Package ‘cds’

January 5, 2016

Type Package

Title Constrained Dual Scaling for Detecting Response Styles

Version 1.0.3

Date 2016-01-05

Author Pieter Schoonees [aut, cre]

Maintainer Pieter Schoonees <schoonees@gmail.com>

Description This is an implementation of constrained dual scaling for detecting response styles in categorical data, including utility functions. The procedure involves adding additional columns to the data matrix representing the boundaries between the rating categories. The resulting matrix is then doubled and analyzed by dual scaling. One-dimensional solutions are sought which provide optimal scores for the rating categories. These optimal scores are constrained to follow monotone quadratic splines. Clusters are introduced within which the response styles can vary. The type of response style present in a cluster can be diagnosed from the optimal scores for said cluster, and this can be used to construct an imputed version of the data set which adjusts for response styles.

Depends R(>= 3.2.3), parallel

Imports MASS, limSolve, clue, colorspace, copula, graphics, methods, stats

LazyLoad yes

LazyData yes

ByteCompile yes

License GPL (>= 2)

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-01-05 14:29:39
R topics documented:

cds-package ..................................................... 2  
addbounds ...................................................... 3  
approxloads ..................................................... 4  
calc.wt.bubbles .............................................. 4  
cds .............................................................. 5  
cds.sim ......................................................... 7  
clean.scales ................................................... 9  
cl_class_ids.cds ............................................. 9  
create.ind ..................................................... 10  
create.rs ...................................................... 10  
createcdssdata ............................................... 11  
datsim .......................................................... 11  
G.start ......................................................... 12  
gen.cop ......................................................... 13  
genPCA .......................................................... 14  
group.ALS ..................................................... 15  
indmat ........................................................... 15  
ispline ........................................................... 16  
L.fun ............................................................. 16  
L.fun.G.upd .................................................... 17  
orthprocr ...................................................... 17  
plot.cds .......................................................... 18  
plot.cdslist ................................................... 19  
print.cds ........................................................ 19  
print.cdsdata ............................................... 20  
rcormat .......................................................... 20  
rcovmat .......................................................... 21  
sensory ........................................................... 21  
sensory.aux .................................................... 22  
simpca ............................................................ 22  
trQnorm .......................................................... 23  
trRnorm .......................................................... 24  
updateG ........................................................... 24

Index ............................................................ 26

cds-package  
Constrained Dual Scaling for Successive Categories

Description

Fit constrained dual scaling for detecting response styles.

Author(s)

Pieter C. Schoonees
addbounds

References

Departmental report available

Description

Augment with Boundaries Between Rating Scale Categories and Rank

Description

Adds $q - 1$ boundaries between the $q$ ratings to the columns of matrix $x$, and convert the rows to rankings, starting with 0 for the lowest ranking. Ties are handled by averaging the total rank for all tied observations.

Usage

addbounds(x, q = max(x), ties = "average")

Arguments

x matrix (or data frame) of $n$ rows and $m$ columns, or an object that can be coerced to a matrix via as.matrix.
q scalar; the number of rating scale categories. Defaults to the maximum entry in $x$.
ties character; handling of ties in rank

Details

Any $x$ which is not a matrix or data frame will cause an error.

Value

A matrix of size $n$ by $m + q - 1$

Author(s)

Pieter C. Schoonees

Examples

set.seed(1234)
mat <- matrix(sample(1:9, 12, replace = TRUE), nrow = 4, ncol = 3)
addbounds(mat, q = 9)
approxloads  

*Low Rank Approximation* $LL'$ of a Square Symmetric Matrix $R$

**Description**

Uses the eigendecomposition of a square, symmetric matrix $R$ to obtain the loadings matrix $L$ such that $R$ is approximated by $LL'$, with $L$ restricted to have $r$ columns. Hence $LL'$ is a rank $r$ approximation of $R$. The eigendecomposition of $R$ is used to obtain $L$ from the first $r$ eigenvectors and eigenvalues. In case `procr.target` is not `NULL`, $L$ is further rotated through orthogonal Procrustes analysis to match as closely as possible the matrix `procr.target` through `orthprocr`.

