Package 'choiceDes'

February 19, 2015

Type Package

Title Design Functions for Choice Studies

Version 0.9-1

Date 2014-11-24

Maintainer Jack Horne <jack@jackhorne.net>

Description This package consists of functions to design DCMs and other types of choice studies (including MaxDiff and other tradeoffs)

License GPL (>= 2)

Depends R (>= 2.15.2), AlgDesign

Author Jack Horne [aut, cre]

NeedsCompilation no

Repository CRAN

Date/Publication 2014-11-24 18:38:56

R topics documented:

choiceDes-package *Design Functions for Choice Studies*

Description

This package consists of 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

This function 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 specified in the call 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

This function generates balanced and blocked choice sets from a *pre-generated* candidate set. Typical use will involve (1) generating a full factorial candidate set (see `dcm.design.gencand`), (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`.

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 <- dcm.design.gencand(c(3,3,4))

## 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

This function 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.gencand *Full factorial design*

Description

Generate a full factorial design set given a vector of factor lengths.

Usage

```
dcm.design.gencand(levs)
```

Arguments

levs	A vector of factor lengths.

Details

This function can be used in concert with dcm.design.cand when the candidate set must be manipulated or restricted before generating a fractional factorial design. Generate the full factorial design using this function, manipulate it with appropriate R code. Or export the candidate generated by this function to Excel or another application, and save the result as a tab-delimited text file that can be imported as part of the call to dcm.design.cand.

Use dcm.design by itself to generate fractional factorial designs from full factorial candidate sets that do not require further manipulation. The required candidate set(s) are generated internally by that function given a vector of factor lengths as input.

Value

A data frame of factors in a full factorial design.

See Also

gen.factorial

Examples

```
levs <- c(3,4,3,2)
cand <- dcm.design.gencand(levs)
```

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	The 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

This function 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
```

get.file *INTERNAL: Get data files*

Description

Internal function to get required design candidate, data, design and any other files from either R sources or as tab-delimited files stored in the working directory.

Usage

```
get.file(dat.in)
```

Arguments

dat.in	The input data to be read referencing either an R data frame or a string referencing a tab-delimited file stored in the working directory.

Details

This internal function obtains data from different sources and formats them as data frames for use in other functions.

If dat.in is already a data frame in R, the data frame is returned. No other types of R objects are permitted.

If data are stored in a tab-delimited file in the working directory, dat.in references the filename as a character string enclosed in quotes. The tab-delimited file must be readable using read.table(*filename*, header=TRUE, sep="\t").

Value

The input data (dat.in) coded as a data frame.

See Also

dcm.design.cand

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

This function 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

This function 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.

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

This function 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"`.

This function is 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