Package 'kcirt'

February 20, 2015

Type Package

Title k-Cube Thurstonian IRT Models

Version 0.6.0

Date 2013-11-27

Author Dave Zes, Jimmy Lewis, Dana Landis @ Korn/Ferry International

Maintainer Dave Zes <zesdave@gmail.com>

Description Create, Simulate, Fit, Solve k-Cube Thurstonian IRT Models

Depends R (>= 3.0.0), mvtnorm, snowfall, corpcor

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-22 18:46:05

R topics documented:

kcirt-package *k-Cube Thurstonian IRT Models*

Description

Create, Simulate, Fit, Solve k-Cube Thurstonian IRT Models.

Details

Package:	kcirt
Type:	Package
Version:	0.6.0
Date:	2014-04-22
License:	GPL (>= 2)

Use `kcirt.model` to define a k-Cube Thurstonian IRT model. The function `kcirt.sim` generates a random realization. The function `kcirt.fitEE` uses an expectation-expectation volley to approximately locate `mu` and `Lambda` and predict the states, `Eta`. The function `kcirt.fitMSS` makes use of metaheuristic stochastic search to further refine the predictions/estimates.

The system of interest is defined as

y_i* = Δ μ + Δ Λ S η_i + Δ ϵ_i

y = 1, if y* > 0

y = 0, otherwise

$Y = (y_1, y_2, ..., y_N)$

where

y_i is the (column) response vector for observation i.

Δ is the Delta matrix.

μ is the column vector of item means (aka, 'utilities').

Λ is the hyperparameter matrix (aka, 'loadings').

S is the Slot matrix.

η_i is the row vector of latent states (aka, 'constructs', or 'scales') for observation i.

$\epsilon_i \sim N[0, \Sigma_s]$ is a column vector of system shocks for observation i.

Author(s)

Dave Zes, Jimmy Lewis, Dana Landis @ Korn/Ferry International

<zesdave@gmail.com>

References

Brown, A., & Maydeu-Olivares, A. (2012, November 12). How IRT Can Solve Problems of Ipsative Data in Forced-Choice Questionnaires. Psychological Methods. Advance online publication. doi: 10.1037/a0030641

`ikcirt.data2Y` *Convert Rank Data to Y*

Description

Converts raw rank data to the dichotomous response matrix, Y. Probably no need for user to call; called by `kcirt.model` and `kcirt.sim`.

Usage

```
ikcirt.data2Y(mxData, mxDelta)
```

Arguments

`mxData`	Ranked Data. A matrix of positive integers (rankings).
`mxDelta`	System Delta matrix, elements in `{-1,0,1}`.

Value

Response matrix with elements in `{NA,0,1}`.

Author(s)

Dave Zes, Korn/Ferry International

See Also

For the inverse of this function, see `ikcirt.Y2data`.

Examples

```
##### here's an itty-bitty example

constructMap.ls <- list(
c(1,1,2,2),
c(2,2,3,3),
c(1,1,3,3)
)

qTypes <- rep("M", length(constructMap.ls))

model <- kcirt.model(constructMap.ls=constructMap.ls, qTypes=qTypes, mxLambda=NULL)
```

```
#### fake data, two cases (cases run column-wise)
mxData <- t(
rbind(
c( c(1, NA, NA, 4), c(4, NA, NA, 1), c(NA, 4, NA, 1) ),
c( c(1, 4, NA, NA), c(NA, NA, 1, 4), c(NA, 4, 1, NA) )
)
)

ikcirt.data2Y(mxData, mxDelta=model$mxDelta)
```

`ikcirt.df1` *Conservative Degrees of Freedom*

Description

A simple, conservative calculation of a model's degrees of freedom. Each question-respondent counts as 1 df.

Usage

```
ikcirt.df1(model, lambdaConstraint = "self")
```

Arguments

`model`	A kcirt model. A named list of `class` 'kcube.irt.model'. Must possess an object from which number of respondents can be gleaned, e.g., `Y`.
`lambdaConstraint`	A scalar string. Please see `kcirt.fitMSS`.

Value

Scalar integer.

Author(s)

Dave Zes, Korn/Ferry International

Examples

```
## Please see example in kcirt.fitMSS.
```

`ikcirt.fun.mss.eta` *MSS Locate Eta*

Description

Used by `kcirt.fitMSS`; not to be called by user.