**Usage**

```r
approxloads(R, r = 3, procr.target = NULL, refl.target = NULL)
```

**Arguments**

- **R**: Square, symmetric matrix $R$ to be approximated
- **r**: The required rank of the approximation
- **procr.target**: Optional; the target matrix for $L$ in the orthogonal Procrustes analysis
- **refl.target**: Optional; the matrix to check against for possible reflections of the loading vectors.

**Examples**

```r
R <- rcormat(10, r = 3)
all.equal(R$L, approxloads(R$R, r = 3, procr.target = R$L))
```

---

calc.wt.bubbles  

*Calculate the Weights for Bubble Plots*

**Description**

Calculate weights for the bubbles in the plot method of `cds` objects. The relative frequencies within a set of groups are used to calculate the size of the bubble so that the area of the bubble is proportional to the relative frequency of the rating category within that group.

**Usage**

```r
calc.wt.bubbles(dat, grp, q, fact = 0.12)
```
Arguments

dat A data set from which to derive the relative frequencies
grp A vector giving the group memberships.
q An integer such that the rating scale is 1:q.
fact A shrinkage factor.

Author(s)

Pieter Schoonees

Description

Uses an alternating nonnegative least squares algorithm combined with a k-means-type algorithm to optimize the constrained group dual scaling criterion outlined in the reference. Parallel computations for random starts of the grouping matrix is supported via package parallel.

Usage

cds(x, K = 4, q = NULL, eps.ALS = 0.001, eps.G = 1e-07, nr.starts.G = 20, nr.starts.a = 5, maxit.ALS = 20, maxit = 50, Gstarts = NULL, astarts = NULL, parallel = FALSE, random.G = FALSE, times.a.multistart = 1, info.level = 1, mc.preschedule = TRUE, seed = NULL, LB = FALSE, reorder.grps = TRUE, rescale.a = TRUE, tol = sqrt(.Machine$doub...)

Arguments

x an object of class "dsdata" (see cds.sim), or a matrix (or object coercible to a matrix) containing the data for n individuals on m objects. The data does not yet contain any additional columns for the rating scale.
K The number of response style groups to look for. If a vector of length greater than one is given, the algorithm is run for each element and a list of class cdslist is returned.
q The maximum rating (the scale is assumed to be 1:q).
eps.ALS Numerical convergence criterion for the alternating least squares part of the algorithm (updates for row and column scores).
eps.G Numerical convergence criterion for the k-means part of the algorithm.
rn.starts.G Number of random starts for the grouping matrix.
rn.starts.a Number of random starts for the row scores.
maxit.ALS  Maximum number of iterations for the ALS part of the algorithm. A warning is given if this maximum is reached. Often it is not a concern if this maximum is reached.

maxit  Maximum number of iterations for the k-means part of the algorithm.

Gstarts  Facility to supply a list of explicit starting values for the grouping matrix G. Each start consists of a two element list: i giving an integer number the start, and G giving the starting configuration as an indicator matrix.

astarts  Supply explicit starts for the a vectors, as a list.

parallel  logical. Should parallelization over starts for the grouping matrix be used?

random.G  logical. Should the k-means part consider the individuals in a random order?

times.a.multistart  The number of times that random starts for the row scores are used. If == 1, then random starts are only used once for each start of the grouping matrix.

info.level  Verbosity of the output. Options are 1, 2, 3 and 4.

mc.preschedule  Argument to mclapply under Unix.

seed  Random seed for random number generators. Only partially implemented.

LB  logical. Load-balancing used in parallelization or not? Windows only.

reorder.grps  logical. Use the Hungarian algorithm to reorder group names so that the trace of the confusion matrix is maximized.

rescale.a  logical. Rescale row score to length sqrt(2n) if TRUE (after the algorithm has converged).

tol  tolerance tol passed to lsei of the limSolve package. Defaults to sqrt(.Machine$double.eps)

update.G  Logical indicating whether or not to update the G matrix from its starting configuration. Useful when clustering is known apriori or not desired.

