Package 'catIrt'

February 19, 2015

Type Package

Version 0.5-0

Date 2014-10-04

Title An R Package for Simulating IRT-Based Computerized Adaptive Tests

Author Steven W. Nydick

Maintainer Steven W. Nydick <nydic001@umn.edu>

Depends R (>= 2.11.0), numDeriv (>= 2012.3-1)

Suggests irtoys, ltm, catR

Description Functions designed to simulate data that conform to basic unidimensional IRT models (for now 3-parameter binary response models and graded response models) along with Post-Hoc CAT simulations of those models with various item selection methods, ability estimation methods, and termination criteria.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-10-05 07:43:12

R topics documented:

catIrt-package *An R Package for Simulating Computerized Adaptive Tests (CATs)*

Description

catIrt provides methods for simulating Computerized Adaptive Tests, including response simulation, item selection methods, ability estimation methods, and test termination methods. Unique in catIrt is support for the graded response model (and, soon, other polytomous models) along with expanding support for classification CAT (including an implementation of the SPRT, GLR, and CI methods).

Details

Package:	catIrt
Type:	Package
Version:	0.5-0
Date:	2014-10-04
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Maintainer: Steven W. Nydick <nydic001@umn.edu>

catIrt *Simulate Computerized Adaptive Tests (CATs)*

Description

catIrt simulates Computerized Adaptive Tests (CATs) given a vector/matrix of responses or a vector of ability values, a matrix of item parameters, and several item selection mechanisms, estimation procedures, and termination criteria.

Usage

```
catIrt( params, mod = c("brm", "grm"),
        resp = NULL,
        theta = NULL,
        catStart = list( n.start = 5, init.theta = 0,
                         select = c("UW-FI", "LW-FI", "PW-FI",
                                    "FP-KL", "VP-KL", "FI-KL", "VI-KL",
```

```
                                    "random"),
                          at = c("theta", "bounds"),
                          it.range = NULL, n.select = 1,
                          delta = .1,
                        score = c("fixed", "step", "random", "WLE", "BME", "EAP"),
                          range = c(-1, 1),
                          step.size = 3, leave.after.MLE = FALSE ),
          catMiddle = list( select = c("UW-FI", "LW-FI", "PW-FI",
                                       "FP-KL", "VP-KL", "FI-KL", "VI-KL",
                                       "random"),
                           at = c("theta", "bounds"),
                           it.range = NULL, n.select = 1,
                           delta = .1,
                           score = c("MLE", "WLE", "BME", "EAP"),
                           range = c(-6, 6),
                           expos = c("none", "SH") ),
          catTerm = list( term = c("fixed", "precision", "info", "class"),
                         score = c("MLE", "WLE", "BME", "EAP"),
                         n.min = 5, n.max = 50,
                         p.term = list(method = c("threshold", "change"),
                                       crit = .25),
                         i.term = list(method = c("threshold", "change"),
                                       crit = 2),
                         c.term = list(method = c("SPRT", "GLR", "CI"),
                                       bounds = c(-1, 1),
                                       categ = c(0, 1, 2),
                                       delta = .1,
                                       alpha = .05, beta = .05,
                                       conf.lev = .95) ),
          ddist = dnorm,
          progress = TRUE, ... )
  ## S3 method for class 'catIrt'
  summary( object, group = TRUE, ids = "none", ... )
  ## S3 method for class 'catIrt'
  plot( x, which = "all", ids = "none",
        conf.lev = .95, legend = TRUE, ask = TRUE, ... )
```

Arguments

`object, x`	a `catIrt` object.
`params`	**numeric:** a matrix of item parameters. If specified as a matrix, the rows must index the items, and the columns must designate the item parameters. For the binary response model, `params` must either be a 3-column matrix (if not using item exposure control), a 4-5-column matrix (with Sympson-Hetter parameters as the last column if using item exposure control), or a 4-5-column matrix (if including the item number as the first column). See **Details** for more information.
`mod`	**character:** a character string indicating the IRT model. Current support is for the 3-parameter binary response model (‘`"brm"`’), and Samejima's graded re-

sponse model (‘"grm"’). The contents of params must match the designation of mod. If mod is left blank, it will be designated the class of resp (if resp inherits either ‘"brm"’ or ‘"grm"’), and if that fails, it will ask the user (if in interactive mode) or error.

resp **numeric:** either a $N \times J$ matrix (where N indicates the number of simulees and J indicates the number of items), a J length vector (if there is only one simulee), or NULL if specifying thetas. For the binary response model (‘"brm"’), resp must solely contain 0s and 1s. For the graded response model (‘"grm"’), resp must solely contain integers $1, \ldots, K$, where K is the number of categories, as indicated by the dimension of params.

theta **numeric:** either a N-dimensional vector (where N indicates the number of simulees) or NULL if specifying resp.

catStart **list:** a list of options for starting the CAT including:

1. n.start: a scalar indicating the number of items that are used for each simulee at the beginning of the CAT. After ‘n.start’ reaches the specified value, the CAT will shift to the middle set of parameters.
2. init.theta: a scalar or vector of initial starting estimates of θ. If init.theta is a scalar, every simulee will have the same starting value. Otherwise, simulees will have different starting values based on the respective element of init.theta.
3. select: a character string indicating the item selection method for the first few items. Items can be selected either through maximum Fisher information or Kullback-Leibler divergence methods or randomly. The Fisher information methods include
   - ‘"UW-FI"’: unweighted Fisher information at a point.
   - ‘"LW-FI"’: Fisher information weighted across the likelihood function.
   - ‘"PW-FI"’: Fisher information weighted across the posterior distribution of θ.

   And the Kullback-Leibler divergence methods include
   - ‘"FP-KL"’: pointwise KL divergence between [P +/- delta], where P is either the current θ estimate or a classification bound.
   - ‘"VP-KL"’: pointwise KL divergence between [P +/- delta/sqrt(n)], where n is the number of items given to this point in the CAT.
   - ‘"FI-KL"’: KL divergence integrated along [P -/+ delta] with respect to P
   - ‘"VI-KL"’: KL divergence integrated along [P -/+ delta/sqrt(n)] with respect to P.

   See itChoose for more information.
4. at: a character string indicating where to select items. If select is ‘"UW-FI"’ and at is ‘"theta"’, then items will be selected to maximize Fisher information at the proximate θ estimates.
5. it.range: Either a 2-element numeric vector indicating the minimum and maximum allowed difficulty parameters for items selected during the starting portion of the CAT (only if mod is equal to ‘"brm"’) or NULL indicating no item parameter restrictions. See itChoose for more information.

6. `n.select`: an *integer* indicating the number of items to select at one time. For instance, if `select` is '`"UW-FI"`', `at` is '`"theta"`', and `n.select` is 5, the item choosing function will randomly select between the top *5* items that maximize expected Fisher information at proximate θ estimates.
7. `delta`: a scalar indicating the multiplier used in initial item selection if a Kullback-Leibler method is chosen.
8. `score`: a character string indicating the θ estimation method. As of now, the options for scoring the first few items are '`"fixed"`' (at `init.thet`), '`"step"`' (by adding or subtracting `step.size` θ estimates after each item), Weighted Likelihood Estimation ('`"WLE"`'), Bayesian Modal Estimation ('`"BME"`'), and Expected A-Posteriori Estimation ('`"EAP"`'). The latter two allow user specified prior distributions through density (`d...`) functions. See `mleEst` for more information.
9. `range`: a 2-element numeric vector indicating the minimum and maximum that θ should be estimated in the starting portion of the CAT.
10. `step.size`: a scalar indicating how much to increment or decrement the estimate of θ if `score` is set to '`"step"`'.
11. `leave.after.MLE`: a logical indicating whether to skip the remainder of the starting items if the user has a mixed response pattern and/or a finite maximum likelihood estimate of θ can be achieved.

`catMiddle` **list:** a list of options for selecting/scoring during the middle of the CAT, including:

1. `select`: a character string indicating the item selection method for the remaining items. See `select` in `catStart` for an explanation of the options.
2. `at`: a character string indicating where to select items. See `select` in `catStart` for an explanation of the options.
3. `it.range`: Either a 2-element numeric vector indicating the minimum and maximum allowed difficulty parameters for items selected during the middle portion of the CAT (only if `mod` is equal to '`"brm"`') or NULL indicating no item parameter restrictions. See `itChoose` for more information.
4. `n.select`: an *integer* indicating the number of items to select at one time.
5. `delta`: a scalar indicating the multiplier used in middle item selection if a Kullback-Leibler method is chosen.
6. `score`: a character string indicating the θ estimation method. As of now, the options for scoring the remaining items are Maximum Likelihood Estimation ('`"MLE"`'), Weighted Likelihood Estimation ('`"WLE"`'), Bayesian Modal Estimation ('`"BME"`'), and Expected A-Posteriori Estimation ('`"EAP"`'). The latter two allow user specified prior distributions through density (`d...`) functions. See `mleEst` for more information.
7. `range`: a 2-element numeric vector indicating the minimum and maximum that θ should be estimated in the middle portion of the CAT.
8. `expos`: a character string indicating whether no item exposure controls should be implemented ('`"none"`') or whether the CAT should use Sympson-Hetter exposure controls ('`"SH"`'). If (and only if) `expos` is equal to '`"SH"`', the last column of the parameter matrix should indicate the probability of an item being administered given that it is selected.

