Package ‘cdcatR’

April 16, 2020

Type Package

Title Cognitive Diagnostic Computerized Adaptive Testing in R

Version 1.0.0

Date 2020-04-16

Description This package holds functions for conducting CD-CAT applications.

License GPL-3

LazyData TRUE

Depends R (>= 3.4.0)

Imports GDINA (>= 2.2.0), ggplot2 (>= 3.3.0), tibble, cowplot, doParallel (>= 1.0.15), foreach, CDM (>= 7.4.19), snow, NPCD, stats

URL `https://github.com/miguel-sorrel/cdcatR`

BugReports `https://github.com/miguel-sorrel/cdcatR/issues`

RoxygenNote 7.1.0

Encoding UTF-8

Author Miguel A. Sorrel [aut, cre, cph],
Pablo Nájera [aut, cph],
Francisco J. Abad [aut, cph]

Maintainer Miguel A. Sorrel <`miguel.sorrel@uam.es`>

R topics documented:

`att.plot` *Plots for attribute mastery estimates*

Description

This function generates a plot monitoring the attribute mastery estimates (*X*: item position, *Y*: mastery probability). If a parametric CD-CAT has been conducted, posterior probabilites (with confident intervals) of mastering each attribute are plotted. If a nonparametric CD-CAT has been conducted (and pseudo-probabilites have been computed), both nonparametric classification and pseudo-posterior probabilites (with confident intervals) of mastering each attribute are plotted. Colors are used in the plots to indicate mastery (green), non-mastery (red), or uncertainty (blue).

Usage

```
att.plot(cdcat.obj, i, k = NULL)
```

Arguments

`cdcat.obj`	An object of class `cdcat`
`i`	Examinee to be plotted
`k`	Attribute/s to be plotted. Default is NULL, which plots all attributes

Value

`att.plot` creates a plot.

`cdcat` *Cognitively based computerized adaptive test application*

Description

`cdcat` conducts a CD-CAT application for a given dataset. Different item selection rules can be used: the general discrimination index (GDI; de la Torre & Chiu, 2016; Kaplan et al., 2015), the Jensen-Shannon divergence index (JSD; Kang et al., 2017; Minchen & de la Torre, 2016; Yigit et al., 2018), the posterior-weighted Kullback-Leibler index (PWKL; Cheng, 2009), the modified PWKL index (MPWKL; Kaplan et al., 2015), the nonparametric item selection method (NPS; Chang et al., 2018), or random selection. Fixed length or fixed precision CD-CAT can be applied. Fixed precision CD-CAT with NPS is available, by using the pseudo-posterior probability of each student mastering each attribute (experimental).

Usage

```
cdcat(
  fit = NULL,
  dat = NULL,
  Q = NULL,
  itemSelect = "GDI",
  MAXJ = 20,
  FIXED.LENGTH = TRUE,
  att.prior = NULL,
  post.initial = NULL,
  max.cut = 0.8,
  NPS.args = list(gate = NULL, pseudo.prob = T, w.type = 1, seed = NULL),
  n.cores = 2,
  i.print = 250
)
```

Arguments

`fit`	Calibrated item bank with the GDINA or CDM package
`dat`	Dataset to be analyzed. If `is.null(dat)` it takes data from the fit object (i.e., post-hoc simulation)
`Q`	Q-matrix to be used in the analysis. If `is.null(Q)` it takes Q from the fit object
`itemSelect`	Item selection rule: GDI, JSD, MPWKL, PWKL, NPS, random
`MAXJ`	Maximum number of items to be applied. Default is 20
`FIXED.LENGTH`	Fixed CAT-length (TRUE) or fixed-precision (FALSE). Default is TRUE
`att.prior`	Prior distribution for MAP/EAP estimates
`post.initial`	Prior distribution for `itemSelect`
`max.cut`	Cutoff for fixed-precision (posterior pattern > max.cut). When `itemSelect = "NPS"` this is evaluated at the attribute level using the pseudo-posterior probabilities for each attribute (*K* posterior probabilities > max.cut). Default is .80. A higher cutoff is recommended when `itemSelect = "NPS"`
`NPS.args`	A list of options when `itemSelect = "NPS"`. `gate = "AND"` or `"OR"`, depending on whether a conjunctive o disjunctive nonparametric CDM is used. `pseudo.prob` = pseudo-posterior probability of each examinee mastering each attribute (experimental). `w.type` = weight type used for computing the pseudo-posterior probability (experimental): 1 = Power-of-2 weight; 2 = Exponential weight. `seed` = NPS has a random component, so a seed is required for consistent results.
`n.cores`	Number of cores to be used during parallelization. Default is 2
`i.print`	Print examinee information. Default is 250