Details

See the reference for more details.

Value

Object of class ds with elements:

G  Grouping indicator matrix.

K  Number of groups K.

opt.crit  Optimum value of the criterion.

a  The 2n-vector of row scores.

bstar  The m-vector of object scores.

bkmat  The matrix of group-specific boundary scores for the ratings.

alphamat  The estimated spline coefficients for each group.

iter  The number of iterations used for the optimal random start wrt the grouping matrix.
time.G.start  The number of seconds it took for the algorithm to converge for this optimal random start.
grp  The grouping of the individuals as obtained by the algorithm.
kloss  Loss value from G update (not equivalent to that of ALS updates).
hitrate, confusion  Confusion and hitrates of original data object contained a grouping vector.
loss.G  Optimality criterion values for the random starts of G.
q  The number of ratings in the Likert scale 1:q
time.total  Total time taken for the algorithm over all random starts
call  The function call.
data  The input data object.

Author(s)

Pieter C. Schoonees

References


Examples

```r
set.seed(1234)
dat <- cds.sim()
out <- cds(dat)
```

Description

Simulate response data for a group of response styles.

Usage

```r
cds.sim(nr.indv = c(100, 100, 100), m = 25, scales = 1:7,
err.coeff = 0.1, alphamat = rbind(c(4, 4, 1), c(1, 4, 4), c(1, 2, 1)),
true.mu = NULL, random = TRUE, same.mu = TRUE, use.copula = FALSE,
reverse.thresh = 1)
```
Arguments

nr.indv A vector giving the number of respondents in each group.
m The number of objects.
scales The rating scale used, 1:q.
err.coeff The standard error used in the underlying normal noise.
alphamat The matrix of spline parameters defining the response styles, with each row containing a response style. No intercepts should be included.
true.mu Optional; a matrix or vector giving the true underlying preferences for the objects.
random Logical indicating whether to apply the response styles in random order
same.mu Logical indicating whether a universal value for mu should be assumed.
use.copula Logical indicating whether to use a correlated dependence structure through a copula.
reverse.thresh A numeric value giving the proportion of observations for which the dependence structure should be reversed. Only applicable when copula is TRUE.

Value

An object of class \texttt{cdsdata}, inheriting from class \texttt{icdsdata}, which is a list with the following slots:

\begin{itemize}
\item \texttt{prers} The pre-response style simulated data
\item \texttt{postrs} The data after adding the response styles
\item \texttt{postbl} The same as \texttt{postrs} in this case
\item \texttt{Fr.cent.rs} The centred Fr matrix for \texttt{postrs}
\item \texttt{Fr.rs} The Fr matrix for \texttt{postrs}
\item \texttt{Fr.cent.bl} The same as \texttt{Fr.cent.rs}, for compatibility with \texttt{icds}
\item \texttt{Fr.bl} The same as \texttt{Fr.rs}, for compatibility with \texttt{icds}
\item \texttt{mu} Matrix of the true underlying preference structure for the objects
\item \texttt{block} Numeric vector identifying the different blocks for incompleteness, in this case a vector of ones
\item \texttt{grp.rs} The response style grouping vector
\item \texttt{alphamat} Matrix of spline parameters for the response styles
\item \texttt{scales} The rating scale 1:q used
\item \texttt{m} Number of objects
\item \texttt{munique} The number of objects seen within each block - equal to zero in this case
\item \texttt{m0} The number of objects seen by all subjects - equal to m in this case
\item \texttt{true.tau} Actual tau used in the simulation with copulae
\item \texttt{call} The function call
\end{itemize}

See Also

\texttt{createcdsdata}
**clean.scales**  
*Impute Optimal Scores for Rating Categories*

**Description**

Replace original ratings with optimal scores based on cds output.

**Usage**

```r
clean.scales(object, data, K, col.subset = NULL, ...)
```

```r
## S3 method for class 'cds'
clean.scales(object, data, K, col.subset = NULL, ...)
```

```r
## S3 method for class 'cdslist'
clean.scales(object, data, K, col.subset = NULL, ...)
```

**Arguments**

- `object`  
  An object of class cds

- `data`  
  An object of class cdsdata to be cleaned, or the original data.

