Package ‘ExpertChoice’

January 20, 2025

Title Design of Discrete Choice and Conjoint Analysis

Version 0.2.0

Description Supports designing efficient discrete choice experiments (DCEs). Experimental designs can be formed on the basis of orthogonal arrays or search methods for optimal designs (Federov or mixed integer programs). Various methods for converting these experimental designs into a discrete choice experiment. Many efficiency measures! Draws from literature of Kuhfeld (2010) and Street et. al (2005) <doi:10.1016/j.ijresmar.2005.09.003>.

ByteCompile true

License MIT + file LICENSE

Encoding UTF-8

LazyData true

BugReports `https://github.com/JedStephens/ExpertChoice/issues`

Depends R (>= 3.6.0)

Imports stats, far, dplyr, DoE.base, rlist, purrr

RoxygenNote 7.1.0

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Jed Stephens [aut, cre] (<`https://orcid.org/0000-0002-0131-703X`>)

Maintainer Jed Stephens <`STPJED001@myuct.ac.za`>

Repository CRAN

Date/Publication 2020-04-03 14:30:02 UTC

Contents

`augment_levels` *Augment levels and B-matrix to Full Factorial Design.*

Description

Augments the full factorial design with a column summarising the levels of that design. Importantly, it also adds the B-matrix as an attribute.

Usage

```
augment_levels(full_factorial)
```

Arguments

`full_factorial`	a ‘data.table‘ generated from the ‘full_factorial‘ function.

Value

a ‘data.frame‘ with an additional column identifying the level and the B-matrix attribute.

References

Street, D. J.; Burgess, L. & Louviere, J. J. Quick and easy choice sets: Constructing optimal and nearly optimal stated choice experiments International Journal of Research in Marketing, 2005 , 22 , 459 - 470

Examples

```
# See Practical Introduction to ExpertChoice Vignette. Step 2.

#Step 1
attrshort  = list(condition = c("0", "1", "2"),
technical =c("0", "1", "2"),
provenance = c("0", "1"))

#Step 2! - the augment_levels function
#' # ff stands for "full fatorial"
 ff  <-  full_factorial(attrshort)
 af  <-  augment_levels(ff)
# af stands for "augmented factorial"
af
# Compare ff and af. - do not confuse them. They serve different purposes.
```

check_overshadow *Check Overshadow - Pareto Dominate Solutions*

Description

Check Overshadow - Pareto Dominate Solutions

Usage

```
check_overshadow(choice_sets)
```

Arguments

`choice_sets`	An object of the choiceset class made by one of the DCE methods.

Value

A matrix of logicals indicating which if any card for a given row is Pareto dominate.

Examples

```
#See Step 7 of the Practical Introduction to ExpertChoice Vignette.
# Step 1
attrshort  = list(condition = c("0", "1", "2"),
technical =c("0", "1", "2"),
provenance = c("0", "1"))

#Step 2
#' # ff stands for "full fatorial"
 ff  <-  full_factorial(attrshort)
 af  <-  augment_levels(ff)
# af stands for "augmented factorial"

# Step 3
# Choose a design type: Federov or Orthogonal. Here an Orthogonal one is used.
nlevels <- unlist(purrr::map(ff, function(x){length(levels(x))}))
fractional_factorial <- DoE.base::oa.design(nlevels = nlevels, columns = "min34")

# Step 4 & 5
# The functional draws out the rows from the original augmented full factorial design.
colnames(fractional_factorial) <- colnames(ff)
fractional <- search_design(ff, fractional_factorial)
# Step 5 - Skipped, but important, see vignette.

# Step 6
# Two modulators c(1,1,1) and c(0,1,1) are specified.
dce_modulo <- modulo_method(
fractional,
list(c(1,1,1),c(0,1,1))
)
```

```
# Step 7! -- Check for Pareto dominate solutions
check_overshadow(dce_modulo)
```

---

construct_question_frame

Convert from choice_sets to a question data

Description

Convert from choice_sets to a question data

Usage

```
construct_question_frame(
  augmented_full_factorial,
  choice_sets,
  randomise_choice_sets = TRUE
)
```

Arguments

`augmented_full_factorial`	The augmented full factorial object.
`choice_sets`	The choice sets list generated from one of the methods. (See Step 6 of the tutorial)
`randomise_choice_sets`	A binary variable indicating if the order of the choice sets should be randomised. Some methods create choice sets which have a systematic order. Randomising the order of the choice sets does not change the alternatives within the choice sets. It simply rearranges the choice_set object in a random manner.