Value

`cdcat` returns an object of class `cdcat`.

est A list of that contains for each examinee the mastery probability estimates at each step of the CAT (`est.cat`) and the items applied (`item.usage`)

specifications A list of that contains all the specifications

References

Chang, Y.-P., Chiu, C.-Y., & Tsai, R.-C. (2019). Nonparametric CAT for CD in educational settings with small samples. *Applied Psychological Measurement, 43*, 543-561.

Cheng, Y. (2009). When cognitive diagnosis meets computerized adaptive testing: CD-CAT. *Psychometrika, 74*, 619-632.

de la Torre, J., & Chiu, C. Y. (2016). General method of empirical Q-matrix validation. *Psychometrika, 81*, 253-273.

Kang, H.-A., Zhang, S., & Chang, H.-H. (2017). Dual-objective item selection criteria in cognitive diagnostic computerized adaptive testing. *Journal of Educational Measurement, 54*, 165-183.

Kaplan, M., de la Torre, J., & Barrada, J. R. (2015). New item selection methods for cognitive diagnosis computerized adaptive testing. *Applied Psychological Measurement, 39*, 167-188.

Minchen, N., & de la Torre, J. (2016, July). *The continuous G-DINA model and the Jensen-Shannon divergence*. Paper presented at the International Meeting of the Psychometric Society, Asheville, NC, United States.

Yigit, H. D., Sorrel, M. A., de la Torre, J. (2018). Computerized adaptive testing for cognitively based multiple-choice data. *Applied Psychological Measurement, 43*, 388-401.