- `K`  
  The number of classes in the solution that must be kept.

- `col.subset`  
  An optional subset

- `...`  
  Additional arguments.

---

**cl_class_ids.cds**  
*S3 Methods for Integration into clue Framework*

**Description**

These methods integrate the class cds into the framework set out in package clue. Use can therefore by made of cl_agreement to calculate concordance measures between different solutions.

**Usage**

```r
## S3 method for class 'cds'
cl_class_ids(x)
```

```r
## S3 method for class 'cds'
is.cl_partition(x)
```

```r
## S3 method for class 'cds'
is.cl_hard_partition(x)
```
## S3 method for class 'cdsdata'
class_ids(x)

## S3 method for class 'cdsdata'
is.cl_partition(x)

## S3 method for class 'cdsdata'
is.cl_hard_partition(x)

### Arguments

- **x**: An object of class `cds`

### create.ind

**Create Indicator Matrix**

### Description

Create an indicator matrix.

### Usage

`create.ind(grp)`

### Arguments

- **grp**: A grouping vector.

### create.rs

**Create a response style**

### Description

Creates a response style by cutting up a quadratic monotone spline.

### Usage

`create.rs(alpha = matrix(c(1, 2, 1), nrow = 1), nr.scale = 7, tvec = c(0, 0.5, 1), xvec = 0:nr.scale/nr.scale, scale = TRUE)`

### Arguments

- **alpha**: vector of spline coefficients
- **nr.scale**: number of rating categories; numeric
- **tvec**: knots for spline functions
- **xvec**: evaluation points for basis functions
- **scale**: logical; scale or not
createcdsdata  

Create a cdsdata Object

Description
Create a cdsdata object from a data frame or matrix.

Usage
createcdsdata(x, q = NULL)

Arguments
- x: A data frame or matrix containing the data.
- q: Optional; the maximum rating category, so that the rating scale used for all items are 1:q.

datsim  

Simulate Data for a Single Response Style

Description
Simulate data containing a single response style.

Usage
datsim(nr.indv = 100, m = 5, scales = 1:7, err.coef = 0.1,
        resp.style = c(-Inf, 1/7, 2/7, 3/7, 4/7, 5/7, 6/7, Inf),
        true.mu = NULL,
        a = 0, b = 1, plot.graph = FALSE, use.copula = FALSE,
        reverse.thresh = 1, ...)

Arguments
- nr.indv: Integer giving the number of individuals required in the sample.
- m: The number of items.
- scales: The rating scale used for all items.
- err.coef: The standard error used in simulating the truncated normal distribution.
- resp.style: A set of cut points across the interval [0, 1] defining the response style transformation.
- true.mu: Optional vector of length m giving the true preferences for the items.
a Lower boundary of the truncation interval for the simulated true preferences.
b Upper boundary for the truncation interval for the simulated true preferences.
plot.graph Logical indicating whether to visualize the response style in a plot.
use.copula Logical indicating whether to simulate dependent items using a copula.
reverse.thresh A proportion giving the proportion of item preferences which should be reversed to induce a negative association.
... Additional arguments passed to plot.

Author(s)

Pieter C. Schoonees

References


---

**G.start**

*Constrained Dual Scaling for a Single Random G Start*

**Description**

Run algorithm for a single G matrix.

**Usage**

```r
G.start(X, nr.starts.a, astarts, maxit, n, m, q, Fr.cent, maxit.ALS, Mmat, eps.G, info.level, times.a multistart, eps.ALS, const, K, random.G, tol, update.G)
```

