Package: choiceDes (via r-universe)

September 2, 2024

Type Package

Title Design Functions for Choice Studies

Version 0.9-3

Date 2018-06-18

Author Jack Horne [aut, cre]

Maintainer Jack Horne <jack@jackhorne.net>

Description Design functions for DCMs and other types of choice studies (including MaxDiff and other tradeoffs).

License GPL (>= 2)

Depends R (>= 2.15.2), AlgDesign

NeedsCompilation no

Date/Publication 2018-06-18 22:49:53 UTC

Repository https://jackhorne.r-universe.dev

RemoteUrl https://github.com/cran/choiceDes

RemoteRef HEAD

RemoteSha f03f31806a117d3a100c9d118dfa0f1df35a618d

Contents

choiceDes-package . . . 2
cp.scree . . . 2
dcm.design . . . 3
dcm.design.cand . . . 4
dcm.design.effcy . . . 6
dcm.design.sort . . . 7
optBlockC . . . 8
optFederovC . . . 9
pw.eval . . . 10
tradeoff.des . . . 11
write.tab . . . 12

Index **14**

choiceDes-package *Design Functions for Choice Studies*

Description

Functions to design DCMs and other types of choice studies (including MaxDiff and other tradeoffs)

cp.scree *Scree plot for tradeoff designs*

Description

Line plot showing the relationship between the criterion used to assess column position balance and the number of iterations in the column position balancing routine.

Usage

```
cp.scree(des)
```

Arguments

des	An R object containing the results from a call to tradeoff.des.

Details

Column position balancing is the most computationally intensive process in calls to tradeoff.des. The number of iterations required for this step is determined by the Rc argument in that function which defaults to the larger of 1,000 or 10 x the number of rows in the design. Large design problems may require a larger number of iterations to achieve optimal column position balance. The plot generated by this function can help to assess whether additional Rc iterations would lead to better column position balance.

See tradeoff.des for additional details.

Value

A line plot showing the relationship between the criterion used to assess column position balance and the number of iterations in the column position balancing routine.

Examples

```
des <- tradeoff.des(12, 4, 10, 9)
cp.scree(des)
```

dcm.design *Optimal fractional factorial design*

Description

Generate an optimal fractional factorial design given vectors of factor lengths.

Usage

```
dcm.design(cand, nb, sets, alts, fname=NULL, Rd=20, print=TRUE)
```

Arguments

cand	A vector of factor lengths, or a list containing vectors of factor lengths if a combinatorial design is needed.
nb	The number of blocks or versions in the final design.
sets	The number of choice sets in *each version* of the final design.
alts	The number of alternatives in each choice set.
fname	A character string, usually ending in ".txt", indiciating the name of the file containing the levels-coded design.
Rd	The number of repeats used by the initial design and blocking algorithms. See arg nRepeats in optFederov and optBlock for additional details
print	Boolean indicating whether there is output to the console during execution.

Details

Generates balanced and blocked choice sets from one or more specified full-factorial candidate set(s) using a modified Federov (1972) algorithm. See optFederov in *AlgDesign* (Wheeler, 2004) for a more complete description of the algorithm. Starting points are chosen randomly (as opposed to by nullification) and may be seeded using set.seed. The D criterion is used for optimization.

See optBlock for a description of the blocking method used.

If fname is not NULL a tab-delimited plain-text file is generated in the working directory containing the levels-coded design.

Large problems will complete faster by setting Rd to a smaller value. However, this may come at the expense of a more efficient design.

Value

levels	A data frame consisting of the levels-coded design with blocks stacked in order. Variables for card, version and task are appended.
effects	A list of the effects-coded, blocked design and diagnostics. See optBlock for additional details.
d.eff	A list containing D efficiency, the variance-covariance matrix, and parameter standard deviations from the effects-coded design. See dcm.design.effcy for additional details.

References

Federov, V.V. (1972). *Theory of optimal experiments.* Academic Press, New York.

Wheeler, R.E. (2004). *AlgDesign.* The R project for statistical computing. (http://www.r-project.org).

See Also

`optFederov`, `optBlock`

Examples

```
## Example 1:
## design from a single candidate set
levs1 <- c(3,3,5,4)
des <- dcm.design(levs1, 10, 6, 2)

## Example 2:
## combinatorial design from more than one candidate set
levs2 <- list(c(3,3), c(5,4))
des <- dcm.design(levs2, 10, 6, 2)
```

`dcm.design.cand` *Optimal fractional factorial design*

Description

Generate an optimal restricted fractional factorial design given a pre-generated candidate set.