`catTerm` **list:** a list of options for stopping/terminating the CAT, including:

1. `term`: a scalar/vector indicating the termination criterion/criteria. CATs can be terminated either through a fixed number of items (‘`"fixed"`’) declared through the `n.max` argument; related to SEM of a simulee (‘`"precision"`’) declared through the `p.term` argument; related to the test information of a simulee at a particular point in the cat (‘`"info"`’) declared through the `i.term` argument; and/or when a simulee falls into a category. If more than one termination criteria is selected, the CAT will terminate after successfully satisfying the *first* of those for a given simulee.
2. `score`: a character string indicating the θ estimation method for all of the responses in the bank. `score` is used to estimate θ given the entire bank of item responses and parameter set. If the theta estimated using all of the responses is far away from θ, the size of the item bank is probably too small. The options for `score` in `catTerm` are identical to the options of `score` in `catMiddle`.
3. `n.min`: an *integer* indicating the minimum number of items that a simulee should "take" before *any* of the termination criteria are checked.
4. `n.max`: an *integer* indicating the maximum number of items to administer before terminating the CAT.
5. `p.term`: a list indicating the parameters of a precision-based stopping rule, only if `term` is ‘`"precision"`’, including:
   (a) `method`: a character string indicating whether to terminate the CAT when the SEM dips below a threshold (‘`"threshold"`’) or changes less than a particular amount (‘`"change"`’).
   (b) `crit`: a scalar indicating either the maximum SEM of a simulee before terminating the CAT or the maximum change in the simulee's SEM before terminating the CAT.
6. `i.term`: a list indicating the parameters of a information-based stopping rule, only if `term` is ‘`"info"`’, including:
   (a) `method`: a character string indicating whether to terminate the CAT when FI exceeds a threshold (‘`"threshold"`’) or changes less than a particular amount (‘`"change"`’).
   (b) `crit`: a scalar indicating either the minimum FI of a simulee before terminating the CAT or the maximum change in the simulee's FI before terminating the CAT.
7. `c.term`: a list indicating the parameters of a classification CAT, only if `term` is ‘`"class"`’ or any of the selection methods are `at` one or more ‘`"bounds"`’, including:
   (a) `method`: a scalar indicating the method used for a classification CAT. As of now, the classification CAT options are the Sequential Probability Ratio Test (‘`"SPRT"`’), the Generalized Likelihood Ratio (‘`"GLR"`’), or the Confidence Interval method (‘`"CI"`’).
   (b) `bounds`: a scalar, vector, or matrix of classification bounds. If specified as a scalar, there will be one bound for each simulee at that value. If specified as a N-dimensional vector, there will be one bound for each simulee. If specified as a $k < N$-dimensional vector, there will be k

bounds for each simulee at those values. And if specified as a $N \times k$-element matrix, there will be k bounds for each simulee.

(c) `categ`: a vector indicating the names of the categories into which the simulees should be classified. The length of `categ` should be one greater than the length of `bounds`.

(d) `delta`: a scalar indicating the half-width of an indifference region when performing an SPRT-based classification CAT or selecting items by Kullback-Leibler divergence. See Eggen (1999) and `KL` for more information.

(e) `alpha`: a scalar indicating the specified Type I error rate for performing an SPRT- based classification CAT.

(f) `beta`: a scalar indicating the specified Type II error rate for performing an SPRT- based classification CAT.

(g) `conf.lev`: a scalar between 0 and 1 indicating the confidence level used when performing a confidence-based (`'"CI"'`) classification CAT.

`ddist` **function:** a function indicating how to calculate prior densities for Bayesian estimation or particular item selection methods. For instance, if you wish to specify a normal prior, `ddist = dnorm`, and if you wish to specify a uniform prior, `ddist = dunif`. Note that it is standard in R to use `d...` to indicate a density. See `itChoose` for more information.

`which` **numeric:** a scalar or vector of integers between 1 and 4, indicating which plots to include. The plots are as follows:

1. Bank Information
2. Bank SEM
3. CAT Information
4. CAT SEM

`which` can also be "none", in which case `plot.catIrt` will not plot any information functions, or it can be "all", in which case `plot.catIrt` will plot all four information functions.

`group` **logical:** TRUE or FALSE indicating whether to display a summary at the group level.

`ids` **numeric:** a scalar or vector of integers between 1 and the number of simulees indicating which simulees to plot and/or summarize their CAT process and *all* of their θ estimates. `ids` can also be "none" (or, equivalently, NULL) or "all".

`conf.lev` **numeric:** a scalar between 0 and 1 indicating the desired confidence level plotted for the individual θ estimates.

`legend` **logical:** TRUE or FALSE indicating whether the plot function should display a legend on the plot.

`ask` **logical:** TRUE or FALSE indicating whether the plot function should ask between plots.

`progress` **logical:** TRUE or FALSE indicating whether the `catIrt` function should display a progress bar during the CAT.

`...` arguments passed to `ddist` or `plot.catIrt`, usually distribution parameters identified by name or graphical parameters.

Details

The function `catIrt` performs a post-hoc computerized adaptive test (CAT), with a variety of user specified inputs. For a given person/simulee (e.g. simulee i), a CAT represents a simple set of stages surrounded by a `while` loop (e.g. Weiss and Kingsbury, 1984):

- Item Selection: The next item is chosen based on a pre-specified criterion/criteria. For example, the classic item selection mechanism is picking an item such that it maximizes Fisher Information at the current estimate of θ_i. Frequently, content balancing, item constraints, or item exposure will be taken into consideration at this point (aside from solely picking the "best item" for a given person). See `itChoose` for current item selection methods.
- Estimation: θ_i is estimated based on updated information, usually relating to the just-selected item and the response associated with that item. In a post-hoc CAT, all of the responses already exist, but in a standard CAT, "item administration" would be between "item selection" and "estimation." The classic estimation mechanism is estimating θ_i based off of maximizing the likelihood given parameters and a set of responses. Other estimation mechanisms correct for bias in the maximum likelihood estimate or add a prior information (such as a prior distribution of θ). If an estimate is untenable (i.e. it returns a non-sensical value or ∞), the estimation procedure needs to have an alternative estimation mechanism. See `mleEst` for current estimation methods.
- Termination: Either the test is terminated based on a pre-specified criterion/critera, or no termination criteria is satisfied, in which case the loop repeats. The standard termination criteria involve a fixed criterion (e.g. administering only 50 items), or a variable criterion (e.g. continuing until the observed SEM is below .3). Other termination criteria relate to cut-point tests (e.g. certification tests, classification tests), that depend not solely on ability but on whether that ability is estimated to exceed a threshold. `catIrt` terminates classification tests based on either the Sequential Probability Ratio Test (SPRT) (see Eggen, 1999), the Generalized Likelihood Ratio (GLR) (see Thompson, 2009), or the Confidence Interval Method (see Kingsbury & Weiss, 1983). Essentially, the SPRT compares the ratio of two likelihoods (e.g. the likelihood of the data given being in one category vs the likelihood of the data given being in the other category, as defined by $B + \delta$ and $B - \delta$ (where B separates the categories and δ is the halfwidth of the indifference region) and compares that ratio with a ratio of error rates (α and β) (see Wald, 1945). The GLR uses the maximum likelihood estimate in place of either $B+\delta$ or $B-\delta$, and the confidence interval method terminates a CAT if the confidence interval surrounding an estimate of θ is fully within one of the categories.

The CAT estimates θ_{i1} (an initial point) based on `init.theta`, and terminates the entire simulation after sequentially terminating each simulee's CAT.

Value

The function `catIrt` returns a list (of class "catIrt") with the following elements:

`cat_theta`	a vector of final CAT θ estimates.
`cat_categ`	a vector indicating the final classification of each simulee in the CAT. If `term` is *not* '`"class"`', `cat_categ` will be a vector of NA values.
`cat_info`	a vector of observed Fisher information based on the final CAT θ estimates and the item responses.

cat_sem	a vector of observed SEM estimates (or posterior standard deviations) based on the final CAT θ estimates and the item responses.
cat_length	a vector indicating the number of items administered to each simulee in the CAT
cat_term	a vector indicating how each CAT was terminated.
tot_theta	a vector of θ estimates given the entire item bank.
tot_categ	a vector indicating the classification of each simulee given the entire item bank.
tot_info	a vector of observed Fisher information based on the entire item bank worth of responses.
tot_sem	a vector of observed SEM estimates based on the entire item bank worth of responses.
true_theta	a vector of true θ values if specified by the user.
true_categ	a vector of true classification given θ.
full_params	the full item bank.
full_resp	the full set of responses.
cat_indiv	a list of θ estimates, observed SEM, observed information, the responses and the parameters chosen for each simulee over the entire CAT.
mod	a list of model specifications, as designated by the user, so that the CAT can be easily reproduced.