Examples

```
######################################
# Example 1.                         #
# CD-CAT simulation for a GDINA obj  #
######################################

#-----------Data generation----------#
Q <- sim180GDINA$simQ
K <- ncol(Q)
dat <- sim180GDINA$simdat.c
att <- sim180GDINA$simalpha.c

#----------Model estimation----------#
fit <- GDINA::GDINA(dat = dat, Q = Q, verbose = 0) # GDINA package
#fit <- CDM::gdina(data = dat, q.matrix = Q, progress = 0) # CDM package

#---------------CD-CAT---------------#
res.FIXJ <- cdcat(fit = fit, dat = dat, FIXED.LENGTH = TRUE, n.cores = 4)
res.VARJ <- cdcat(fit = fit, dat = dat, FIXED.LENGTH = FALSE, n.cores = 4)

#---------------Results--------------#
res.FIXJ$est[[1]] # estimates for the first examinee (fixed-length)
res.VARJ$est[[1]] # estimates for the first examinee (fixed-precision)
att.plot(cdcat.obj = res.FIXJ, i = 1) # plot for the first examinee (fixed-length)
att.plot(cdcat.obj = res.VARJ, i = 1) # plot  for the first examinee (fixed-precision)
# FIXJ summary
res.FIXJ.sum.real <- cdcat.summary(cdcat.obj = res.FIXJ, alpha = att) # vs. real accuracy
res.FIXJ.sum.real$recovery$plotPCV
res.FIXJ.sum.real$recovery$plotPCA
# VARJ summary
res.VARJ.sum.post <- cdcat.summary(cdcat.obj = res.VARJ, alpha = att)
res.VARJ.sum.post$CATlength$stats
res.VARJ.sum.post$CATlength$plot
res.VARJ.sum.post$recovery
# Post-hoc CAT simulation (only if dat is fit$options$dat)
att.J <- GDINA::personparm(fit, "MAP")[, -(K+1)] # GDINA package
# att.J <- t(sapply(strsplit(as.character(fit$pattern$map.est), ""), as.numeric)) # CDM package
class.J <- GDINA::ClassRate(att, att.J) # upper-limit for accuracy
res.FIXJ.sum.post <- cdcat.summary(cdcat.obj = res.FIXJ, alpha = att.J)
res.FIXJ.sum.post$recovery$plotPCV + ggplot2::geom_hline(yintercept = class.J$PCV[K], color = "red")
res.FIXJ.sum.post$recovery$plotPCA + ggplot2::geom_hline(yintercept = class.J$PCA, color = "red")

######################################
# Example 2.                         #
# CD-CAT simulation for multiple     #
# GDINA objs and comparison of       #
# performance on a validation sample #
######################################

#----------------Data----------------#
Q <- sim180combination$simQ
K <- ncol(Q)
parm <- sim180combination$simcatprob.parm
dat.c <- sim180combination$simdat.c
att.c <- sim180combination$simalpha.c
dat.v <- sim180combination$simdat.v
att.v <- sim180combination$simalpha.v

#-----(multiple) Model estimation----#
fitTRUE <- GDINA::GDINA(dat = dat.c, Q = Q, catprob.parm = parm,

           control = list(maxitr = 0), verbose = 0)
fitGDINA <- GDINA::GDINA(dat = dat.c, Q = Q, verbose = 0)
fitDINA <- GDINA::GDINA(dat = dat.c, Q = Q, model = "DINA", verbose = 0)
fitDINO <- GDINA::GDINA(dat = dat.c, Q = Q, model = "DINO", verbose = 0)
fitACDM <- GDINA::GDINA(dat = dat.c, Q = Q, model = "ACDM", verbose = 0)
LR2step <- LR.2step(fitGDINA)
models <- LR2step$models.adj.pvalues
fitLR2 <- GDINA::GDINA(dat = dat.c, Q = Q, model = models, verbose = 0)

#---------------CD-CAT---------------#
fit.l <- list(fitTRUE, fitGDINA, fitDINA, fitDINO, fitACDM, fitLR2)
res.FIXJ.l <- res.VARJ.l <- list()
for(mm in 1:length(fit.l)) {
 fit <- fit.l[[mm]]
 res.FIXJ.l[[mm]] <- cdcat(fit = fit, dat = dat.v, FIXED.LENGTH = TRUE, n.cores = 4)
 res.VARJ.l[[mm]] <- cdcat(fit = fit, dat = dat.v, FIXED.LENGTH = FALSE, n.cores = 4)
}

#---------------Results--------------#
fitbest <- GDINA::GDINA(dat = dat.v, Q = Q, catprob.parm = parm,
          control = list(maxitr = 1), verbose = 0)
fitbest.acc <- GDINA::personparm(fitbest, "MAP")[, -(K+1)]
class.J <- GDINA::ClassRate(att.v, fitbest.acc) # upper-limit for accuracy
# FIXJ comparison
res.FIXJ.sum.post.summary <- cdcat.summary(cdcat.obj = res.FIXJ.l, alpha = att.v)
res.FIXJ.sum.post.summary$PCVcomp + ggplot2::geom_hline(yintercept = class.J$PCV[K], color = "red")
res.FIXJ.sum.post.summary$PCAmcomp + ggplot2::geom_hline(yintercept = class.J$PCA, color = "red")
# VARJ comparison
res.VARJ.sum.post.summary <- cdcat.summary(cdcat.obj = res.VARJ.l, alpha = att.v)
res.VARJ.sum.post.summary$stats
res.VARJ.sum.post.summary$plots
res.VARJ.sum.post.summary$recovery

######################################
# Example 3.                         #
# Nonparametric CD-CAT for           #
# small-scale assessment             #
######################################

#-----------Data generation----------#
```

```
Q <- sim180DINA$simQ
K <- ncol(Q)
N <- 50
dat <- sim180DINA$simdat.c[1:N,]
att <- sim180DINA$simalpha.c[1:N,]

#--------Nonparametric CD-CAT--------#
res.NPS.FIXJ <- cdcat(dat = dat, Q = Q, itemSelect = "NPS",
                      FIXED.LENGTH = TRUE, MAXJ = 30,
                NPS.args = list(gate = "AND", pseudo.prob = TRUE, w.type = 1, seed = 12345),
                      n.cores = 4)
res.NPS.VARJ <- cdcat(dat = dat, Q = Q, itemSelect = "NPS",
                      FIXED.LENGTH = FALSE, MAXJ = 30, max.cut = 0.95,
                NPS.args = list(gate = "AND", pseudo.prob = TRUE, w.type = 1, seed = 12345),
                      n.cores = 4)

#---------------Results--------------#
res.NPS.FIXJ$est[[1]] # estimates for the first examinee (fixed-length)
res.NPS.VARJ$est[[1]] # estimates for the first examinee (fixed-precision)
att.plot(res.NPS.FIXJ, i = 1) # plot for estimates for the first examinee (fixed-length)
att.plot(res.NPS.VARJ, i = 1) # plot for estimates for the first examinee (fixed-precision)
# FIXJ summary
res.NPS.FIXJ.sum.real <- cdcat.summary(cdcat.obj = res.NPS.FIXJ, alpha = att) # vs. real accuracy
res.NPS.FIXJ.sum.real$recovery$plotPCV
res.NPS.FIXJ.sum.real$recovery$plotPCA
# VARJ summary
res.NPS.VARJ.sum.post <- cdcat.summary(cdcat.obj = res.NPS.VARJ, alpha = att)
res.NPS.VARJ.sum.post$CATlength$stats
res.NPS.VARJ.sum.post$CATlength$plot
res.NPS.VARJ.sum.post$recovery
# Post-hoc CAT simulation
fit <- NPCD::AlphaNP(Y = dat, Q = Q, gate = "AND")
att.J <- fit$alpha.est
class.J <- GDINA::ClassRate(att, att.J) # upper-limit for accuracy
res.NPS.FIXJ.sum.post <- cdcat.summary(cdcat.obj = res.NPS.FIXJ, alpha = att.J)
res.NPS.FIXJ.sum.post$recovery$plotPCV + ggplot2::geom_hline(yintercept = class.J$PCV[K],
                                                             color = "firebrick3")
res.NPS.FIXJ.sum.post$recovery$plotPCA + ggplot2::geom_hline(yintercept = class.J$PCA,
                                                             color = "firebrick3")
```

