt.ability(c(0,1,2,2), params, ind.dichot = c(1,2), method = "EAP", control = list(D = 1))

## if all are dichotomous
params <- list("3pl" = list(a = c(1,1), b = c(0, 1), c = c(0,0)), "gpcm" = NULL)
irt.ability(c(0,1), params, ind.dichot = c(1,2), method = "MLE")
irt.ability(c(0,1), params, ind.dichot = c(1,2), method = "MAP")
irt.ability(c(0,1), params, ind.dichot = c(1,2), method = "EAP")

## if all are polytomous
params <- list("3pl" = NULL, "gpcm" = list(a = c(1,1), d = list(item1 = c(0,1,2,3,4), item2 = c(0, .5,1, 1.5))))
irt.ability(c(2,3), params, method = "MLE")
irt.ability(c(2,3), params, method = "MAP")
irt.ability(c(2,3), params, method = "EAP")

## With standard error
irt.ability(c(2,3), params, method = "MLE", std.err = TRUE)
irt.ability(c(2,3), params, method = "MAP", std.err = TRUE)
irt.ability(c(2,3), params, method = "EAP", std.err = TRUE)
```

```
## From Test Scoring (Thissen) Page 115

params <- list("3pl" = list(a = c(1,2), b = c(0, 1), c = c(0,0)),
              "gpcm" = NULL)
irt.ability(c(0,1), params, ind.dichot = c(1,2), method = "MLE", control=list(D=1))

a <- c(1.45, 1.84, 2.55, 2.27, 3.68, 4.07, 2.26, 1.87, 2.19, 1.33)
b <- c(-.6, -.82, -1.6, -.87, -1.41, -1.33, -1.16, -.11, -.64, -1.23)
params <- list("3pl" = list(a = a, b = b, c = rep(0, 10)),
              "gpcm" = NULL)
x <- c(rep(0,9),1)
irt.ability(x, params, ind.dichot = c(1:10), method = "EAP", control=list(D=1))
```

Description

JML is used to estimate item parameters for the Rasch model.

Usage

```
jml(...)
## Default S3 method:
jml(dat, con = 1e-3, bias=FALSE, ...)
## S3 method for class 'formula'
jml(formula, data, na.action, subset, con = 1e-3, bias = FALSE, ...)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
data	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>jml</code> is called.
dat	A data frame or matrix with item responses. Implemented only for the <code>jml.default</code> method.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to <code>getOption("na.action")</code> .
subset	an optional vector specifying a subset of observations to be used.
con	Convergence criterion
bias	Implements the correction for bias as described by Wright and Stone
...	Not implemented

Details

Models for `jml` are specified symbolically. A typical model has the form `~item1 + item2` where the terms to the right of the `~` are the columns of the data matrix containing the binary item responses.

Value

A list with class "jml" containing the following components:

Estimate	the value of Rasch item parameter (b-value)
Std.Error	the standard error of the item parameter
Infit	The Rasch infit statistic
Outfit	The Rasch outfit statistic
Iterations	the number of Newton-Raphson iterations used
model.frame	the data matrix used for estimating item parameters. In JML estimation, individuals with all items correct and all items incorrect cannot be used in the calibration. Hence, they are dropped from the original data matrix provided by the data argument.

Author(s)

Harold Doran

Examples

```
set.seed(1234)
tmp <- data.frame(item1 = sample(c(0,1), 20, replace=TRUE), item2 = sample(c(0,1), 20, replace=TRUE),
item3 = sample(c(0,1), 20, replace=TRUE), item4 = sample(c(0,1), 20, replace=TRUE), item5 = sample(c(0,1), 20, repla

## Formula interface
fm1 <- jml(~ item1 + item2 + item3 + item4 + item5, data = tmp)
summary(fm1)
coef(fm1)
plot(fm1)

## Default interface
fm1 <- jml(tmp)
summary(fm1)
```

Description

MML is used to estimate the population parameters (i.e., mean and variance) for an IRT model. Here, we treat item parameters as known and maximize the marginal distribution to obtain parameter estimates for the (normal) population distribution

Usage

```

mml(...)
## Default S3 method:
mml(data, Q = 20, params, ...)
## S3 method for class 'formula'
mml(formula, data, na.action, subset, Q = 20, params, ...)