Note

Both `summary.catIrt` and `plot.catIrt` return different objects than the original `catIrt` function. `summary.catIrt` returns summary labeled summary statistics, and `plot.catIrt` returns evaluation points (x values, information, and SEM) for each of the plots. Moreover, if in interactive mode and missing parts of the `catStart`, `catMiddle`, or `catTerm` arguments, the `catIrt` function will interactively ask for each of those and return the set of arguments in the "catIrt" object.

Author(s)

Steven W. Nydick <nydic001@umn.edu>

References

Eggen, T. J. H. M. (1999). Item selection in adaptive testing with the sequential probability ratio test. *Applied Psychological Measurement*, *23*, 249 – 261.

Kingsbury, G. G., & Weiss (1983). A comparison of IRT-based adaptive mastery testing and a sequential mastery testing procedure. In D. J. Weiss (Ed.), *New horizons in testing: Latent trait test theory and computerized adaptive testing* (pp. 257–283). New York, NY: Academic Press.

Thompson, N. A. (2009). Using the generalized likelihood ratio as a termination criterion. In D. J. Weiss (Ed.), *Proceedings of the 2009 GMAC conference on computerized adaptive testing*.

Wainer, H. (Ed.). (2000). *Computerized Adaptive Testing: A Primer (2nd Edition)*. Mahwah, NJ: ELawrence Erlbaum Associates.

Wald, A. (1945). Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, *16*, 117 – 186.

Weiss, D. J., & Kingsbury, G. G. (1984). Application of computerized adaptive testing to educational problems. *Journal of Educational Measurement*, *21*, 361-375.

See Also

FI, itChoose, KL, mleEst, simIrt

Examples

```
## Not run:

#########################
# Binary Response Model #
#########################
set.seed(888)
# generating random theta:
theta <- rnorm(50)
# generating an item bank under a 2-parameter binary response model:
b.params <- cbind(a = runif(100, .5, 1.5), b = rnorm(100, 0, 2), c = 0)
# simulating responses:
b.resp <- simIrt(theta = theta, params = b.params, mod = "brm")$resp


## CAT 1 ##
# the typical, classic post-hoc CAT:
catStart1 <- list(init.theta = 0, n.start = 5,
                  select = "UW-FI", at = "theta",
                  n.select = 4, it.range = c(-1, 1),
                  score = "step", range = c(-1, 1),
                  step.size = 3, leave.after.MLE = FALSE)
catMiddle1 <- list(select = "UW-FI", at = "theta",
                   n.select = 1, it.range = NULL,
                   score = "MLE", range = c(-6, 6),
                   expos = "none")
catTerm1 <- list(term = "fixed", n.min = 10, n.max = 50)

cat1 <- catIrt(params = b.params, mod = "brm",
               resp = b.resp,
               catStart = catStart1,
               catMiddle = catMiddle1,
               catTerm = catTerm1)

# we can print, summarize, and plot:
cat1                                        # prints theta because
                                            # we have fewer than
                                            # 200 simulees
summary(cat1, group = TRUE, ids = "none")   # nice summary!

summary(cat1, group = FALSE, ids = 1:4)     # summarizing people too! :)

par(mfrow = c(2, 2))
plot(cat1, ask = FALSE)              # 2-parameter model, so expected FI
                                     # and observed FI are the same
par(mfrow = c(1, 1))

# we can also plot particular simulees:
```

```
par(mfrow = c(2, 1))
plot(cat1, which = "none", ids = c(1, 30), ask = FALSE)
par(mfrow = c(1, 1))


## CAT 2 ##
# using Fixed Point KL info rather than Unweighted FI to select items:
catStart2 <- catStart1
catMiddle2 <- catMiddle1
catTerm2 <- catTerm1

catStart2$leave.after.MLE <- TRUE         # leave after mixed response pattern
catMiddle2$select <- "FP-KL"
catMiddle2$at <- "bounds"
catMiddle2$delta <- .2
catTerm2$c.term <- list(bounds = 0)
cat2 <- catIrt(params = b.params, mod = "brm",
               resp = b.resp,
               catStart = catStart2,
               catMiddle = catMiddle2,
               catTerm = catTerm2)
cor(cat1$cat_theta, cat2$cat_theta)       # very close!

summary(cat2, group = FALSE, ids = 1:4)   # rarely 5 starting items!


## CAT 3/4 ##
# using "precision" rather than "fixed" to terminate:
catTerm1$term <- catTerm2$term <- "precision"
catTerm1$p.term <- catTerm2$p.term <- list(method = "threshold", crit = .3)
cat3 <- catIrt(params = b.params, mod = "brm",
               resp = b.resp,
               catStart = catStart1,
               catMiddle = catMiddle1,
               catTerm = catTerm1)
cat4 <- catIrt(params = b.params, mod = "brm",
   resp = b.resp,
   catStart = catStart2,
   catMiddle = catMiddle2,
   catTerm = catTerm2)

mean(cat3$cat_length - cat4$cat_length) # KL info results in slightly more items


## CAT 5/6 ##
# classification CAT with a boundary of 0 (with default classification stuff):
catTerm5 <- list(term = "class", n.min = 10, n.max = 50,
                 c.term = list(method = "SPRT",
                               bounds = 0, delta = .2,
                               alpha = .10, beta = .10))
cat5 <- catIrt(params = b.params, mod = "brm",
               resp = b.resp,
               catStart = catStart1,
```

```
               catMiddle = catMiddle1,
               catTerm = catTerm5)
cat6 <- catIrt(params = b.params, mod = "brm",
               resp = b.resp,
               catStart = catStart1,
               catMiddle = catMiddle2,
               catTerm = catTerm5)

# how many were classified correctly?
mean(cat5$cat_categ == cat5$tot_categ)

# using a different selection mechanism, we get the similar results:
mean(cat6$cat_categ == cat6$tot_categ)


## CAT 7 ##
# we could change estimation to EAP with the default (normal) prior:
catMiddle7 <- catMiddle1
catMiddle7$score <- "EAP"
cat7 <- catIrt(params = b.params, mod = "brm", # much slower!
               resp = b.resp,
               catStart = catStart1,
               catMiddle = catMiddle7,
               catTerm = catTerm1)
cor(cat1$cat_theta, cat7$cat_theta)            # pretty much the same


## CAT 8 ##
# let's specify the prior as something strange:
cat8 <- catIrt(params = b.params, mod = "brm",
               resp = b.resp,
               catStart = catStart1,
               catMiddle = catMiddle7,
               catTerm = catTerm1,
               ddist = dchisq, df = 4)

cat8   # all positive values of "theta"


## CAT 9 ##
# finally, we can have:
#   - more than one termination criteria,
#   - individual bounds per person,
#   - simulating based on theta without a response matrix.
catTerm9 <- list(term = c("fixed", "class"),
                 n.min = 10, n.max = 50,
                 c.term = list(method = "SPRT",
                               bounds = cbind(runif(length(theta), -1, 0),
                                              runif(length(theta), 0, 1)),
                               delta = .2,
                               alpha = .1, beta = .1))
cat9 <- catIrt(params = b.params, mod = "brm",
               resp = NULL, theta = theta,
```

```
                catStart = catStart1,
                catMiddle = catMiddle1,
                catTerm = catTerm9)

summary(cat9)   # see "... with Each Termination Criterion"


#########################
# Graded Response Model #
#########################
# generating random theta
theta <- rnorm(201)
# generating an item bank under a graded response model:
g.params <- cbind(a = runif(100, .5, 1.5), b1 = rnorm(100), b2 = rnorm(100),
                                           b3 = rnorm(100), b4 = rnorm(100))

# the graded response model is exactly the same, only slower!
cat10 <- catIrt(params = g.params, mod = "grm",
                resp = NULL, theta = theta,
                catStart = catStart1,
                catMiddle = catMiddle1,
                catTerm = catTerm1)

# warning because it.range cannot be specified for graded response models!

# if there is more than 200 simulees, it doesn't print individual thetas:
cat10


## End(Not run)

# play around with things - CATs are fun - a little frisky, but fun.
```

FI *Calculate Expected and Observed Fisher Information for IRT Models*

Description

FI calculates expected and/or observed Fisher information for various IRT models given a vector of ability values, a vector/matrix of item parameters, and an IRT model. It also calculates test information and expected/observed standard error of measurement.