Usage

```
dcm.design.cand(cand, nb, sets, alts, fname=NULL, Rd=20, print=TRUE)
```

Arguments

`cand`	A data frame of columns representing factors in the design OR a tab-delimited file readable using `read.table(`*filename*`)`. If `cand` is not a data frame, an external file is assumed.
`nb`	The number of blocks or versions in the final design.
`sets`	The number of choice sets in *each version* of the final design.
`alts`	The number of alternatives in each choice set.
`fname`	A character string, usually ending in ".txt", indiciating the name of the file containing the levels-coded design.
`Rd`	The number of repeats used by the initial design and blocking algorithms. See arg `nRepeats` in `optFederov` and `optBlock` for additional details
`print`	Boolean indicating whether there is output to the console during execution.

Details

Generates balanced and blocked choice sets from a *pre-generated* candidate set. Typical use will involve (1) generating a full factorial candidate set (see `gen.factorial`), (2) manipulating levels as desired (e.g., adding restrictions) and, (3) using the manipulated set as input into the function.

Design optimization and blocking use the same algorithms as those in `dcm.design`.

If `fname` is not `NULL` a tab-delimited plain-text file is generated in the working directory containing the levels-coded design.

Value

`levels`	A data frame consisting of the levels-coded design with blocks stacked in order. Variables for card, version and task are appended.
`effects`	A list of the effects-coded, blocked design and diagnostics. See `optBlock` for additional details.
`d.eff`	A list containing `D` efficiency, the variance-covariance matrix, and parameter standard deviations from the effects-coded design. See `dcm.design.effcy` for additional details.

References

Federov, V.V. (1972). *Theory of optimal experiments.* Academic Press, New York.

Wheeler, R.E. (2004). *AlgDesign.* The R project for statistical computing. (http://www.r-project.org).

See Also

`dcm.design`, `optFederov`, `optBlock`

Examples

```
## generate full factorial candidate set
cand <- gen.factorial(c(3,3,4), factors="all")

## restrict the candidate set so that level 3 in the first factor
## cannot occur with level 1 in the second factor
remove.rows <- which(cand[,1] == 3 & cand[,2] == 1)
cand.restr <- cand[-remove.rows,]

## generate the design from the restricted candidate set
## and check that no design rows violate the restriction
des <- dcm.design.cand(cand.restr, 10, 6, 2)
which(des$levels[,4] == 3 & des$levels[,5] == 1)
```

dcm.design.effcy *INTERNAL: Calculate design efficiencies*

Description

Internal function to calculate mathematical efficiencies of designs.

Usage

```
dcm.design.effcy(des)
```

Arguments

des	An effects-coded design to be evaluated.

Details

Calculates overall D-efficiency, the variance-covariance matrix, and standard deviations for each parameter from an effects coded design.

Called internally by dcm.design and dcm.design.cand.

Value

D	Overall D-efficiency: *det(M)^(-1/k)*, where *det(M)* is the determinant of the dispersion matrix *X'X*, and *k* is the number of parameters.
V	Variance-covariance matrix derived from *M*.
s	Parameter standard deviations: *sqrt(diag(V))*.

See Also

dcm.design, dcm.design.cand

Examples

```
des <- dcm.design(c(3,3,4,3), 10, 8, 3)$effects$design
eff <- dcm.design.effcy(des)
```

dcm.design.sort *INTERNAL: Append other design variables*

Description

Internal function to append card, version and task variables to design.

Usage

```
dcm.design.sort(design, nb, sets, alts)
```

Arguments

design	A levels-coded design generated by either dcm.design or dcm.design.cand.
nb	The number of blocks or versions in the final design.
sets	The number of choice sets in *each version* of the final design.
alts	The number of alternatives in each choice set.

Details

Randomizes the order of rows within each block of the design using runif and appends card, version and task variables as appropriate.

This function is called internally by dcm.design and dcm.design.cand.

Value

A data frame containing the levels-coded design with rows randomized within block, and with card, version and task variables appended.