```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
data	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>mml</code> is called.
params	item parameters organized as a list of lists. For more info see the irt.ability function help page.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to <code>getOption("na.action")</code> .
subset	an optional vector specifying a subset of observations to be used.
Q	Number of quadrature points used for Gauss-Hermite quadrature
...	A list of control parameters, D a constant usually fixed at 1.7 to bring the logistic function into coincidence with the probit. startVal Optional starting values to use for parameters.

Details

Models for `mml` are specified symbolically. A typical model has the form `~item1 + item2` where the terms to the right of the `~` are the columns of the data matrix containing the binary item responses.

Value

A list with class "mml" containing the following components:

Estimate	the estimates of the population parameters
Std.Error	the standard error of the parameters

Author(s)

Harold Doran

Examples

```

N <- 5
aa <- simRasch(200,N, mu=0, sigma=1)
tmp <- aa$data
b <- aa$gen
params <- list("3pl" = list(a = rep(1,N), b = b, c = rep(0,N)), "gpcm" = NULL)
colnames(tmp) <- paste('item', 1:5, sep='')

## Formula interface
fm1 <- mml(~ item1 + item2 + item3 + item4 + item5, data = tmp, params = params, control=list(D=1, startVal=c(0,1)))
summary(fm1)
coef(fm1)

## Default interface
fm1 <- mml(tmp, params = params, control=list(D=1, startVal=c(0,1)))
summary(fm1)

```

plaus.val

IRT Plausible Values

Description

Draw random samples from the posterior density of an IRT Model

Usage

```
plaus.val(x, params, PV = 5, ind.dichot, M = 3, max = 5000, ...)
```

Arguments

x	Vector of item responses
params	Item parameters organized as a list of lists
PV	Number of plausible values desired
ind.dichot	Denotes which items in x are multiple choice
M	Constant
max	Maximum length of vector for storing results
...	Further arguments passed to posterior

Details

Uses Rejection Sampling to generate random samples from the IRT posterior density

Value

Returns random samples from the posterior density of an IRT model

Author(s)

Harold C. Doran

Examples

```
# all mix
params <- list("3pl" = list(a = c(1,1), b = c(0, 1), c = c(0,0)),
              "gpcm" = list(a = c(1,1), d = list(item1 = c(0,1,2,3,4), item2 = c(0,.5,1, 1.5))))

plaus.val(x = c(0,1,2,2), params = params, ind.dichot = c(1,2))
plaus.val(x = c(0,1,2,2), params = params, ind.dichot = c(1,2), control=list(mu=2, sigma=3))
```

posterior

*IRT Posterior Density***Description**

Compute the density of theta

Usage

```
posterior(x, theta, params, ind.dichot, control = list())
```

Arguments

<code>x</code>	A vector of observed item responses
<code>theta</code>	Ability estimate
<code>params</code>	Item parameters organized as a list of lists
<code>ind.dichot</code>	Indicator denoting which items in <code>x</code> are multiple choice
<code>control</code>	A list of control parameters, D A constant usually fixed at 1.7 to bring the logistic function into coincidence with the probit mu Mean of the prior distribution sigma Standard deviation of the prior distribution Q Number of quadrature points used in the Gauss-Hermite approximation

Author(s)

Harold C. Doran

Examples

```
params <- list("3pl" = list(a = c(1,1), b = c(0, 1), c = c(0,0)),
              "gpcm" = list(a = c(1,1), d = list(item1 = c(0,1,2,3,4), item2 = c(0,.5,1, 1.5))))
posterior(x = c(0,1,2,2), theta = 1, params = params, ind.dichot=c(1,2))
```

scoreCon	<i>Function to create a score conversion table from Rasch item parameters</i>
----------	---

Description

Creates a score conversion table using the estimated item parameters

Usage

```
scoreCon(b_vector)
## Default S3 method:
scoreCon(b_vector)
```

Arguments

b_vector vector of estimated item parameters

Value

A list with class "scoreCon" containing the following components:

Estimate	the maximum likelihood estimate (MLE, or ability estimate) conditional on the item parameters
Std.Error	the estimated standard error of the ability estimate
Raw.Score	the associated raw test score. Since Rasch does not use pattern scoring, all raw scores have the same MLE.