Usage

```
FI( params, theta, type = c("expected", "observed"), resp = NULL )
## S3 method for class 'brm'
FI( params, theta, type = c("expected", "observed"), resp = NULL )
## S3 method for class 'grm'
FI( params, theta, type = c("expected", "observed"), resp = NULL )
```

Arguments

`params`	**numeric:** a vector or matrix of item parameters. If specified as a matrix, the rows must index the items, and the columns must designate the item parameters. Furthermore, if calculating *expected* information, the number of rows must match the number of columns of `resp`. The class of `params` must match the model: either '`"brm"`' or '`"grm"`'. For the binary response model, `params` must either be a 3-dimensional vector or a 3-column matrix. See **Details** for more information.
`theta`	**numeric:** a vector of ability values, one for each simulee. If calculating *expected* information, the length of `theta` must match the number of rows of `resp`, unless `theta` is a scalar, in which case `resp` could also be a vector of length `nrow(params)`.
`type`	**character:** a character string indicating the type of information, either '`"expected"`' or '`"observed"`'. For the 1-parameter and 2-parameter binary response model (of class '`"brm"`' with the third column of `params` set to 0), both '`"expected"`' and '`"observed"`' information are identical. See **Details** for more information.
`resp`	**numeric:** either a $N \times J$ matrix (where N indicates the number of simulees and J indicates the number of items), a N length vector (if there is only one item) or a J length vector (if there is only one simulee). For the binary response model ('`"brm"`'), `resp` must solely contain 0s and 1s. For the graded response model ('`"grm"`'), `resp` must solely contain integers $1, \ldots, K$, where K is the number of categories, as indicated by the dimension of `params`.

Details

The function `FI` returns item information, test information, and standard error of measurement for the binary response model ('`"brm"`') or the graded response model ('`"grm"`'). If the log likelihood is twice differentiable, expected Fisher information is the negative, expected, second derivative of the log likelihood with respect to the parameter. For the binary response model, expected item information simplifies to the following:

$$I(\theta_i|a_j, b_j, c_j) = \frac{\left(\frac{\partial p_{ij}}{\partial \theta_i}\right)^2}{p_{ij}(1 - p_{ij})}$$

where $\partial p_{ij}/\partial \theta_i$ is the partial derivative of p_{ij} with respect to θ, and p_{ij} is the probability of response, as indicated in the help page for `simIrt`.

For the graded response model, expected item information simplifies to the following:

$$I(\theta_i|a_j, b_{j1}, \ldots, b_{j(k-1)}) = \sum_k \frac{\left(\frac{\partial P_{ijk}}{\partial \theta_i}\right)^2}{P_{ijk}}$$

where $\partial P_{ijk}/\partial \theta_i$ is the partial derivative of P_{ijk} with respect to θ, and P_{ijk} is the probability of responding in category k as indicated in the help page for `simIrt`. See van der Linden and Pashley (2010).

Observed information is the negative second derivative of the log-likelihood. For the binary response model (‘"brm"’) with 2-parameters (such that the third column of the parameter matrix is set to 0), observed and expected information are identical because the second derivative of their log-likelihoods do not contain observed data. See Baker and Kim (2004), pp. 66 – 69.

For all models, test information is defined as the following:

$$T(\theta_i) = \sum_j I_j(\theta_i)$$

where $I(\theta_i)_j$ is shorthand for Fisher information of simulee i on item j. Finally, the standard error of measurement (SEM) is the inverse, square-root of test information. `FI` is frequently used to *select* items in a CAT and to estimate the precision of $\hat{\theta}_i$ after test termination.

Value

`FI`, `FI.brm`, and `FI.grm` return a list of the following elements:

`item`	either: (1) a $N \times J$ matrix of item information for each simulee to each item; (2) a J-length vector of item information for one simulee to each item; or (3) an N-length vector of item information for all simulees to one item, depending on the dimensions of `params` and `theta`.
`test`	an N-length vector of test information, one for each simulee. Test information is the sum of item information across items. See **Details** for more information.
`sem`	an N-length vector of expected or observed standard error of measurement for each simulee, which is the inverse-square-root of test information. See **Details** for more information.
`type`	either ‘"observed"’ or ‘"expected"’, indicating the *type* of information calculated.

Author(s)

Steven W. Nydick <nydic001@umn.edu>

References

Baker, F. B., & Kim, S.-H. (2004). *Item Response Theory: Parameter Estimation Techniques, Second Edition*. New York, NY: Marcel Dekker, Inc.

Dodd, B. G., De Ayala, R. J., & Koch, W. R. (1995). Computerized adaptive testing with polytomous items. *Applied Psychological Measurement*, *19*, 5 – 22.

Embretson, S. E., & Reise, S. P. (2000). *Item Response Theory for Psychologists*. Mahway, NJ: Lawrence Erlbaum Associates.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*, 79 – 86.

van dr Linden, W. J. & Pashley, P. J. (2010). Item selection and ability estimation in adaptive testing. In W. J. van der Linden & C. A. W. Glas (Eds.), *Elements of Adaptive Testing*. New York, NY: Springer.

See Also

catIrt, KL, simIrt

Examples

```
#########################
# Binary Response Model #
#########################

## 1 ##
set.seed(888)
# generating random theta:
theta <- rnorm(20)
# generating an item bank under a 2-parameter binary response model:
b.params <- cbind(a = runif(100, .5, 1.5), b = rnorm(100, 0, 2), c = 0)
# simulating responses using random theta:
b.mod <- simIrt(params = b.params, theta = theta, mod = "brm")

# you can indicate class of params or extract it from simIrt object:
class(b.params) <- "brm"

# calculating expected and observed information:
e.info <- FI(params = b.params, theta = theta, type = "expected")
o.info <- FI(params = b.params, theta = theta, type = "observed", resp = b.mod$resp)

# 2-parameter model, so e.info will be equal to o.info:
all(signif(e.info$item) == signif(o.info$item))


## 2 ##
# generating an item bank under a 3-parameter binary response model:
b.params2 <- cbind(a = runif(100, .5, 1.5), b = rnorm(100, 0, 2), c = .2)
# simulating responses using pre-specified thetas:
b.mod2 <- simIrt(params = b.params2, mod = "brm")

# calculating expected and observed information:
# (if you don't indicate class, you can extract from simIrt object)
e.info2 <- FI(params = b.params2, theta = b.mod2$theta, type = "expected")
o.info2 <- FI(params = b.params2, theta = b.mod2$theta, type = "observed",
                                  resp = b.mod2$resp)

# 3-parameter model, so e.info will not be equal to o.info:
all(signif(e.info2$item) == signif(o.info2$item))


## 3 ##
# if theta is a scalar, item will be a vector and test will be a scalar:
e.info3 <- FI(params = b.params2, theta = 0, type = "expected")
dim(e.info3$item)       # no dimension because it's a vector
length(e.info3$item)    # of length equal to the number of items

# if params is a vector, item will be a matrix with one row:
```

```
e.info4 <- FI(params = c(1, 2, 0), theta = c(1, 2), type = "expected")
dim(e.info4$item)

# if you don't class params, FI will assume a binary response model.


#########################
# Graded Response Model #
#########################
set.seed(999)
# generating random theta
theta <- rnorm(10)
# generating an item bank under a graded response model:
g.params <- cbind(a = runif(30, .5, 1.5), b1 = rnorm(30), b2 = rnorm(30),
                                          b3 = rnorm(30), b4 = rnorm(30))
# you can sort the parameters yourself:
g.params <- cbind(g.params[ , 1],
                  t(apply(g.params[ ,2:dim(g.params)[2]], MARGIN = 1,
                                                         FUN = sort)))
# simulating responses using random theta:
g.mod <- simIrt(params = g.params, theta = theta, mod = "grm")

# calculating expected and observed information:
class(g.params) <- "grm"   # always indicate model or extract from simulation.
e.info5 <- FI(params = g.params, theta = theta, type = "expected")
o.info5 <- FI(params = g.params, theta = theta, type = "observed", resp = g.mod$resp)

# grm, so e.info will not be equal to o.info:
all(signif(e.info5$item) == signif(o.info5$item))

# if thet is a vector and params is a vector, item will be a J x N matrix:
dim(e.info5$item)

# if you don't want to sort the parameters, you can extract from simIrt object:
e.info6 <- FI(params = g.mod$params[ , -1], theta = g.mod$theta, type = "expected")

# but you first need to remove column 1 (the item number column).
```

---

itChoose *Choose the Next Item in a CAT*

Description

itChoose chooses the next item in a CAT based on the remaining items and a variety of item selection algorithms.