See Also

dcm.design, dcm.design.cand, tradeoff.des

Examples

```
## INTERNAL USE ONLY
```

optBlockC *Optimal design blocking*

Description

INTERNAL: Simplified wrapper for blocking of experimental designs using optBlock in the *AlgDesign* package.

Usage

```
optBlockC(withinData, blocksizes, nRepeats=5)
```

Arguments

withinData	A matrix or data frame describing the variables. Data types allowed include: factors, effects-coded designs, or dummy-coded designs. Other data types will lead to errors or unbalanced designs. If the columns are not named, they will be assumed to have the names X1, X2, etc. The number of rows in withinData must be at least as large as the sum of the number of terms plus the number of blocks.
blocksizes	A vector giving the block sizes for each block. The length of blocksizes specifies the number of blocks.
nRepeats	The number of times the entire process is repeated.

Details

Simplified wrapper for optBlock that optimizes blocks on a pre-existing design or a set of factors using the D criterion. Does not permit whole plot factors to interact with within plot factors. See optBlock for additional details.

Value

D	*det(M)^(1/k)*, where *det(M)* is the determinant of the normalized dispersion matrix *M*, or *m=X'X/N*, where each row of *X* has the appropriate block mean subtracted.
diagonality	*|M|/P^(1/k)*, where *P* is the product of the diagonal elements of *M*.
Blocks	A list of blocks, labeled B1, B2, etc.
design	A data frame. The design with blocks in stacked order.
rows	Numeric row numers of the design rows corresponding to withinData rows.

References

Wheeler, R.E. (2004). optBlock. *AlgDesign.* The R project for statistical computing. (http://www.r-project.org).

Examples

```
##INTERNAL USE ONLY
```

`optFederovC` *Optimal design*

Description

INTERNAL: Simplified wrapper for calculating exact algorithmic designs using Federov's exchange algorithm. Based on `optFederov` in the *AlgDesign* package.

Usage

```
optFederovC(modelData, nTrials, nRepeats=5)
```

Arguments

`modelData`	The candidate list. A matrix or data frame describing the variables. If a matrix is input and the columns are not named, they will be assigned names X1, X2, etc. Permitted data types include factors or levels- or effects-coded designs.
`nTrials`	The number of trials in the final design.
`nRepeats`	The number of times the whole process is repeated.

Details

Generates exact algorithmic designs using Federov's exchange algorithm, and optimizing the `D` criterion. See `optFederov` for algorithmic details. A vignette is also available by typing `vignette("AlgDesign")`.

Input data, i.e., `modelData`, must be of a form that `model.matrix(~., modelData` results in an effects-coded design or candidate set.

Value

`D`	The *k*th root of the generalized variance: *det(M)^(1/k)*, where *det(M)* is the determinant of the normalized dispersion matrix, or *m=Z'Z/n*, where *Z=X[rows,]*.
`A`	The average coefficient variance: *trace(M')/k*, where *M'* is the inverse of *M*.
`Ge`	The minimax normalized variance over *X*, expressed as an efficiency with respect to the optimal approximate theory design. It is defined as *k/max(d)*, where *max(d)* is the maximum normalized variance over *X*, or the maximum of *x'(M')x*, over all rows *x'* of *X*.
`Dea`	A lower bound on `D` efficiency for approximate theory designs. It is equal to *exp(1-1/Ge)*.
`design`	The design.
`rows`	A numerical vector of the design row numbers from `modelData`.

References

Wheeler, R.E. (2004). optFederov. *AlgDesign.* The R project for statistical computing. (http://www.r-project.org).

Examples

```
##INTERNAL USE ONLY
```

`pw.eval` *INTERNAL: Evaluate two-way frequencies*

Description

Internal function to evaluate whether the two-way (pairwise) frequencies of items in a matrix are balanced.

Usage

```
pw.eval(items, shown, drows, des)
```

Arguments

`items`	The total number of items shown in a tradeoff exercise.
`shown`	The number of items shown in each tradeoff task.
`drows`	The number of rows in the tradeoff design matrix.
`des`	A matrix consisting of the levels-coded tradeoff design.

Details

Evaluates the two-way (pairwise) frequencies of items in a tradeoff design matrix and returns those frequencies as well as the off-diagonal mean and standard deviation of the frequencies.

This function is called internally by `tradeoff.des`.