Usage

```
ikcirt.fun.mss.eta(jj, iimss, jjmss, rndTrys, mxHatEta, penalty, usetruesigma)
```

Arguments

`jj`	Index for `snowfall` cluster call.
`iimss`	Row index of `mxHatEta` to presently search.
`jjmss`	Column index of `mxHatEta` to presently search.
`rndTrys`	Values to try.
`mxHatEta`	Predicted State matrix.
`penalty`	Scalar string. How to measure quality of fit? Currently either 'logit', 'L2', 'L2c', 'miscat'.
`usetruesigma`	Scalar boolean. Use actual (assumed) system variance?

Value

Fit cost. A scalar real-valued number.

Note

I cannot imagine a situation where the user could profit by calling this function. It is not namespaced for its possible didactic value.

Author(s)

Dave Zes, Korn/Ferry International

See Also

See Also `kcirt.fitMSS`, `ikcirt.fun.mss.mu`, `ikcirt.fun.mss.lambda`.

`ikcirt.fun.mss.lambda` *MSS Locate Lambda*

Description

Used by `kcirt.fitMSS`; not to be called by user.

Usage

```
ikcirt.fun.mss.lambda(jj, iimss, jjmss, rndTrys, mxHatLambda, penalty, usetruesigma)
```

Arguments

`jj`	Index for `snowfall` cluster call.
`iimss`	Row index of `mxHatLambda` to presently search.
`jjmss`	Column index of `mxHatLambda` to presently search.
`rndTrys`	Values to try.
`mxHatLambda`	Estimated Hyperparameter matrix.
`penalty`	Scalar string. How to measure quality of fit? Currently either 'logit', 'L2', 'L2c', 'miscat'.
`usetruesigma`	Scalar boolean. Use actual (assumed) system variance?

Value

Fit cost. A scalar real-valued number.

Note

I cannot imagine a situation where the user could profit by calling this function. It is not namespaced for its possible didactic value.

Author(s)

Dave Zes, Korn/Ferry International

See Also

See Also `kcirt.fitMSS`, `ikcirt.fun.mss.mu`, `ikcirt.fun.mss.eta`.

`ikcirt.fun.mss.mu` *MSS Locate mu*

Description

Used by `kcirt.fitMSS`; not to be called by user.

Usage

```
ikcirt.fun.mss.mu(jj, iimss, rndTrys, hatMu, useSysCov, penalty)
```

Arguments

`jj`	Index for `snowfall` cluster call.
`iimss`	Index of `hatMu` to presently search.
`rndTrys`	Values to try.
`hatMu`	Estimated item means.
`useSysCov`	Either the true or estimated System Variance matrix.
`penalty`	Scalar string. How to measure quality of fit? Currently either 'logit', 'L2', 'L2c', 'miscat'.

Value

Fit cost. A scalar real-valued number.

Note

I cannot imagine a situation where the user could profit by calling this function. It is not namespaced for its possible didactic value.

Author(s)

Dave Zes, Korn/Ferry International

See Also

See Also `kcirt.fitMSS`, `ikcirt.fun.mss.lambda`, `ikcirt.fun.mss.eta`.

ikcirt.rndData1 *Random Data*

Description

Generate a random realization of a minimally defined k-Cube Thurstonian IRT Model. Probably no need for user to call; called by kcirt.sim.

Usage

```
ikcirt.rndData1(N, qTypes, mxDelta, ns)
```

Arguments

N	Number of observational units.
qTypes	Is a question to be fully ranked or most/least format. A character vector whose length is the number of blocks. Each element in 'R', 'M'.
mxDelta	Delta matrix.
ns	Response dimensions. A vector of length equal to the number of questions, each element giving the number of items in the question.

Value

Ranked Data matrix.

Author(s)

Dave Zes, Korn/Ferry International

Examples

```
constructMap.ls <- list(
c(1,2),
c(2,3),
c(1,3)
)

qTypes <- rep("R", length(constructMap.ls))

mod <- kcirt.model(constructMap.ls=constructMap.ls, qTypes=qTypes, mxLambda=NULL)

N <- 50
set.seed(99999)
mod <- kcirt.sim(model=mod, N=N)

ikcirt.rndData1(N=N, qTypes=qTypes, mxDelta=mod$mxDelta, ns=mod$ns)
```

ikcirt.Ustar2data *Convert* U* *to Data*

Description

Convert the non-differenced latent responses to Data. Probably no need for user to call; called by kcirt.sim.