Note

Scores cannot be provided for individuals with 0 correct or all incorrect because the likelihood function is unbounded in these cases. Other scoring methods, such as the EAP or MAP are available for these scenarios. See [irt.ability](#) for more general scoring methods

Author(s)

Harold Doran

Examples

```
set.seed(1234)
tmp <- data.frame(item1 = sample(c(0,1), 20, replace=TRUE), item2 = sample(c(0,1), 20, replace=TRUE),
item3 = sample(c(0,1), 20, replace=TRUE), item4 = sample(c(0,1), 20, replace=TRUE), item5 = sample(c(0,1), 20, repla

## Estimate item parameters from JML (or provide them as a numeric vector)
fm1 <- jml(~ item1 + item2 + item3 + item4 + item5, data = tmp)
summary(scoreCon(coef(fm1)))
```

`simRasch`*Rasch Simulator*

Description

Simulates item response data for the Rasch model

Usage

```
simRasch(Nt, Nb, mu = 0, sigma = 1)
```

Arguments

<code>Nt</code>	Number of individuals
<code>Nb</code>	Number of items
<code>mu</code>	Mean of person distribution
<code>sigma</code>	Standard deviation of person distribution

Value

<code>data</code>	The simulated response data
<code>generating.values</code>	The true difficulty parameters used to generate the data
<code>theta</code>	The true ability estimates used to generate the data

Author(s)

Harold C. Doran

Examples

```
simRasch(200, 10) # 200 persons 10 items
```

`SL`*Stocking Lord Equating Procedure*

Description

This function estimates the A and B parameters (intercept and slope) such that two tests with common items can be linked. Common items must be multiple choice items based on the 3-parameter logistic model.

Usage

```
SL(...)
## Default S3 method:
SL(params1, params2, con = 1e-3, ...)
```

Arguments

params1	item parameters of the common items used for linking organized as a list of lists for test A. See irt.ability for more details.
params2	item parameters of the common items used for linking organized as a list of lists for test B. See irt.ability for more details.
con	Criterion for convergence
...	a list of control parameters. These are actually passed from the function <code>SLderivs</code> , D a constant usually fixed at 1.7 to bring the logistic function into coincidence with the probit. mu mean of the normal prior distribution. sigma standard deviation of the normal prior distribution. Q number of quadrature points used in the Gauss-Hermite approximation.

Value

A list with class "SL" containing the following components:

A.Parameter	The Stocking-Lord intercept
B.Parameter	The Stocking-Lord slope
A Parameter	Mean/Sigma intercept used as the starting values
B Parameter	Mean/Sigma slope used as the starting values

Author(s)

Harold Doran

References

Kolen, M.J., Brennan, R.L. (2004). Test equating, scaling, and linking. Second Edition. Springer.

Examples

```
## This is 3PL and are values from page 171 of Kolen and Brennan
params1 <- list("3pl" = list(a = c(.4, 1.7, 1.2), b = c(-1.1, .9, 2.2), c = c(.1, .2, .1)), "gpcm" = NULL)
params2 <- list("3pl" = list(a = c(.5, 1.6, 1), b = c(-1.5, .5, 2), c = c(.1, .2, .1)), "gpcm" = NULL)
SL(params1, params2, control=list(Q=30, mu=0, sigma=1))
```

 SLderivs *Stocking-Lord Derivatives*

Description

This function computes the gradient vector and the hessian needed for the SL function to operate

Usage

```
SLderivs(params1, params2, A = 1, B = 0, control = list())
```

Arguments

params1	item parameters of the common items used for linking organized as a list of lists for test A
params2	item parameters of the common items used for linking organized as a list of lists for test B
A	The A parameter for the SL transformation
B	The B parameter for the SL transformation
control	See function SL for details on control elements

Author(s)

Harold C. Doran

Examples

```
## Not run
## SLderivs(params1, params2, control=list(Q=10), A = A, B = B)
```