**Arguments**

- `X` List of two elements, namely i giving the number of the start and G given the starting configuration
- `nr.starts.a` The number or random starts for a to use in the ALS.
- `astarts` Explicit starts for a, if applicable.
- `maxit` The maximum number of iterations with respect to G.
- `n` The number of respondents.
- `m` The number of items.
- `q` The maximum rating category such that the rating scale is 1:q.
- `Fr.cent` The centred Fr matrix.
- `maxit.ALS` The maximum number of ALS iterations.
Mmat: The basis matrix for the quadratic monotone splines.
eps.G: The absolute error tolerance for the G updates.
info.level: Integer controlling the amount of information printed.
times.a.multistart: The number of times random starts for a is used.
eps.ALS: The absolute error tolerance for the ALS.
const: The constant part of the loss function.
K: The number of groups.
random.G: The random argument passed to updateG.
tol: The tolerance tol passed to lsei of the limSolve package.
update.G: Logical indicating whether or not to update the starting configuration G in X

---

gen.cop

Generate a Copula

Description

Generate correlated data multivariate categorical data via a copula.

Usage

gen.cop(n, tauvec = c(0.2, 0.35), nr.cols = c(10, 10), 
true.mu = runif(sum(nr.cols)), err.coeff = 0.1, random = FALSE, 
reverse = TRUE, reverse.thresh = 0.75)

Arguments

n: Integer; the number of samples to draw.
tauvec: A vector of association parameters for each of the Clayton copulae (see copClayton),
of the same length as nr.cols.
nr.cols: A vector giving the number of columns to draw from each of the copulae.
true.mu: A vector giving the mean for each of the columns in the data.
err.coeff: The standard errors for underlying normal distribution.
random: Logical indicating whether or not the samples should be presented in random order.
reverse: Logical indicating whether some of the simulated variables should be reversed to have negative association or not.
reverse.thresh: The proportion of columns to reverse.
**Description**

Generate a response style data set from a specific correlation matrix, clean the data with constrained dual scaling and report the original, cleaned and contaminated correlation matrices in a list.

**Usage**

```r
genPCA(nr.indv = rep(100, 5), m = 10, q = 7, r = 3, err.coeff = 0.1,
    alphamat = rbind(c(0.5, 2, 4), c(10, 2, 10), c(1, 2, 1), c(4, 2, 0.5),
    c(0.1, 2, 0.1))[1:length(nr.indv), ], randomize = TRUE, ...)
```

**Arguments**

- `nr.indv` Vector; number of individuals in each response style group. It is passed to `simpca`.
- `m` scalar; Number of items.
- `q` scalar; Number of rating categories, such that the rating scale is 1:q.
- `r` scalar; Rank of simulated correlation matrices.
- `err.coeff` scalar; Standard deviation used in simulations that is passed on to `simpca`.
- `alphamat` matrix; Contains the spline parameters for the different response styles that is passed to `simpca`.
- `randomize` logical; See `simpca`.
- `...` Arguments passed to `cds`.

**Value**

A list with components:

- `Rsims` Correlation matrix from which the sample was generated
- `Rclean` Correlation matrix for the cleaned data
- `Rcont` Correlation matrix for the contaminated data

**Author(s)**

Pieter C. Schoonees
Alternating Least Squares with Groups for Constrained Dual Scaling

Description

Alternating least-squares for estimating row and column scores in constrained dual scaling, where different groups are allowed for.

Usage

group.ALS(a, m, q, G, Fr.cent, eps = 0.1, maxit = 50, Mmat, info.level = 2, const, K, n, tol)

Arguments

- **a**: A 2n-vector of row scores.
- **m**: Integer; the number of items.
- **q**: Integer; the rating scale from 1:q.
- **G**: An indicator matrix of size n by K.
- **Fr.cent**: The centred F_r matrix.
- **eps**: The numerical tolerance level for the loss.
- **maxit**: Integer; the maximum number of iterations allowed.
- **Mmat**: Matrix of spline basis functions.
- **info.level**: Integer controlling the amount of information printed.
- **const**: The constant part of the loss function.
- **K**: The number of latent classes.
- **n**: The number of samples.
- **tol**: tolerance tol passed to lsei of the limSolve package

Create an Indicator Matrix

Description

Creates an indicator matrix from a grouping vector.