Usage

```
ikcirt.Ustar2data(Ustar, qTypes, mxDelta, ns)
```

Arguments

Ustar	Non-differenced latent responses matrix.
qTypes	Is a question to be fully ranked or most/least format. A character vector whose length is the number of blocks. Each element in {'R', 'M'}.
mxDelta	System Delta matrix, elements in {-1,0,1}.
ns	Response dimensions. A vector of length equal to the number of questions, each element giving the number of items in the question.

Value

Ranked Data matrix.

Author(s)

Dave Zes, Korn/Ferry International

ikcirt.Y2data *Convert Y to Rank Data*

Description

Converts a dichotomous response matrix, Y, to rank data. Probably no need for user to call; called by kcirt.model.

Usage

```
ikcirt.Y2data(Y, mxDelta, ns)
```

Arguments

Y	Response matrix.
mxDelta	System Delta matrix. A q x m matrix with elements in {-1,0,1}.
ns	Response dimensions. A vector of length equal to the number of questions, each element giving the number of items in the question.

Value

Ranked Data matrix.

Author(s)

Dave Zes, Korn/Ferry International

See Also

For the inverse of this function, see ikcirt.data2Y.

Examples

```
##### here's an itty-bitty example

constructMap.ls <- list(
c(1,1,2,2),
c(2,2,3,3),
c(1,1,3,3)
)

qTypes <- rep("M", length(constructMap.ls))

model <- kcirt.model(constructMap.ls=constructMap.ls, qTypes=qTypes, mxLambda=NULL)

#### fake data, two cases (cases run column-wise)
mxData <- t(
rbind(
c( c(1, NA, NA, 4), c(4, NA, NA, 1), c(NA, 4, NA, 1) ),
c( c(1, 4, NA, NA), c(NA, NA, 1, 4), c(NA, 4, 1, NA) )
)
)

#### convert data to Y
Y <- ikcirt.data2Y(mxData=mxData, mxDelta=model$mxDelta)

#### convert Y back to data
ikcirt.Y2data(Y=Y, mxDelta=model$mxDelta, ns=model$ns)
```

---

kcirt.fitEE *Least Squares k-Cube Thurstonian IRT Fitting*

Description

k-Cube Thurstonian IRT Fitting using a least-squares expectation-expectation algorithm.

Usage

```
kcirt.fitEE(model, mxHatLambda, maxIter = 40, lambda.ridge = 0.3, Seta.ridge=0.01)
```

Arguments

model	A kcirt model. A named list of class 'kcube.irt.model'.
mxHatLambda	An initial guess for the Hyperparameters.
maxIter	Maximum number of iterations.
lambda.ridge	Non-negative real-valued scalar. Amount of Ridge shrinkage on hatLambda crossproduct for LS stages.
Seta.ridge	Non-negative real-valued scalar. Amount of Ridge shrinkage on SEta crossproduct for LS stages.

Details

This function can be thought of as an expectation-expectation procedure. The starting Hyperparameters, mxHatLambda, are used to predict mxEta (this prediction is commonly called mxHatEta is this package), and so on, back and forth. The procedure stops when either the L2 cost first bottoms out, or maxIter is met.

In many cases, this function alone produces excellent-performing estimates/predictions. The user may pass the returned model to kcirt.fitMSS for further refinement.

Value

A kcirt model. A named list of class 'kcube.irt.model'.

Author(s)

Dave Zes, Korn/Ferry International

See Also

See Also kcirt.fitMSS.

Examples

```
constructMap.ls <- list(
c(1,1,2,2),
c(1,1,3,3),
c(2,2,3,3),
c(1,1,2,2),
c(1,1,3,3),
c(2,2,3,3)
)

qTypes <- rep("R", length(constructMap.ls))

mod <- kcirt.model(constructMap.ls=constructMap.ls, qTypes=qTypes, mxLambda=NULL)
```

```
N <- 300
set.seed(99999)
mod <- kcirt.sim(model=mod, N=N)

ikcirt.df1(mod, "self")

mxHatLambda <- mod$mxLambda - matrix( rnorm( sum(mod$ns)^2, 0, 0.3 ), sum(mod$ns), sum(mod$ns) )

mod2 <- kcirt.fitEE(model=mod, mxHatLambda=mxHatLambda, maxIter=40)
```

`kcirt.fitMSS` *Metaheuristic k-Cube Thurstonian IRT Fitting*

Description

Use metaheuristic stochastic search to locate k-Cube Thurstonian IRT hyperparameters and states.