Usage

```
itChoose( left_par, mod = c("brm", "grm"),
          numb = 1, n.select = 1,
```

```
            cat_par = NULL, cat_resp = NULL, cat_theta = NULL,
            select = c("UW-FI", "LW-FI", "PW-FI",
                       "FP-KL", "VP-KL", "FI-KL", "VI-KL",
                       "random"),
            at = c("theta", "bounds"),
            range = c(-6, 6), it.range = NULL,
            delta = NULL, bounds = NULL,
            ddist = dnorm, quad = 33, ... )
```

Arguments

`left_par`	**numeric:** a matrix of item parameters from which to choose the next item. The rows must index the items, and the columns must designate the item parameters (in the appropriate order, see `catIrt`). The first column of `left_par` must indicate the *item numbers*, as `itChoose` returns not *just* the item parameters but also the bank number corresponding to those parameters. See **Details** for more information.
`mod`	**character:** a character string indicating the IRT model. Current support is for the 3-parameter binary response model (‘`"brm"`’), and Samejima's graded response model (‘`"grm"`’). The contents of `params` must match the designation of `mod`. See `catIrt` or `simIrt` for more information.
`numb`	**numeric:** a scalar indicating the number of items to *return* to the user. If `numb` is less than `n.select`, then `itChoose` will randomly select `numb` items from the top `n.select` items according to the item selection algorithm.
`n.select`	**numeric:** an *integer* indicating the number of items to randomly select between at one time. For instance, if `select` is ‘`"UW-FI"`’, `at` is ‘`"theta"`’, `numb` is 3, and `n.select` is 8, then `itChoose` will randomly select *3* items out of the top *8* items that maximize Fisher information at `cat_theta`.
`cat_par`	**numeric:** either NULL or a matrix of item parameters that have already been administered in the CAT. `cat_par` only needs to be specified if letting `select` equal either ‘`"LW-FI"`’, ‘`"PW-FI"`’, ‘`"VP-KL"`’ or ‘`"VI-KL"`’. The format of `cat_par` must be the same as the format of `left_par`. See **Details** for more information.
`cat_resp`	**numeric:** either NULL or a vector of responses corresponding to the items specified in `cat_par`. `cat_par` only needs to be specified if letting `select` equal either ‘`"LW-FI"`’ or ‘`"PW-FI"`’.
`cat_theta`	**numeric:** either NULL or a scalar corresponding to the current ability estimate. `cat_theta` is not needed if selecting items at ‘`"bounds"`’ or using ‘`"LW-FI"`’ or ‘`"PW-FI"`’ as the item selection algorithm.
`select`	**character:** a character string indicating the desired item selection method. Items can be selected either through maximum Fisher information or Kullback-Leibler divergence methods or randomly. The Fisher information methods include • ‘`"UW-FI"`’: unweighted Fisher information at a point. • ‘`"LW-FI"`’: Fisher information weighted across the likelihood function. • ‘`"PW-FI"`’: Fisher information weighted across the posterior distribution of θ.

And the Kullback-Leibler divergence methods include

- '"FP-KL"': pointwise KL divergence between [P +/- delta], where P is either the current θ estimate or a classification bound.
- '"VP-KL"': pointwise KL divergence between [P +/- delta/sqrt(n)], where n is the number of items given to this point in the CAT.
- '"FI-KL"': KL divergence integrated along [P -/+ delta] with respect to P
- '"VI-KL"': KL divergence integrated along [P -/+ delta/sqrt(n)] with respect to P.

See **Details** for more information.

`at` **character:** a character string indicating where to select items.

`range` **numeric:** a 2-element numeric vector indicating the range of values that `itChoose` should average over if `select` equals '"LW-FI"' or '"PW-FI"'.

`it.range` **numeric:** Either a 2-element numeric vector indicating the minimum and maximum allowed difficulty parameters for selected items (only if `mod` is equal to '"brm"') or NULL indicating no item parameter restrictions.

`delta` **numeric:** a scalar indicating the multiplier used in item selection if a Kullback-Leibler method is chosen. For fixed-point KL divergence, `delta` is frequently .1 or .2, whereas in variable-point KL divergence, `delta` usually corresponds to 95 or 97.5 percentiles on a normal distribution.

`bounds` **numeric:** a vector of fixed-points/bounds from which to select items if `at` equals '"bounds"'.

`ddist` **function:** a function indicating how to calculate prior densities if `select` equals '"PW-FI"' (i.e., weighting Fisher information on the posterior distribution). See `catIrt` for more information.

`quad` **numeric:** a scalar indicating the number of quadrature points when `select` equals '"LW-FI"' or '"PW-FI"'. See **Details** for more information.

`...` arguments passed to `ddist`, usually distribution parameters identified by name.

Details

The function `itChoose` returns the next item(s) to administer in a CAT environment. The item selection algorithms fall into three major types: Fisher information, Kullback-Leibler divergence, and random.

- If choosing items based on Fisher information (`select` equals '"UW-FI"', '"LW-FI"', or '"PW-FI"'), then items are selected based on some aggregation of Fisher information (see `FI`). The difference between the three Fisher information methods are the weighting functions used (see van der Linden, 1998; Veerkamp & Berger, 1997). Let

$$I(w_{ij}|a_j, b_j, c_j) = \int_{-\infty}^{\infty} w_{ij} I_j(\theta) \mu(d\theta)$$

be the "average" Fisher information, weighted by real valued function w_{ij}. Then all three Fisher information criteria can be explained solely as using different weights. Unweighted Fisher information ('"UW-FI"') sets w_{ij} equal to a Dirac delta function with all of its mass

either on theta (if at equals '"theta"') or the nearest classification bound (if at equals '"bounds"'). Likelihood-Weighted Fisher information ('"UW-FI"') sets w_{ij} equal to the likelihood function given all of the previously administered items (Veerkamp & Berger, 1997). And Posterior-Weighted Fisher information ('"PW-FI"') sets w_{ij} equal to the likelihood function times the prior distribution specified in ddist (van der Linden, 1998). All three algorithms select items based on maximizing the respective criterion with '"UW-FI"' the most popular CAT item selection algorithm and equivalent to maximizing Fisher information at a point (Pashley & van der Linden, 2010).

- If choosing items based on Kullback-Leibler divergence (select equals '"FP-KL"', '"VP-KL"', '"FI-KL"', or '"VI-KL"'), then items are selected based on some aggregation of KL divergence (see KL).
  - The Pointwise KL divergence criteria (select equals '"FP-KL"' and '"VP-KL"') compares KL divergence at two points:

$$KL(w_{ij}|a_j, b_j, c_j) = KL_j(P + w_{ij}||P - w_{ij})$$

    The difference between '"FP-KL"' and '"VP-KL"' are the weights used. Fixed Pointwise KL divergence ('"FP-KL"') sets w_{ij} equal to 'delta', and Variable Pointwise KL divergence ('"VP-KL"') sets w_{ij} equal to 'delta' multiplied by $1/\sqrt{n}$, where n is equal to the number of items given to this point in the CAT (see Chang & Ying, 1996).
  - The Integral KL divergence criteria (select equals '"FI-KL"' and '"VI-KL"') integrates KL divergence across a small area:

$$KL(w_{ij}|a_j, b_j, c_j) = \int_{P-w_{ij}}^{P+w_{ij}} KL_j(\theta||P)d\theta$$

    As in Pointwise KL divergence, Fixed Integral KL divergence ('"FI-KL"') sets w_{ij} equal to 'delta', and Variable Integral KL divergence ('"VI-KL"') sets w_{ij} equal to 'delta' multiplied by $1/\sqrt{n}$ (see Chang & Ying, 1996).

  All KL divergence criteria set P equal to theta (if at equals '"theta"') or the nearest classification bound (if at equals '"bounds"') and select items based on maximizing the respective criterion.
- If select is '"random"', then itChoose randomly picks the next item(s) out of the remaining items in the bank.

Value

itChoose returns a list of the following elements:

params	a matrix corresponding to the next item or numb items to administer in a CAT with the first column indicating the item number
info	a vector of corresponding information for the numb items of params.
type	the type of information returned in info, which is equal to the item selection algorithm.

Author(s)

Steven W. Nydick <nydic001@umn.edu>

References

Chang, H.-H., & Ying, Z. (1996). A global information approach to computerized adaptive testing. *Applied Psychological Measurement*, *20*, 213 – 229.

Pashley, P. J., & van der Linden, W. J. (2010). Item selection and ability estimation in adaptive testing. In W. J. van der Linden & C. A. W. Glas (Eds.), *Elements of adaptive testing* (pp. 3 – 30). New York, NY: Springer.

van der Linden, W. J. (1998). Bayesian item selection criteria for adaptive testing. *Psychometrika*, *63*, 201 – 216.

Veerkamp, W. J. J., & Berger, M. P. F. (1997). Some new item selection criteria for adaptive testing. *Journal of Educational and Behavioral Statistics*, *22*, 203 – 226.