`cdcat.summary` *Summary information for a* `cdcat` *object*

Description

This function provides classification accuracy and CAT length results for `cdcat` object. If a list of `cdcat` objects is included, these objects are compared through different in different tables and figures.

Usage

```
cdcat.summary(cdcat.obj, alpha, label = NULL)
```

Arguments

`cdcat.obj`	An object or list of objects of class `cdcat`
`alpha`	N x K matrix with the attribute patterns to be compared to the `cdcat` results
`label`	labels for the `cdcat` objects. If NULL (by default), the models are used as labels

Value

`cdcat.summary` returns an object of class `cdcat.summary`.

`gen.itembank` *Item bank generation*

Description

This function can be used to generate a calibration and a validation sample of responses to an item bank. The user can provide a Q-matrix or create one defining a set of arguments. Item quality is sampled from a uniform distribution with mean = *IQ* and variance = *VAR*. Item parameters are generated with monotonicity constraint. Data are generated following either a uniform, multivariate normal, or higher-order distribution.

Usage

```
gen.itembank(
  N.c,
  N.v = NULL,
  Q = NULL,
  gen.Q = list(J = NULL, K = NULL, propK.J = NULL, nI = NULL, minJ.K = NULL, max.Kcor =
    NULL),
  IQ,
  VAR,
  model = "GDINA",
  min.param = 0,
  gen.att = list(att.dist = "uniform", mvnorm.cor = 0, mvnorm.mean = 0, mvnorm.cutoff =
    0, higher.order.a = 0, higher.order.b = 0),
  seed = NULL
)
```

Arguments

`N.c`	Sample size for the calibration sample
`N.v`	Sample size for the validation sample. If NULL (by default), no validation data are provided
`Q`	Q-matrix
`gen.Q`	A list of arguments to generate a Q-matrix if `Q` is not provided. `J`: number of items. `K`: number of attributes. `propK.J`: vector summing up to 1 that determines the proportion of 1-attribute, 2-attribute, ..., items (See `Examples`). `nI`: minimum number of identity matrices to be included in the Q-matrix. `minJ.K`: minimum number of items measuring each attribute. `max.Kcor`: maximum correlation allowed between two attributes