 SSI *Similar Student Index*

Description

The similar student index uses a K nearest neighbor algorithm to generate a set of conditional norms for the outcome variable. The conditional norm is constructed on the basis of the K students in the data most like student i who are used as the comparison set

Usage

```
SSI(...)
## Default S3 method:
SSI(mf, y, k, ...)
## S3 method for class 'formula'
SSI(formula, data, id, k, na.action, subset, ...)
```

Arguments

<code>formula</code>	a formula of the form <code>lhs ~ rhs</code> where <code>lhs</code> is a numeric variable giving the data values and <code>rhs</code> also numeric variables giving the conditioning variables used to identify the nearest neighbor.
<code>data</code>	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>SSI</code> is called.
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. Defaults to <code>getOption("na.action")</code> .
<code>subset</code>	an optional vector specifying a subset of observations to be used.
<code>id</code>	the individual (student) id identifying records in the data
<code>k</code>	the number of nearest neighbors to choose. <code>k</code> cannot be larger than the total number of pairwise comparisons in the data.
<code>mf</code>	a model frame with the variables used for conditioning. Only implemented for the default method.
<code>y</code>	the numeric outcome variable. Only implemented for the default method
<code>...</code>	Not implemented

Details

Implementation of the K nearest neighbor method is based on the euclidean distance metric. Because the process identifies the k nearest neighbors for each record in the data, the process can be relatively slow, executing in $O(n^2 \log n)$

Value

A list with class "SSI" containing the following components:

<code>Zscore</code>	the conditional z-score for each record in the data)
<code>percentile</code>	the conditional percentile for each record in the data
<code>ID</code>	the individual's record id
<code>Iterations</code>	the number of Newton-Raphson iterations used
<code>model.frame</code>	the data matrix used for estimating the conditional norms. This data frame can differ from the original data depending on the use of <code>na.action</code> .

Author(s)

Harold Doran

Examples

```
## Generate sample data
## construct a norm for the math score based on the k = 20
## other individuals in the data most like student i.
## readScore and scienceScore are used as the conditioning variables
## to compute the euclidean norm.
set.seed(1234)
tmp <- data.frame(ID = 1:100, mathScore = rnorm(100), readScore = rnorm(100), scienceScore = rnorm(100))
(result <- SSI(mathScore ~ readScore + scienceScore, tmp, k = 20, id=ID, na.action = na.omit))
summary(result)
str(result)
head(result$model.frame)
```

stringMatch

Implementation of the Levenshtein Algorithm

Description

Function to compare the similarity of two different character strings

Usage

```
stringMatch(string.1, string.2, normalize = c("YES", "NO"), penalty = 1, case.sensitive = FALSE)
```

Arguments

string.1	The first character string
string.2	The second character string
normalize	a character string indicating which method to use; if normalize = 'YES', then the edit distance is normalized to fall in the interval [0,1]
penalty	The edit cost
case.sensitive	logical; if TRUE, then a penalty occurs for differences in case of a character

Value

a numeric value. If normalize = 'YES', then the edit distance is normalized to fall in the interval [0,1]. Else, the Levenshtein edit distance is returned.

Author(s)

Harold C. Doram

References

http://en.wikipedia.org/wiki/Levenshtein_distance

Examples

```
## Return edit distance
stringMatch('William Clinton', 'Bill Clinton', normalize='NO')

## Return normalized edit distance
stringMatch('William Clinton', 'Bill Clinton', normalize='YES')

## Ignore differences in case
stringMatch('Bill Clinton', 'bill Clinton', normalize='YES', case.sensitive = FALSE)

## Do not ignore differences in case
stringMatch('Bill Clinton', 'bill Clinton', normalize='YES', case.sensitive = TRUE)
```

stringProbs	<i>Function to determine the probability of a particular normalized edit distance</i>
-------------	---

Description

Given a large database of character strings to be compared, this function takes a random sample, without replacement, from the first character string, and compares it to all other string.2 characters.

Usage

```
stringProbs(dat, N = 10)
```

Arguments

dat	A data frame with two columns. Column 1 contains the first string and column 2 contains the second.
N	The number of string.1 character strings to sample.