Value

`tbl`	A matrix of the two-way frequencies (pairs of items) in the tradeoff design.
`od.mean`	The mean of the off-diagonal elements in `tbl`.
`od.stdv`	The standard deviation of the off-diagonal elements in `tbl`.

See Also

`tradeoff.des`

Examples

```
## INTERNAL USE ONLY
```

`tradeoff.des` *MaxDiff and other tradeoff designs*

Description

Generate a design to be used for MaxDiff and related tradeoff exercises.

Usage

```
tradeoff.des(items, shown, vers, tasks, fname=NULL, Rd=20, Rc=NULL, print=TRUE)
```

Arguments

`items`	The total number of items in the tradeoff exercise.
`shown`	The number of items shown in each tradeoff task.
`vers`	The number of blocks or versions in the final design.
`tasks`	The number of tradeoff tasks in *each version* of the final design.
`fname`	A character string, usually ending in ".txt", indicating the name of the file containing the tradeoff design.
`Rd`	The number of iterations in the design and blocking processes.
`Rc`	The number of iterations in the item by column position optimization routine.
`print`	Boolean indicating whether there is output to the console during execution.

Details

Replicates the functionality of Sawtooth Software MaxDiff Designer for designing MaxDiff and related tradeoff tasks.

A modified Federov (1972) algorithm is applied to a factor equal in length to the number of items to optimize the BIB design at `vers` x `tasks` rows and `shown` columns.

The optimized design is evaluated for one-way frequencies (equal representation of each item across all versions and column positions). Designs are also optimized for two-way or pairwise balance across all tasks. Column position balance is the more computationally-intensive process. The number of iterations required for this step is determined by the `Rc` argument which defaults to the larger of 1,000 or 10 x the number of rows in the design. Large design problems may require a larger number of iterations to achieve optimal column position balance.

Once an optimal design has been found, it is blocked into versions using `optBlock` to ensure equal representation of items *within* each version. See Wheeler (2004) for a more complete description of the modified Federov and blocking algorithms used in optimizing these designs.

If `fname` is not `NULL` a tab-delimited plain-text file is generated in the working directory containing the levels-coded design.

Value

design	A matrix consisting of the optimized design and additional variables for card, version and task.
balance	Tables of one-way item frequencies, two-way (pairwise) item frequencies, and item frequencies by column position. Means and standard deviations are calculated from all elements of the one-way and column position tables, and from the off-diagonal elements of the two-way (pairwise) table.
Rc.crit	The criterion that is minimized to achieve column position balance is output as a vector (crit.vec) along with the number of iterations executed since the last change in this criterion (crit.stable). If crit.stable is large or is a large proportion of the total number of iterations (Rc), the solution is stable in terms of column position balance. If crit.stable is small, the solution is likely unstable and column position balance could be improved by increasing Rc. See also cp.scree which produces a line plot of crit.vec as a function of Rc.
time.elapsed	The time required for the function to execute.

References

Federov, V.V. (1972). *Theory of optimal experiments.* Academic Press, New York.

Wheeler, R.E. (2004). *AlgDesign.* The R project for statistical computing. (http://www.r-project.org).

Examples

```
## Example 1:
## typical MaxDiff design with 12 items
des <- tradeoff.des(12, 4, 10, 9)

## Example 2:
## typical paired comparisons design with 14 items
des <- tradeoff.des(14, 2, 6, 14)
```

---

write.tab *INTERNAL: Write a data frame as tab-delimited file*

Description

Internal function that acts as an alias to write.table, appending extra arguments.

Usage

```
write.tab(x, f)
```

Arguments

x	A data frame object in R.
f	A character string, usually ending in "*.txt", indicating the name of the file to be generated.

Details

Writes a data frame to the file indicated by `f`, using `write.table` and appending the following arguments: `row.names=FALSE`, `col.names=TRUE`, `quote=FALSE`, and `sep="\t"`.

Called internally by `dcm.design`, `dcm.design.cand`, and `tradeoff.des`.

Value

Does not return any value.

See Also

`dcm.design`, `dcm.design.cand`, `tradeoff.des`

Examples

```
## INTERNAL USE ONLY
```

Index

choiceDes-package, 2
cp.scree, 2

dcm.design, 3
dcm.design.cand, 4
dcm.design.effcy, 6
dcm.design.sort, 7

optBlockC, 8
optFederovC, 9

pw.eval, 10

tradeoff.des, 11

write.tab, 12