`IQ`	Item discrimination (mean for the uniform distribution). *IQ* = *P*(**1**) - *P*(**0**) (Sorrel, Abad, Olea, de la Torre, and Barrada, 2017; NÃ¡jera, Sorrel, de la Torre, and Abad, in press)
`VAR`	Item discrimination (variance for the uniform distribution)
`model`	A character vector with one model for each item, or a single value to be used for all items. The possible options include `"DINA"`, `"DINO"`, `"ACDM"`, and `"GDINA"`. One-attribute items will be coded as `"GDINA"`
`min.param`	Minimum value for the delta parameter of the principal effects of each attribute. Only usable if `model` = `"ACDM"` or `model` = `"GDINA"`
`gen.att`	A list of arguments for the attribute profiles generation. Uniform attribute distribution by default. See `simGDINA` function of package `GDINA`
`seed`	A scalar to use with `set.seed`

Value

`gen.itembank` returns an object of class `gen.itembank`.

simdat.c A matrix with the calibration dataset

simalpha.c A matrix with the alpha patterns for the calibration dataset

simdat.v A matrix with the validation dataset (only if `N.v` is specified)

simalpha.v A matrix with the alpha patterns for the validation dataset (only if `N.v` is specified)

simQ Q-matrix

simcatprob.parm A list of non-zero categories success probabilities for each latent group in each item

simdelta.parm A list of delta parameters for each item

specifications A list that contains all the specifications

References

Kaplan, M., de la Torre, J., & Barrada, J. R. (2015). New item selection methods for cognitive diagnosis computerized adaptive testing. *Applied Psychological Measurement, 39*, 167-188.

Najera, P., Sorrel, M. A., de la Torre, J., & Abad, F. J. (in press). Improving robustness in Q-matrix validation using an iterative and dynamic procedure. *Applied Psychological Measurement*.

Sorrel, M. A., Abad, F. J., Olea, J., de la Torre, J., & Barrada, J. R. (2017). Inferential item-fit evaluation in cognitive diagnosis modeling. *Applied Psychological Measurement, 41*, 614-631.

Examples

```
####################################
# Example 1.                       #
# Generate calibration sample      #
# and item parameters providing    #
# a Q-matrix, with G-DINA model    #
# and uniform attribute            #
# distribution                     #
####################################

Q <- sim180GDINA$simQ
model <- rep("GDINA", each = nrow(Q))
IQ <- .70 # P(1), IQ = Low item quality in Kaplan, de la Torre & Barrada (2015)
```

```
VAR <- .10 # High variance in Kaplan et al. (2015)
bank <- gen.itembank(N.c = 1000, Q = Q, IQ = IQ, VAR = VAR, model = model)

####################################
# Example 2.                       #
# Generate calibration sample      #
# and item parameters providing    #
# a Q-matrix, with different       #
# models and higher-order          #
# attribute distribution           #
####################################

Q <- sim180GDINA$simQ
K <- ncol(Q)
model <- sample(c("DINA", "DINO", "ACDM"), size = nrow(Q), replace = TRUE)
IQ <- .70
VAR <- .10
bank <- gen.itembank(N.c = 1000, Q = Q, IQ = IQ, VAR = VAR, model = model,
                     gen.att = list(att.dist = "higher.order",
                      higher.order.a = runif(K, 0.8, 1.3),
                      higher.order.b = seq(-2, 2, length.out = K)))

####################################
# Example 3.                       #
# Generate calibration and         #
# validation samples and item      #
# parameters without providing a   #
# Q-matrix (using gen.Q arguments) #
####################################

J <- 150
K <- 5
propK.J <- c(0.5, 0.3, 0, 0.2)
nI <- 3
minJ.K <- 50
max.Kcor <- 0.5
IQ <- .80
VAR <- .05
min.param <- 0.1
bank <- gen.itembank(N.c = 1000, N.v = 2000,
                     gen.Q = list(J = J, K = K, propK.J = propK.J,
                     nI = nI, minJ.K = minJ.K, max.Kcor = max.Kcor),
                     IQ = IQ, VAR = VAR, min.param = min.param)
```

`LR.2step` *Item-level model comparison using 2LR test*

Description

This function evaluates whether the saturated G-DINA model can be replaced by reduced CDMs without significant loss in model data fit for each item using two-step likelihood ratio test (2LR). Sorrel, de la Torre, Abad, and Olea (2017) and Ma & de la Torre (2018) can be consulted for details.