Details

The argument N can be equal to the total number of rows in the dataframe, but it cannot be larger.

Value

A dataframe with the cumulative probability of each normalized edit distance data

Author(s)

Harold C. Doran

References

See Levenshtein documentation distributed with this package

Examples

```
dat <- data.frame(fname1 = c('Joseph McCall', 'Paul Jones', 'Larry Everett', 'Sam Thompson', 'Sally Fields', 'Doug  
fname2 = c('Joe McCall', 'Paul Jones', 'Barry Everett', 'Samuel Thompson', 'Sally Fields', 'Douglas Carter', 'Willi  
  
## Randomly sample five names from the data.  
stringProbs(dat, N=5)
```

theta.max

Maximum Likelihood for IRT

Description

Returns the MLE for the Rasch Model given a set of item parameters

Usage

```
theta.max(x, betas)
```

Arguments

x	Vector of item responses
betas	Vector of Rasch location parameters

Author(s)

Harold Doran

See Also

irt.ability for a more general function

Examples

```
x <- c(1,1,1,0,0,1)  
b <- c(-1, -.5, 0, 0, .5, 1)  
theta.max(x, b)
```

 wrongProb

Function to find the probability of choosing the wrong option.

Description

This functions supplements the cheat function. It uses classical test methods for computing the probability of choosing the incorrect test option to the ith item.

Usage

```
wrongProb(...)
## Default S3 method:
wrongProb(dat, key, ...)
## S3 method for class 'formula'
wrongProb(formula, data, na.action = NULL, subset, key, ...)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
data	an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>cheat</code> is called.
dat	A data frame or matrix with item responses. Implemented only for the default method.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to <code>getOption("na.action")</code> .
subset	an optional vector specifying a subset of observations to be used.
key	a numeric vector containing the correct responses to each test item
...	Not implemented

Value

a list containing the probability of choosing each incorrect option for each test item appearing on the test.

Author(s)

Harold Doran

See Also

See Also as [cheat](#)

Examples

```
NumStu <- 30
NumItems <- 50
dat <- matrix(0, nrow=NumStu, ncol=NumItems)
set.seed(1234)
for(i in 1:NumStu){
  dat[i,] <- sample(1:4, NumItems, replace=TRUE)
}
dat <- data.frame(dat)

## Add in explicit answer copying
dat[(NumStu+1),] <- dat[NumStu,]
dat[(NumStu+2),] <- c(dat[(NumStu-1), 1:25], dat[(NumStu-2), 26:50 ])

## Answer Key
set.seed(1234)
key <- sample(1:4, NumItems, replace=TRUE)

## Formula interface
ff <- as.formula(paste('~', paste( names(dat), collapse= "+")))
mm <- wrongProb(ff, data = dat, key = key)

## Default interface
mm <- wrongProb(dat, key = key)
```

Index

*Topic **misc**

- alpha, [3](#)
- cheat, [4](#)
- choose.M, [6](#)
- class.acc, [7](#)
- classical, [8](#)
- irt.ability, [9](#)
- jml, [11](#)
- mml, [12](#)
- plaus.val, [14](#)
- posterior, [15](#)
- scoreCon, [16](#)
- simRasch, [17](#)
- SL, [17](#)
- SLderivs, [19](#)
- SSI, [19](#)
- stringMatch, [21](#)
- stringProbs, [22](#)
- theta.max, [23](#)
- wrongProb, [24](#)

*Topic **package**

- MiscPsycho-package, [2](#)

alpha, [3](#)

cheat, [4](#), [24](#)
choose.M, [6](#)
class.acc, [7](#)
classical, [8](#)

fit.Stats (jml), [11](#)

irt.ability, [9](#), [16](#), [18](#)

jml, [11](#)

MiscPsycho (MiscPsycho-package), [2](#)
MiscPsycho-package, [2](#)
mml, [12](#)

plaus.val, [14](#)

posterior, [15](#)

scoreCon, [16](#)
simRasch, [17](#)
SL, [17](#)
SLderivs, [19](#)
SSI, [19](#)
stringMatch, [21](#)
stringProbs, [22](#)

theta.max, [23](#)

wrongProb, [5](#), [24](#)