Usage

```
kcirt.fitMSS(model, lambdaConstraint = "self", kcpus = 2, penalty = "logit",
usetruesigma = TRUE, mss.sd = 0.2, nsearch = 19, l2zvarpow = 0,
xmu.shrink=0, xlambda.shrink=0, xeta.shrink=0.4)
```

Arguments

`model`	A kcirt model; a named list of `class` "kcube.irt.model".
`lambdaConstraint`	Scalar string specifying how to constrain Lambda during fitting. See Details.
`kcpus`	Scalar positive integer telling `snowfall` how many threads to initialize. Set to 1 for no parallel processing.
`penalty`	Scalar string. How to measure quality of fit? Currently either 'logit', 'L2', 'L2c', 'miscat'.
`usetruesigma`	Scalar boolean. Use actual (assumed) system variance?
`mss.sd`	Scalar positive number or vector of length 3. Size of search function standard deviation for, in order, `mu`, `Lambda`, and `Eta` searches.
`nsearch`	Scalar positive integer. How many candidate values to draw from under the MSS function. Typically 14-20.
`l2zvarpow`	Scalar positive integer. Only used when `penalty` is 'L2c'. Power by which to raise `var(2*Y-1)`; the amount of 'smoothing' to impose on `2*Y-1`. Note that when zero, setting `penalty` to 'L2c' produces identical results to 'L2' – though slightly slower computationally.
`xmu.shrink`	Scalar non-negative real-valued. Only used when `penalty` is 'logit'. How much to shrink utility estimates towards zero.
`xlambda.shrink`	Scalar non-negative real-valued. Only used when `penalty` is 'logit'. How much to shrink loading estimates towards zero.
`xeta.shrink`	Scalar non-negative real-valued. Only used when `penalty` is 'logit'. How much to shrink state predictions towards zero.

Details

Note: As of kcirt version >= 0.6, the argument `logitshrinkcoef` has been removed. Use `xeta.shrink` instead. **lambdaConstraint** defines the model k-cube, i.e., the item crosstalk space. Currently accepts `self`, only diagonal elements of `Lambda`; `withinx`, additionally includes within block items pointing to **ex**ogenous constructs; `withini`, additionally includes items within block pointing to same construct. `betweenx`, diagonal elements and all other items that point to different constructs; `betweeni`, additionally includes items that point to same construct. `priorx` is similar to `betweenx` except that only previous items crosstalk with item, i.e., `Lambda` is lower-block-diagonal; `priori` additionally permits crosstalk between prior items pointing to same construct.

penalty defines the objective function to be minimized. `logit` minimizes the logistic deviance. `L2` minimizes sum of squares between `2*Y-1` and `2*Yhat-1`. `L2c` is similar to `L2`, except that it pre-projects (up to a constant) `2*Y-1` and `2*Yhat-1` using `var(2*Y-1)`. Note that `NA` values in `Y` are replaced with 0.5. Finally, `miscat`, minimizes the misclassification rate using a confusion table between the observations, `Y`, and the predicted states (scales), `hatEta`, split at zero.

Excepting very adventurous explorations with this model, the defaults for **lambdaConstraint** and **penalty** will likely best serve the user.

Value

A kcirt model. A named list of `class` 'kcube.irt.model'.

Author(s)

Dave Zes, Korn/Ferry International

See Also

See Also `kcirt.fitEE`.