See Also

catIrt, FI, KL, mleEst, simIrt

Examples

```
#########################
# Binary Response Model #
#########################
## Not run:
set.seed(888)
# generating an item bank under a binary response model:
b.params <- cbind(a = runif(100, .5, 1.5), b = rnorm(100, 0, 2), c = .2)
# simulating responses using default theta:
b.mod <- simIrt(theta = 0, params = b.params, mod = "brm")

# separating the items into "administered" and "not administered":
left_par <- b.mod$params[1:95, ]
cat_par <- b.mod$params[96:100, ]
cat_resp <- b.mod$resp[ , 96:100]

# running simIrt automatically adds the item numbers to the front!

# attempting each item selection algorithm (except random):
uwfi.it <- itChoose(left_par = left_par, mod = "brm",
                    numb = 1, n.select = 1,
                    cat_theta = 0,
                    select = "UW-FI",
                    at = "theta")
lwfi.it <- itChoose(left_par = left_par, mod = "brm",
                    numb = 1, n.select = 1,
                    cat_par = cat_par, cat_resp = cat_resp,
                    select = "LW-FI")
pwfi.it <- itChoose(left_par = left_par, mod = "brm",
                    numb = 1, n.select = 1,
                    cat_par = cat_par, cat_resp = cat_resp,
                    select = "PW-FI", ddist = dnorm, mean = 0, sd = 1)

fpkl.it <- itChoose(left_par = left_par, mod = "brm",
```

```
                    numb = 1, n.select = 1,
                    cat_theta = 0,
                    select = "FP-KL",
                    at = "theta", delta = 1.96)
vpkl.it <- itChoose(left_par = left_par, mod = "brm",
                    numb = 1, n.select = 1,
                    cat_par = cat_par, cat_theta = 0,
                    select = "VP-KL",
                    at = "theta", delta = 1.96)
fikl.it <- itChoose(left_par = left_par, mod = "brm",
                    numb = 1, n.select = 1,
                    cat_theta = 0,
                    select = "FI-KL",
                    at = "theta", delta = 1.96)
vikl.it <- itChoose(left_par = left_par, mod = "brm",
                    numb = 1, n.select = 1,
                    cat_par = cat_par, cat_theta = 0,
                    select = "VI-KL",
                    at = "theta", delta = 1.96)

# which items were the most popular?
uwfi.it$params  # 61 (b close to 0)
lwfi.it$params  # 55 (b close to -2.5)
pwfi.it$params  # 16 (b close to -0.5)
fpkl.it$params  # 61 (b close to 0)
vpkl.it$params  # 61 (b close to 0)
fikl.it$params  # 16 (b close to -0.5)
vikl.it$params  # 16 (b close to -0.5)

# if we pick the top 10 items for "FI-KL":
fikl.it2 <- itChoose(left_par = left_par, mod = "brm",
                     numb = 10, n.select = 10,
                     cat_theta = 0,
                     select = "FI-KL",
                     at = "theta", delta = 1.96)

# we find that item 61 is the third best item
fikl.it2$params

# why did "LW-FI" pick an item with a strange difficulty?
cat_resp

# because cat_resp is mostly 0 ...
# --> so the likelihood is weighted toward negative numbers.

#########################
# Graded Response Model #
#########################
set.seed(999)
# generating an item bank under a graded response model:
g.params <- cbind(runif(100, .5, 1.5), rnorm(100), rnorm(100),
                                       rnorm(100), rnorm(100), rnorm(100))
# simulating responses (so that the parameters are ordered - see simIrt)
```

```
left_par <- simIrt(theta = 0, params = g.params, mod = "grm")$params

# now we can choose the best item for theta = 0 according to FI:
uwfi.it2 <- itChoose(left_par = left_par, mod = "brm",
                     numb = 1, n.select = 1,
                     cat_theta = 0,
                     select = "UW-FI",
                     at = "theta")
uwfi.it2

## End(Not run)
```

KL *Calculate Kullback-Leibler Divergence for IRT Models*

Description

KL calculates the IRT implementation of Kullback-Leibler divergence for various IRT models given a vector of ability values, a vector/matrix of item responses, an IRT model, and a value indicating the half-width of an indifference region.

Usage

```
KL( params, theta, delta = .1 )
## S3 method for class 'brm'
KL( params, theta, delta = .1 )
## S3 method for class 'grm'
KL( params, theta, delta = .1 )
```

Arguments

params	**numeric:** a vector or matrix of item parameters. If specified as a matrix, the rows must index the items, and the columns must designate the item parameters. Furthermore, if calculating *expected* information, the number of rows must match the number of columns of resp. The class of params must match the model: either '"brm"' or '"grm"'. For the binary response model, params must either be a 3-dimensional vector or a 3-column matrix. See **Details** for more information.
theta	**numeric:** a vector of ability values, one for each simulee. When performing a classification CAT, theta should be the boundary points for which to choose the next item.
delta	**numeric:** a scalar or vector indicating the half-width of the indifference KL will estimate the divergence between $\theta - \delta$ and $\theta + \delta$ using $\theta + \delta$ as the "true model." If delta is a vector, then KL will use recycling to make the length of theta and delta match. See **Details** for more information.

Details

The function KL returns item divergence and test divergence for the binary response model (‘"brm"’) and the graded response model (‘"grm"’). KL-divergence is defined as the following:

$$KL(\theta_2||\theta_1) = E_{\theta_2} \log \left[\frac{L(\theta_2)}{L(\theta_1)} \right]$$

where $L(\theta)$ stands for the likelihood of θ. Essentially, KL-divergence is the expected log-likelihood gain when using the true model in place of an alternative model.

For the binary response model, KL-divergence for an item simplifies to the following:

$$KL_j(\theta_2||\theta_1)_j = p_j(\theta_2) \log \left[\frac{p_j(\theta_2)}{p_j(\theta_1)} \right] + [1 - p_j(\theta_2)] \log \left[\frac{1 - p_j(\theta_2)}{1 - p_j(\theta_1)} \right]$$

where p_{ij} is the probability of response, as indicated in the help page for simIrt

For the graded response model, KL-divergence for an item simplifies to the following:

$$KL_j(\theta_2||\theta_1) = \sum_k P_{jk}(\theta_2) \log \left[\frac{P_{jk}(\theta_2)}{P_{jk}(\theta_1)} \right]$$

where $P_{jk}(\theta_2)$ is the probability of θ_2 responding in category k as indicated in the help page for simIrt. See Eggen (1999) as applied to classification CAT and van der Linden and Pashley (2010) more generally.

Because of the properties of likelihood functions in item response models, test information is simply the sum of the item informations, or:

$$KL(\theta_2||\theta_1) = \sum_j KL_j(\theta_2||\theta_1)$$

KL is frequently used to select items in a classification CAT where the hypotheses (e.g. being in one category versus another category are well defined). If "being in the upper category" is θ_2 and "being in the lower category" is θ_1, then $\theta_2 = B + \delta$ and $\theta_1 = B - \delta$ where B is the boundary separating the lower category from the upper category. Conversely, if using KL to select items in a precision CAT, then $\theta_2 = \hat{\theta}_i + \delta$ and $\theta_1 = \hat{\theta}_i$ where $\hat{\theta}_i$ is the current, *best* estimate of θ. See catIrt for more information.

Value

KL, KL.brm, and KL.grm return a list of the following elements:

item	either: (1) a $N \times J$ matrix of item information for each simulee to each item; (2) a J-length vector of item information for one simulee to each item; or (3) an N-length vector of item information for all simulees to one item, depending on the dimensions of params, theta, annd delta.
test	an N-length vector of test information, one for each simulee. Test information is the sum of item information across items. See **Details** for more information.

Note

Kullback-Leibler divergence in IRT is not *true* KL divergence, as the expectation is with respect to a model that is not necessarily true. Furthermore, it is not reciprocal, as $KL(\theta_1||\theta_2) \neq KL(\theta_2||\theta_1)$. There have been other KL-based item selection measures proposed, including global information. See Eggen (1999) and `itChoose`.

Author(s)

Steven W. Nydick <nydic001@umn.edu>

References

Eggen, T. J. H. M. (1999). Item selection in adaptive testing with the sequential probability ratio test. *Applied Psychological Measurement*, *23*, 249 – 261.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*, 79 – 86.

van dr Linden, W. J. & Pashley, P. J. (2010). Item selection and ability estimation in adaptive testing. In W. J. van der Linden & C. A. W. Glas (Eds.), *Elements of Adaptive Testing*. New York, NY: Springer.