Usage

indmat(grp.vec, K = length(unique(grp.vec)))

Arguments

- **grp.vec**: Numeric vector giving the group membership.
- **K**: Scalar indicating the number of groups. Defaults to the number of unique elements in grp.vec.
ispline

Quadratic monotone spline basis function for given knots.

Description
Calculate basis functions for monotone quadratic splines.

Usage
ispline(xvec, tvec = c(0, 0.5, 1), intercept = TRUE)

Arguments
- xvec: Vector at which to evaluate the basis functions.
- tvec: Vector of spline knots: lower endpoint, interior knot, upper endpoint.
- intercept: Logical; should an intercept be included or not?

Lfun

Calculate Constrained Dual Scaling Loss

Description
Calculate the loss function for constrained dual scaling.

Usage
Lfun(a.cur, bkmat, G, Fr.cent, n, m, q, const, K)

Arguments
- a.cur: The current value for a.
- bkmat: Current value of bkmat.
- G: Current value G.
- Fr.cent: Current value of the centred Fr.
- n: Number of respondents.
- m: Number of items.
- q: Number for rating scale categories so that the rating scale is 1:q.
- const: Constant part of the loss function
- K: Number of response style groups.
Calculate Loss for G Update

Description

Loss function used for updating G. This is not equivalent to the original loss function, as only a part of the total loss depends on G.

Usage

Lfun.G.upd(G, a.cur, bwts2, Fr.bk, n, m, q, K)

Arguments

- G: The current value for G.
- a.cur: The current value for a.
- bwts2: The current value of the squared b weights.
- Fr.bk: Current product between Fr.cent and bk.
- n: Number of respondents.
- m: Number of items.
- q: Number for rating scale categories so that the rating scale is 1:q.
- K: Number of response style groups.

Orthogonal Procrustes Analysis

Description

Simple function to rotate matrix X so that it matches the target matrix Z as closely as possible, by minimizing \|Z - XQ\| where Z and X are of the same size and Q is an orthogonal matrix. The algorithm is based on the singular value decomposition (SVD) (see e.g. the reference).

Usage

orthprocr(Z, X)

Arguments

- Z: The target matrix
- X: The matrix to be rotated, which must be of the same size as Z.
Value

A list with the following 2 elements:

- **Q**  The rotation matrix
- **XQ** The matrix X after rotation

References


---

**plot.cds**  *Plot cds Objects*

Description

Plot method for cds objects

Usage

```r
## S3 method for class 'cds'
plot(x, which = 1L:3L, type = "l", lty = 1, lwd = 2,
     show.legend = TRUE, col = colorspace::rainbow_hcl(nr), bty.legend = "n",
     intercept = ncol(x$alphamat) == 4, scale = FALSE, add = FALSE,
     exp.factor = 1.2, bubble.factor = 0.12, cont.factor = 0.01, pch = 15,
     ...)
```

Arguments

- **x** An object of class cds.
- **which** A numeric vector: a subset of 1:3 specifying the plots to produce.
- **type** Passed to `matplot`.
- **lty** Passed to `matplot`.
- **lwd** Passed to `matplot`.
- **show.legend** Logical; should a legend be added to the plot or not.
- **col** Passed to `matplot`.
- **bty.legend** Passed to `legend`.
- **intercept** Logical indicating whether to plot the intercept.
- **scale** Logical indicating whether an intercept should be included or not.
- **add** Logical; add to plot or not?
- **exp.factor** Factor for expanding the x- and y-limits.
- **bubble.factor** Passed to `calc.wt.bubbles` as argument `fact`.
- **cont.factor** Continuity correction to apply in case one of the alpha’s are equal to zero.
- **pch** Plotting character to use.
- **...** Additional arguments passed to `points`.
plot.cdslist

Plot a cdslist Object

Description
Create a scree plot and bubble plots for all elements in a cdslist object.

Usage

```r
## S3 method for class 'cdslist'
plot(x, which = 2L, ...)
```

Arguments
- `x`: An object of class cdslist.
- `which`: The which argument passed to `plot.cds`
- `...`: Additional arguments passed to `plot.cds`.

print.cds

Print cds Object

Description
Print method for cds objects.