Examples

```
constructMap.ls <- list(
c(1,1,2,2),
c(1,1,3,3),
c(2,2,3,3),
c(1,1,2,2),
c(1,1,3,3),
c(2,2,3,3),
c(1,2),
c(2,3),
c(1,3)
)


qTypes <- rep("R", length(constructMap.ls))

mod <- kcirt.model(constructMap.ls=constructMap.ls, qTypes=qTypes, mxLambda=NULL)

N <- 200
set.seed(99999)
```

```
mod <- kcirt.sim(model=mod, N=N)


ikcirt.df1(mod, "self")

####### create initial guess for hyperparameters (aka loadings)
mod$mxHatLambda <- mod$mxLambda - matrix( rnorm( sum(mod$ns)^2, 0, 0.3 ), sum(mod$ns), sum(mod$ns) )

####### need to assign hat states and utilities
mod$mxHatEta <- matrix(0, N, sum(mod$nuc))
mod$hatMu <- rep(0, sum(mod$ns))

## Not run:
###### run MSS fit -- performance is R^2 btwn true and pred states (aka scales)
mod <- kcirt.fitMSS(model=mod, lambdaConstraint="self", kcpus=2, penalty="L2",
usetruesigma=TRUE, mss.sd=1)
mod$performance

###### run again ...
mod <- kcirt.fitMSS(model=mod, lambdaConstraint="self", kcpus=2, penalty="L2",
usetruesigma=TRUE, mss.sd=1)
mod$performance

###### run a few more times ... ...

## End(Not run)
```

`kcirt.model` *Create k-Cube Thurstonian IRT Model*

Description

Create a k-Cube Thurstonian IRT Model skeleton.

Usage

```
kcirt.model(constructMap.ls, qTypes, data = NULL, Y = NULL, mu = 0, mxLambda = NULL,
covEta = 1, covShocks = 1, deltaType=1)
```

Arguments

`constructMap.ls`	How the constructs map to the items. A list of vectors, each vector representing a block (question in the instrument); each element of a vector giving the construct index.
`qTypes`	Is a question to be fully ranked or most/least format. A character vector whose length is the number of blocks. Each element in `{"R", "M"}`. Note that only accepts 'R' when corresponding block has 3 or 2 items.

data	Data matrix. The number of columns is the number of observational units, the number of rows is the total number of items.
Y	Response cohort matrix, each element in {NA,0,1}.
mu	True item means. A real-valued vector whose length is the total number of items.
mxLambda	True hyperparameters. A real-valued square matrix, or a vector. If vector, mxLambda is assumed to be diagonal (no item crosstalk) – vector is recycled
covEta	True covariance of states. A square, symmetric real-valued matrix.
covShocks	True covariance of shocks. A square, symmetric real-valued matrix.
deltaType	Tell function in what pattern items are compared. Scalar. Directly affects structure of the Delta matrix. Currently one of two integer values. 1 is default. 2 is that implied by Brown and Maydeau-Oliveras.

Value

A kcirt model. A named list of class "kcube.irt.model".

Author(s)

Dave Zes, Korn/Ferry International

Examples

```
constructMap.ls <- list(
c(1,2),
c(2,3),
c(1,3)
)

qTypes <- rep("R", length(constructMap.ls))

mod <- kcirt.model(constructMap.ls=constructMap.ls, qTypes=qTypes, mxLambda=NULL)

## view contents
mod
```

---

kcirt.sim *Simulate a k-Cube Thurstonian IRT Model*

Description

Given model parameters, create a random realization of a k-Cube Thurstonian IRT Model skeleton.

Usage

```
kcirt.sim(model, N, type = "Eta")
```

Arguments

model	A kcirt model. A named list of class 'kcube.irt.model'.
N	Number of observations.
type	Currently ignored.

Value

A kcirt model. A named list of class 'kcube.irt.model'.

Author(s)

Dave Zes, Korn/Ferry International

Examples

```
## Please see example in, e.g., kcirt.fitMSS.
```

---

kcirt.ystarinfo *Calculate State Information*

Description

Calculate System Information imparted to States through $Y*$.

Usage

```
kcirt.ystarinfo(model)
```

Arguments

model	A kcirt model. A named list of class 'kcube.irt.model'.

Value

A numeric, square, symmetric matrix, whose number of rows equals the number of model states (scales).

Author(s)

Dave Zes, Korn/Ferry International

Examples

```
constructMap.ls <- list(
c(1,1,2,2),
c(1,1,3,3),
c(2,2,3,3),
c(1,1,2,2),
c(1,1,3,3),
c(2,2,3,3),
c(1,2),
c(2,3),
c(1,3)
)


qTypes <- rep("R", length(constructMap.ls))

mod <- kcirt.model(constructMap.ls=constructMap.ls, qTypes=qTypes, mxLambda=NULL)

mxInfo <- kcirt.ystarinfo(mod)

mxInfo

mxErrorVar <- solve(mxInfo)

mxErrorVar
```

Index