Usage

```
LR.2step(fit, p.adjust.method = "holm", alpha.level = 0.05)
```

Arguments

`fit`	Calibrated item bank with GDINA or CDM package
`p.adjust.method`	Correction method for p-values. Possible values include "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". See p.adjust function from stats for additional details. Default is holm
`alpha.level`	Alpha level for decision. Default is 0.05

Value

`LR2.step` returns an object of class `LR2.step`

LR2 LR2 statistics

pvalues p-values associated with the wald statistics

adj.pvalues adjusted p-values associated with the wald statistics

df degrees of freedom

models.adj.pvalues Models selected using the rule *largestp* (Ma, Iaconangelo, & de la Torre, 2016). All statistics whose p-values are less than `alpha.level` are rejected. All statistics with p-value larger than `alpha.level` define the set of candidate reduced models. The G-DINA model is retained if all statistics are rejected. Whenever the set includes more than one model, the model with the largest p-value was selected as the best model for that item.

References

Ma, W. & de la Torre, J. (2018). Category-level model selection for the sequential G-DINA model. *Journal of Educational and Behavorial Statistic, 44*, 45-77.

Ma, W., Iaconangelo, C., & de la Torre, J. (2016). Model similarity, model selection and attribute classification. *Applied Psychological Measurement, 40*, 200-217.

Sorrel, M. A., de la Torre, J., Abad, F. J., & Olea, J. (2017). Two-step likelihood ratio test for item-level model comparison in cognitive diagnosis models. *Methodology, 13*, 39-47.

Examples

```
Q <- sim180DINA$simQ
dat <- sim180DINA$simdat.c
resGDINA <- GDINA::GDINA(dat = dat, Q = Q, model = "GDINA",verbose = FALSE)
resCDM <- CDM::gdina(data = dat, q.matrix = Q, rule = "GDINA", progress = FALSE)
LR2.GDINA <- LR.2step(fit = resGDINA)
LR2.CDM <- LR.2step(fit = resCDM)
mean(LR2.GDINA$models.adj.pvalues[which(rowSums(Q) != 1)] ==
      sim180DINA$specifications$model[which(rowSums(Q) != 1)])
mean(LR2.CDM$models.adj.pvalues[which(rowSums(Q) != 1)] ==
      sim180DINA$specifications$model[which(rowSums(Q) != 1)])
```

`sim180combination` *Simulated data (180 items, a combination of DINA, DINO, and A-CDM items)*

Description

Simulated data, Q-matrix and item parameters for a 180-item bank with 5 attributes. Data generated using the `gen.itembank` function.

Usage

```
sim180combination
```

Format

A list with components:

`simdat.c` Simulated responses of 500 examinees

`simalpha.c` Simulated attribute patterns of 500 examinees

`simQ` Simulated Q-matrix

`simcatprob.parm` Simulated success probabilities for each latent group in each item

`simdelta.parm` Simulated delta parameters for each item

`specifications` A list that contains all the specifications that were used in the `gen.itembank` function

`sim180DINA` *Simulated data (180 items, DINA model)*

Description

Simulated data, Q-matrix and item parameters for a 180-item bank with 5 attributes. Data generated using the `gen.itembank` function.

Usage

```
sim180DINA
```

Format

A list with components:

`simdat.c` Simulated responses of 500 examinees

`simalpha.c` Simulated attribute patterns of 500 examinees

`simQ` Simulated Q-matrix

`simcatprob.parm` Simulated success probabilities for each latent group in each item

`simdelta.parm` Simulated delta parameters for each item

`specifications` A list that contains all the specifications that were used in the `gen.itembank` function `sim180GDINA` *Simulated data (180 items, G-DINA model)*

Description

Simulated data, Q-matrix and item parameters for a 180-item bank with 5 attributes. Data generated using the `gen.itembank` function.

Usage

```
sim180GDINA
```

Format

A list with components:

`simdat.c` Simulated responses of 500 examinees

`simalpha.c` Simulated attribute patterns of 500 examinees

`simQ` Simulated Q-matrix

`simcatprob.parm` Simulated success probabilities for each latent group in each item

`simdelta.parm` Simulated delta parameters for each item

`specifications` A list that contains all the specifications that were used in the `gen.itembank` function

Index