Usage

```r
## S3 method for class 'cds'
print(x, ...)
```

Arguments
- `x`: A cds object.
- `...`: Unimplemented.
### print.cdsdata

**Print dsdata Objects**

**Description**

This is a simple print method for object that inherits from the class cdsdata.

**Usage**

```r
# S3 method for class 'cdsdata'
print(x, ...)  
```

**Arguments**

- **x**: A cdsdata object
- **...**: Unimplemented.

### rcormat

**Randomly Generate Low-Rank Correlation Matrix**

**Description**

Generate a correlation matrix as \( R = LL' \) where the rows of \( L \) are of length 1, \( L \) is of rank \( r \) and the matrix \( L \) is sparse (depending on \( \text{sparse.prop} \)). The loadings in \( L \) are sampled from a standard normal distribution, after which \( \text{sparse.prop} \) is used to set a randomly chosen number of loadings in each row equal to zero. To ensure that a correlation matrix results, the rows are normalized.

**Usage**

```r
rcormat(m, r = 3L, sparse.prop = 0.5)  
```

**Arguments**

- **m**: integer; the number of variables.
- **r**: integer; the required rank.
- **sparse.prop**: the proportion of zeros in the rows of the matrix.

**Value**

A list with the following components:

- **R**: The sampled correlation matrix
- **L**: The loading matrix

**Examples**

```r
R <- rcormat(m = 10)$R  
eigen(R)  
```
rcovmat

Construct a Structured Covariance Matrix for Simulations

Description
Construct a low-rank covariance matrix with specified eigenvalues, where the eigenvectors are simulated from uniform distributions.

Usage
`rcovmat(eigs = k:1, m = 10, k = 2, perc = list(c(0.4, 0.2, 0.4), c(0.2, 0.4, 0.4)), limits = list(l1 = c(0.5, 1), l2 = c(-1, -0.5), l3 = c(-0.1, 0.1)), random = TRUE)`

Arguments
- `eigs` Vector of $k$ eigenvalues.
- `m` Integer; the number of rows and columns of the matrix.
- `k` Integer; the rank of the matrix.
- `perc` List of $k$ vectors giving the sampling proportions for the uniform sampling of the eigenvectors, for each dimension.
- `limits` List of length 2 vectors, one for each uniform sample, giving the lower and upper bounds of the uniform distribution.
- `random` Logical; randomize the order of the loading per dimension or not.

sensory

sensory Data

Description
Data from 268 panellists rating each of 20 different products on 7 attributes. It is presented in a data frame with 268 observations on 140 variables. Each observation represents a different trained panellist. The columns correspond to products and items. The 20 different products are coded by alphabetic letters from A to T, and the items are coded from 1 to 7. So item C5 corresponds to product C being rated on item 5.

Examples
`data(sensory)`
sensory.aux

Auxiliary Information for sensory Data

Description

Auxiliary Information for sensory Data

Format

A data frame with 268 observations on the following 3 variables.

- **Gender**: a factor with levels `F` for females and `M` for males
- **Age**: a factor for age with levels `14 to 24`, `25 to 34`, `35 to 44`, and `45 to 55`
- **Consumption**: a factor for consumption with levels `Heavy consumer`, `Light consumer`, and `Medium consumer`

Source

obtained ~~

Examples

data(sensory.aux)

---

simpca

Simulate Data with a Specific Principal Components Structure and Response Style Contamination

Description

Simulate normally distributed data with specific covariance structure and randomly sampled means. Adds response style contamination.

Usage

```r
simpca(nr.indv = rep(200, 5), m = 10, q = 7, R = rcormat(m = m), err.coeff = 0.1, alphamat = rbind(c(0.5, 2, 4), c(10, 2, 10), c(1, 2, 1), c(4, 2, 0.5), c(0.1, 2, 0.1))[1:length(nr.indv),], randomize = FALSE)
```
**Arguments**

- `nr.indv`: Numeric vector of group sizes.
- `m`: Integer; then number of variables to simulate.
- `q`: Integer; the rating scale used \( 1 : q \).
- `R`: List with entry named ‘R’ which is the simulated correlation matrix.
- `err.coeff`: Standard error for each variable, added unto \( R \).
- `alphamat`: Matrix containing splines coefficients for the construction of response styles.
- `randomize`: Logical; should the rows of the data be randomly permuted or not?