See Also

`catIrt`, `FI`, `itChoose`, `simIrt`

Examples

```
#########################
# Binary Response Model #
#########################

## 1 ##
set.seed(888)
# generating random theta:
theta <- rnorm(20)
# generating an item bank under a 3-parameter binary response model:
b.params <- cbind(a = runif(100, .5, 1.5), b = rnorm(100, 0, 2), c = .2)

# you can indicate class of params or extract it from simIrt object:
class(b.params) <- "brm"

# calculating KL information with delta = .1:
k.info1 <- KL(params = b.params, theta = theta, delt = .1)
# changing delta to .2
k.info2 <- KL(params = b.params, theta = theta, delt = .2)

# notice how the overall information has increased when increasing delt:
k.info1$test; k.info2$test

# also compare with Fisher information:
f.info <- FI(params = b.params, theta = theta, type = "expected")
```

```
k.info2$test; f.info$test

# Fisher information is much higher because of how it weighs things.

## 2 ##
# we can maximize information at a boundary - say "0":
k.info3 <- KL(params = b.params, theta = 0, delta = .1)
b.params[which.max(k.info3$item), ]

# notice how the a parameter is high while the b parameter is close to
# 0, so item selection is working.

# does Fisher information choose a different item?
f.info2 <- FI(params = b.params, theta = 0, type = "expected")
b.params[which.max(f.info2$item), ]

# nope - although with more items, who knows?


#########################
# Graded Response Model #
#########################

## 1 ##
set.seed(999)
# generating random theta
theta <- rnorm(20)
# generating an item bank under a graded response model:
g.params <- cbind(runif(100, .5, 1.5), rnorm(100), rnorm(100),
                                       rnorm(100), rnorm(100), rnorm(100))
# simulating responses (so that the parameters are ordered - see simIrt)
g.params <- simIrt(theta = theta, params = g.params, mod = "grm")$params[ , -1]

# we can calculate KL information as before, noting that class(g.params) is "grm"
class(g.params)     # so we don't need to set it ourselves

# and now KL info with delt = .1
k.info4 <- KL(theta = theta, params = g.params)

# KL information is higher when more boundaries
k.info4$test
k.info1$test

# Note: k.info1 would be exactly the same if calculated with the "grm"
#       rather than the "brm"

## 2 ##
# we can also maximize information at boundary "0"
k.info5 <- KL(params = g.params, theta = 0, delta = .1)
g.params[which.max(k.info5$item), ]

# notice how the a parameter is high while the b parameters are pretty spread out.
```

```
# does Fisher information choose a different item?
f.info3 <- FI(params = g.params, theta = 0, type = "expected")
g.params[which.max(f.info3$item), ]

# nope - although with more items, who knows?
```

`mleEst` *Estimate Ability in IRT Models*

Description

`mleEst`, `wleEst`, `bmeEst`, and `eapEst` estimate ability in IRT models. `mleEst` is Maximum Likelihood Information, `wleEst` is Weighted Likelihood Information (see **Details**), `bmeEst` is Bayesian-Modal Estimation, and `eapEst` is Expected-A-Posterior Estimation.

Usage

```
mleEst( resp, params, range = c(-6, 6), mod = c("brm", "grm"), ... )
wleEst( resp, params, range = c(-6, 6), mod = c("brm", "grm"), ... )
bmeEst( resp, params, range = c(-6, 6), mod = c("brm", "grm"),
        ddist = dnorm, ... )
eapEst( resp, params, range = c(-6, 6), mod = c("brm", "grm"),
        ddist = dnorm, quad = 33, ... )
```

Arguments

`resp`	**numeric:** either a $N \times J$ matrix (where N indicates the number of simulees and J indicates the number of items), a N length vector (if there is only one item) or a J length vector (if there is only one simulee). For the binary response model (‘`"brm"`’), `resp` must solely contain 0s and 1s. For the graded response model (‘`"grm"`’), `resp` must solely contain integers $1, \ldots, K$, where K is the number of categories, as indicated by the dimension of `params`.
`params`	**numeric:** a vector or matrix of item parameters. If specified as a matrix, the rows must index the items, and the columns must designate the item parameters.
`range`	**numeric:** a two-element numeric vector indicating the minimum and maximum over which to optimize a likelihood function (`mleEst`) or posterior distribution (`bmeEst`), find roots to a score function (`wleEst`), or integrate over (`eapEst`).
`mod`	**character:** a character string indicating the IRT model. Current support is for the 3-parameter binary response model (‘`"brm"`’), and Samejima's graded response model (‘`"grm"`’). See `simIrt` for more information.
`ddist`	**function:** a function that calculates prior densities for Bayesian estimation. For instance, if you wish to specify a normal prior, `ddist = dnorm`, and if you wish to specify a uniform prior, `ddist = dunif`. Note that it is standard in R to use `d...` to indicate a density.
`quad`	**numeric:** a scalar indicating the number of quadrature points when using `eapEst`. See **Details** for more information.
`...`	arguments passed to `ddist`, usually distribution parameters identified by name.

Details

These functions return estimated "ability" for the binary response model (‘"brm"’) and the graded response model (‘"grm"’). The only difference between the functions is how they estimate ability.

The function `mleEst` searches for a maximum of the log-likelihood with respect to each individual θ_i and uses $[T(\theta)]^{-1/2}$ as the corresponding standard error of measurement (SEM), where $T(\theta)$ is the observed test information function at θ, as described in `FI`.

The function `bmeEst` searches for the maximum of the log-likelihood after a log-prior is added, which effectively maximizes the posterior distribution for each individual θ_i. The SEM of the `bmeEst` estimator uses the well known relationship (Keller, 2000, p. 10)

$$V[\theta|\boldsymbol{u}_i]^{-1} = T(\theta) - \frac{\partial \log[p(\theta)]}{\partial \theta^2}$$

where $V[\theta|\boldsymbol{u}_i]$ is the variance of θ after taking into consideration the prior distribution and $p(\theta)$ is the prior distribution of θ. The function `bmeEst` estimates the second derivative of the prior distribution uses the `hessian` function in the `numDeriv` package.

The function `wleEst` searches for the root of a modified score function (i.e. the first derivative of the log-likelihood with something added to it). The modification corrects for bias in fixed length tests, and estimation using this modification results in what is called Weighted Maximum Likelihood (or alternatively, the Warm estimator) (see Warm, 1989). So rather than maximizing the likelihood, `wleEst` finds a root of:

$$\frac{\partial l(\theta)}{\partial \theta} + \frac{H(\theta)}{2I(\theta)}$$

where $l(\theta)$ is the log-likelihood of θ given a set of responses and item parameters, $I(\theta)$ is expected test information to this point, and $H(\theta)$ is a correction constant defined as:

$$H(\theta) = \sum_j \frac{p'_{ij} p''_{ij}}{p_{ij}[1 - p_{ij}]}$$

for the binary response model, where p'_{ij} is the first derivative of p_{ij} with respect to θ, p''_{ij} is the second derivative of p_{ij} with respect to θ, and p_{ij} is the probability of response, as indicated in the help page for `simIrt`, and

$$H(\theta) = \sum_j \sum_k \frac{P'_{ijk} P''_{ijk}}{P_{ijk}}$$

for the graded response model, where P'_{ijk} is the first derivative of P_{ijk} with respect to θ, P''_{ijk} is the second derivative of P_{ijk}, and P_{ijk} is the probability of responding in category k as indicated in the help page for `simIrt`. The SEM of the `wleEst` estimator uses an approximation based on Warm (1989, p. 449):

$$V(\theta) \approx \frac{T(\theta) + \frac{H(\theta)}{2I(\theta)}}{T^2(\theta)}$$

The function eapEst finds the mean and standard deviation of the posterior distribution given the log-likelihood, a prior distribution (with specified parameters), and the number of quadrature points using the standard Bayesian identity with summations in place of integrations (see Bock and Mislevy, 1982). Rather than using the adaptive, quadrature based integrate, eapEst uses the flexible integrate.xy function in the sfsmisc package. As long as the prior distribution is reasonable (such that the joint distribution is relatively smooth), this method should work.

Value

mleEst, wleEst, bmeEst, and eapEst return a list of the following elements:

theta	an N-length vector of ability values, one for each simulee.
info	an N-length vector of observed test information, one for each simulee. Test information is the sum of item information across items. See FI for more information.
sem	an N-length vector of observed standard error of measurement (or posterior standard deviation) for each simulee. See FI for more information.

Note

For the binary response model ('"brm"'), it makes no sense to estimate ability with a non-mixed response pattern (all 0s or all 1s). The user might want to include enough items in the model to allow for reasonable estimation.

Weighted likelihood estimation (wleEst) uses uniroot to find the root of the modified score function, so that the end points of 'range' must evaluate to opposite signs (or zero). Rarely, the end points of 'range' will evaluate to the same sign, so that uniroot will error. In these cases, uniroot will extend the interval until the end points of the (modified) range are opposite signs.

Author(s)

Steven W. Nydick <nydic001@umn.edu>

References

Bock, R. D., & Mislevy, R. J. (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement*, *6*, 431 – 444.

Embretson, S. E., & Reise, S. P. (2000). *Item Response Theory for Psychologists*. Mahway, NJ: Lawrence Erlbaum Associates.

Keller (2000). *Ability estimation procedures in computerized adaptive testing* (Technical Report). New York, NY: American Institute of Certified Public Accountants.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, *54*, 427 – 450.

van dr Linden, W. J. & Pashley, P. J. (2010). Item selection and ability estimation in adaptive testing. In W. J. van der Linden & C. A. W. Glas (Eds.), *Elements of Adaptive Testing*. New York, NY: Springer.

See Also

catIrt, eap, hessian, mlebme, simIrt, uniroot

Examples

```
## Not run:

#########################
# Binary Response Model #
#########################
set.seed(888)
# generating random theta:
theta <- rnorm(201)
# generating an item bank under a 2-parameter binary response model:
b.params <- cbind(a = runif(100, .5, 1.5), b = rnorm(100, 0, 2), c = 0)
# simulating responses using specified theta:
b.resp <- simIrt(theta = theta, params = b.params, mod = "brm")$resp


# estimating theta using all four methods:
est.mle1 <- mleEst(resp = b.resp, params = b.params, mod = "brm")$theta
est.wle1 <- wleEst(resp = b.resp, params = b.params, mod = "brm")$theta
est.bme1 <- bmeEst(resp = b.resp, params = b.params, mod = "brm",
                   ddist = dnorm, mean = 0, sd = 1)$theta
est.eap1 <- eapEst(resp = b.resp, params = b.params, mod = "brm",
                   ddist = dnorm, mean = 0, sd = 1, quad = 33)$theta

# eap takes a while!

# all of the methods are highly correlated:
cor(cbind(theta = theta, mle = est.mle1, wle = est.wle1,
                         bme = est.bme1, eap = est.eap1))

# you can force eap to be positive:
est.eap2 <- eapEst(resp = b.resp, params = b.params, range = c(0, 6),
                                  mod = "brm", ddist = dunif, min = 0, max = 6)$theta

est.eap2

# if you only have a single response, MLE will give junk!
mleEst(resp = 0, params = c(1, 0, .2), mod = "brm")$theta

# the others will give you answers that are not really determined by the response:
wleEst(resp = 0, params = c(1, 0, .2), mod = "brm")$theta
bmeEst(resp = 0, params = c(1, 0, .2), mod = "brm")$theta
eapEst(resp = 0, params = c(1, 0, .2), mod = "brm")$theta


#########################
# Graded Response Model #
#########################
set.seed(999)
```

```
# generating random theta
theta <- rnorm(400)
# generating an item bank under a graded response model:
g.params <- cbind(a = runif(100, .5, 1.5), b1 = rnorm(100), b2 = rnorm(100),
                                           b3 = rnorm(100), b4 = rnorm(100))
# simulating responses using random theta:
g.mod <- simIrt(params = g.params, theta = theta, mod = "grm")

# pulling out the responses and the parameters:
g.params2 <- g.mod$params[ , -1]       # now the parameters are sorted
g.resp2 <- g.mod$resp

# estimating theta using all four methods:
est.mle3 <- mleEst(resp = g.resp2, params = g.params2, mod = "grm")$theta
est.wle3 <- wleEst(resp = g.resp2, params = g.params2, mod = "grm")$theta
est.bme3 <- bmeEst(resp = g.resp2, params = g.params2, mod = "grm",
                   ddist = dnorm, mean = 0, sd = 1)$theta
est.eap3 <- eapEst(resp = g.resp2, params = g.params2, mod = "grm",
                   ddist = dnorm, mean = 0, sd = 1, quad = 33)$theta

# and the correlations are still pretty high:
cor(cbind(theta = theta, mle = est.mle3, wle = est.wle3,
                         bme = est.bme3, eap = est.eap3))

# note that the graded response model is just a generalization of the brm:
cor(est.mle1, mleEst(resp = b.resp + 1, params = b.params[ , -3], mod = "grm")$theta)
cor(est.wle1, wleEst(resp = b.resp + 1, params = b.params[ , -3], mod = "grm")$theta)
cor(est.bme1, bmeEst(resp = b.resp + 1, params = b.params[ , -3], mod = "grm")$theta)
cor(est.eap1, eapEst(resp = b.resp + 1, params = b.params[ , -3], mod = "grm")$theta)


## End(Not run)
```

`simIrt` *Simulate Responses to IRT Models*

Description

`simIrt` simulates responses to various IRT models given a vector of ability values and a vector/matrix of item parameters.

Usage

```
simIrt( theta = seq(-3, 3, by = 0.1), params, mod = c("brm", "grm") )
```

Arguments

`theta`	**numeric:** a vector of ability values, one for each simulee.

`params` **numeric:** a vector or matrix of item parameters. If specified as a matrix, the rows must index the items, and the columns must designate the item parameters. For the binary response model, (‘`"brm"`’), `params` must either be a 3-element vector or a 3-column matrix. See **Details** for more information.

`mod` **character:** a character string indicating the IRT model. Current support is for the 3-parameter binary response model (‘`"brm"`’), and Samejima's graded response model (‘`"grm"`’). The contents of `params` must match the designation of `mod`. See **Details** for more information.

Details

The function `simIrt` returns a response matrix of class "brm" or "grm" depending on the model. For the binary response model, the probability of endorsing item j for simulee i is the following (Embretson & Reise, 2000):

$$p_{ij} = Pr(u_{ij} = 1|\theta_i, a_j, b_j, c_j) = c_j + (1 - c_j)\frac{1}{1 + \exp[-a(\theta - b)]}$$

For the graded response model, the probability of endorsing at or above boundary k of item j for simulee i is the following:

$$p_{ijk} = Pr(u_{ij} \geq k|\theta_i, a_j, b_k) = \frac{1}{1 + \exp[-a(\theta - b_k)]}$$

so that the probability of scoring *in* category k is, $P_{ijk} = Pr(u_{ij} = k|\theta_i, a_j, \boldsymbol{b}) = 1 - p_{ijk}$ if $k = 1$; p_{ijk} if $k = K$; and $p_{ij(k-1)} - p_{ijk}$ otherwise, where K is the number of categories, so that $K - 1$ is the number of boundaries.

Assuming perfect model fit, `simIrt` generates the probability of responding in a category, simulates a random, uniform deviate, and compares the probability of response with the location of the deviate. For instance, for the binary response model, if $p_{ij} = .7$, so that $q_{ij} = 1 - p_{ij} = .3$, `simIrt` will generate a uniform deviate (u_{ij}) between 0 and 1. If $u_{ij} < p_{ij}$, the simulee will score a 1, and otherwise, the simulee will score a 0.

Value

The function `simIrt` returns a list of the following elements:

`resp` a matrix of class "brm" or "grm" depending on the model used. The dimensions of the matrix will be $N \times J$ (persons by items), and will contain 0s and 1s for the binary response model or $1 \ldots K$ for the graded response model, where *K* indicates the number of categories.

`params` a matrix of class "brm" or "grm" containing the item parameters used in the simulation. In the case of "grm", the threshold parameters will be ordered so that they will work in other functions.

`theta` a vector of theta used in the simulation. If `theta` is not specified by the user, it will default to a 201-length vector of evenly spaced points between -3 and 3.

Author(s)

Steven W. Nydick <nydic001@umn.edu>

References

Embretson, S. E., & Reise, S. P. (2000). *Item Response Theory for Psychologists*. Mahway, NJ: Lawrence Erlbaum Associates.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph Supplement*, *34*, 100 – 114.

van der Linden, W. J. & Hambleton, R. K. (2010). *Handbook of Modern Item Response Theory*. New York, NY: Springer.

See Also

catIrt

Examples

```
#########################
# Binary Response Model #
#########################
set.seed(888)
# generating an item bank under a binary response model:
b.params <- cbind(a = runif(100, .5, 1.5), b = rnorm(100, 0, 2), c = .2)
# simulating responses using default theta:
b.mod <- simIrt(params = b.params, mod = "brm")

# same type of model without a guessing (c) parameter:
b.params2 <- cbind(a = runif(100, .5, 1.5), b = rnorm(100, 0, 2), c = 0)
b.mod2 <- simIrt(params = b.params2, mod = "brm")

# now generating a different theta:
theta <- rnorm(201)
b.mod3 <- simIrt(theta = theta, params = b.params2, mod = "brm")

# notice all of the responses are 0 or 1:
unique(as.vector(b.mod$resp))

# and the percentages (in general) increase as theta increases:
apply(b.mod$resp, 1, mean)    # theta = seq(-3, 3, by = 0.1)


#########################
# Graded Response Model #
#########################
set.seed(999)
# generating an item bank under a graded response model:
# (as many categories as your heart desires!)
g.params <- cbind(a = runif(10, .5, 1.5), b1 = rnorm(10), b2 = rnorm(10),
                                          b3 = rnorm(10))
# simulating responses using default theta (automatically sorts boundaries):
```

```
g.mod <- simIrt(params = g.params, mod = "grm")

# notice how the old parameters were not sorted:
g.params
# but the new parameters are sorted from column 2 on:
g.mod$params

# don't use these parameters with the binary response model:
try(simIrt(params = g.params, mod = "brm"), silent = TRUE)[1]

# a better parameter set for the graded response model:
g.params2 <- cbind(runif(100, .5, 1.5), b1 = runif(100, -2, -1), b2 = runif(100, -1, 0),
                                        b3 = runif(100, 0, 1), b4 = runif(100, 1, 2))
g.mod2 <- simIrt(params = g.params2, mod = "grm")

# notice all of the responses are positive integers:
unique(as.vector(g.mod$resp))
unique(as.vector(g.mod2$resp))

# and the responses (in general) increase as theta increases:
apply(g.mod2$resp, 1, mean)
```

Index