---

**trQnorm**

*Truncated Normal Quantiles*

**Description**

Quantile function of the truncated normal distribution.

**Usage**

```r
trQnorm(p, mean = 1, sd = 1, a = 0, b = 1)
```

**Arguments**

- `p`: Vector of probabilities.
- `mean`: The mean of the distribution.
- `sd`: The standard deviation.
- `a`: Lower truncation point.
- `b`: Upper truncation point.

**Author(s)**

Pieter C. Schoonees
trRnorm  

**Truncated Normal Sampling**

**Description**
Random numbers from truncated univariate normal.

**Usage**
```
trRnorm(n, mu = 0, sd = 1, a = -Inf, b = Inf)
```

**Arguments**
- **n**  
The number of points to sample.
- **mu**  
The mean of the distribution.
- **sd**  
The standard deviation.
- **a**  
The lower truncation point.
- **b**  
The upper truncation point.

updateG  

**Update the Grouping Matrix**

**Description**
Updates the grouping matrix.

**Usage**
```
updateG(G, a, bwts2, Fr.bk, const, n, m, q, random = FALSE, info.level = 3)
```

**Arguments**
- **G**  
Grouping matrix.
- **a**  
Current value of the row scores.
- **bwts2**  
Squared column weights.
- **Fr.bk**  
Product of Fr and bkmatrix.
- **const**  
Constant part of the loss function.
- **n**  
The number of observations.
- **m**  
The number of items.
- **q**  
The number of rating categories.
- **random**  
Logical indicating whether to randomize the observations.
- **info.level**  
Integer controlling the amount of printed.
updateG

Author(s)

Pieter Schoonees
Index

*Topic **datasets**
- sensory, 21
- sensory.aux, 22

*Topic **multivariate**
- cds, 5
- cds.sim, 7
- createcdsdata, 11
- datsim, 11
- G.start, 12
- gen.cop, 13
- group.ALS, 15
- ispline, 16
- Lfun, 16
- Lfun.G.upd, 17
- trQnorm, 23
- trRnorm, 24
- updateG, 24

*Topic **package**
- cds-package, 2

*Topic **splines**
- create.rs, 10

*Topic **utility**
- calc.wt.bubbles, 4
- clean.scales, 9
- create.ind, 10
- plot.cds, 18
- print.cds, 19

- addbounds, 3
- approxloads, 4
- as.matrix, 3

- calc.wt.bubbles, 4, 18
- cds, 5, 9, 14
- cds-package, 2
- cds.sim, 7
- cl_agreement, 9
- cl_class_ids cds, 9
- cl_class_ids cdsdata
  - (cl_class_ids cds), 9
- clean.scales, 9
- copClayton, 13
- create.ind, 10
- create.rs, 10
- createcdsdata, 8, 11
- datsim, 11
- G.start, 12
- gen.cop, 13
- genPCA, 14
- group.ALS, 15
- indmat, 15
- is.cl_hard_partition.cds
  - (cl_class_ids cds), 9
- is.cl_hard_partition.cdsdata
  - (cl_class_ids cds), 9
- is.cl_partition.cds (cl_class_ids cds), 9
- is.cl_partition.cdsdata
  - (cl_class_ids cds), 9
- ispline, 16
- legend, 18
- Lfun, 16
- Lfun.G.upd, 17
- lsei, 6, 13, 15

- matplot, 18
- orthprocr, 4, 17

- plot, 12
- plot.cds, 18, 19
- plot.cdslist, 19
- points, 18
- print.cds, 19
- print.cdsdata, 20
- rcormat, 20

26
INDEX

rcovmat, 21
sensory, 21, 22
sensory.aux, 22
simpca, 14, 22
trQnorm, 23
trRnorm, 24
updateG, 13, 24