Value

a data.frame object

Examples

```
#See Step 9 of Practical Introduction to ExpertChoice vignette.

# Step 1
attrshort  = list(condition = c("0", "1", "2"),
technical =c("0", "1", "2"),
provenance = c("0", "1"))

#Step 2
# ff stands for "full fatorial"
 ff  <-  full_factorial(attrshort)
 af  <-  augment_levels(ff)
# af stands for "augmented factorial"

# Step 3
# Choose a design type: Federov or Orthogonal. Here an Orthogonal one is used.
nlevels <- unlist(purrr::map(ff, function(x){length(levels(x))}))
fractional_factorial <- DoE.base::oa.design(nlevels = nlevels, columns = "min34")

# Step 4 & 5
# The functional draws out the rows from the original augmented full factorial design.
colnames(fractional_factorial) <- colnames(ff)
fractional <- search_design(ff, fractional_factorial)
# Step 5 (skipped, but important, see vignette)

# Step 6
# Two modulators c(1,1,1) and c(0,1,1) are specified.
dce_modulo <- modulo_method(
fractional,
list(c(1,1,1),c(0,1,1))
)

# Step 7 and Step 8 are very important for the design, but skipped here.

# Step 9! -- Construct a question frame to use with your study.
# Note the use of af here.
questions <- construct_question_frame(af, dce_modulo)
levels(questions$condition) <- c("bad", "good", "excellent")
levels(questions$technical) <- c("poor", "fair", "skilled")
levels(questions$provenance) <- c("none", "present")
questions
```

dce_efficiency *Efficiency Measures for Discrete Choice Experiments*

Description

Efficiency Measures for Discrete Choice Experiments

Usage

```
dce_efficiency(augmented_full_factorial, choice_sets)
```

Arguments

augmented_full_factorial	The level augmented full factorial. See tutorial step 2.
choice_sets	A list of choice sets generated by one of the methods used to convert from fractional factorial designs.

Value

a list of named output.

References

Street, D.J., Burgess, L. and Louviere, J.J., 2005. Quick and easy choice sets: constructing optimal and nearly optimal stated choice experiments. International Journal of Research in Marketing, 22(4), pp.459-470.

Examples

```
# See Step 8 of the Practical Introduction to ExpertChoice vignette.
# Step 1
attrshort  = list(condition = c("0", "1", "2"),
technical =c("0", "1", "2"),
provenance = c("0", "1"))

#Step 2
# ff stands for "full fatorial"
 ff  <-  full_factorial(attrshort)
 af  <-  augment_levels(ff)
# af stands for "augmented factorial"

# Step 3
# Choose a design type: Federov or Orthogonal. Here an Orthogonal one is used.
nlevels <- unlist(purrr::map(ff, function(x){length(levels(x))}))
fractional_factorial <- DoE.base::oa.design(nlevels = nlevels, columns = "min34")

# Step 4 & 5
# The functional draws out the rows from the original augmented full factorial design.
colnames(fractional_factorial) <- colnames(ff)
fractional <- search_design(ff, fractional_factorial)
# Step 5 (skipped, but important, see vignette)

# Step 6
# Two modulators c(1,1,1) and c(0,1,1) are specified.
dce_modulo <- modulo_method(
fractional,
list(c(1,1,1),c(0,1,1))
)

# Step 7 (skipped)

# Step 8! -- Inspect the D-efficiency using the Street et. al method of the DCE design.
# NOTE: the af is used at this stage not the ff.
dce_efficiency(af, dce_modulo)
```

`fractional_factorial_efficiency`
Fractional Factorial Design Efficiency

Description

Fractional Factorial Design Efficiency

Usage

```
fractional_factorial_efficiency(formula, searched_fractional_factorial)
```

Arguments

`formula`	A specification, in formula form, of the desired effects sought to be estimated.
`searched_fractional_factorial`	a fractional factorial generated as the result of a ‘search_design‘.

Value

a list with the following objects: 1. X - This is the formula expanded version of the fractional factorial which was passed to the function. 2. information_mat - This is the information matrix described by the associated note. Note: it is rounded to three decimal places to ease reading. 3. inv_information_mat - This is the inverse of the information matrix. Note: it is rounded to three decimal places to ease reading. 4. lamda_mat - This is the diagonal elements of the Lamda Matrix described by Kuhfeld (pg. 62). The elements are the eigen values of the inv_information_mat. 5. inv_diag - This is the diagonal elements of the inv_information_mat. (May be of use to some researchers...) 6. GWLP - This is the generalised world lengths for the searched design. (Note: this would not change depending on what is in the formula expansion.) 7. A_eff - This is the A-efficiency of the design given the particular formula expansion. 8. D_eff - This is the D-efficiency of the design given the particular formula expansion.

References

Kuhfeld, W. F. Marketing Research Methods in SAS Experimental Design, Choice, Conjoint, and Graphical Techniques 2010.

Examples

```
# See step 5 of the Practical Introduction to ExpertChoice vignette.

# Step 1
attrshort  = list(condition = c("0", "1", "2"),
technical =c("0", "1", "2"),
provenance = c("0", "1"))

#Step 2
# ff stands for "full fatorial"
```

```
 ff  <-  full_factorial(attrshort)
 af  <-  augment_levels(ff)
# af stands for "augmented factorial"

# Step 3
# Choose a design type: Federov or Orthogonal. Here an Orthogonal one is used.
nlevels <- unlist(purrr::map(ff, function(x){length(levels(x))}))
fractional_factorial <- DoE.base::oa.design(nlevels = nlevels, columns = "min34")

# Step 4
# The functional draws out the rows from the original augmented full factorial design.
colnames(fractional_factorial) <- colnames(ff)
fractional <- search_design(ff, fractional_factorial)

# Step 5! - The fractional_factorial_efficiency function
# The formula requires reference to the original attributes of the design.
# Check for the main effects.
fractional_factorial_efficiency(~ condition + technical + provenance, fractional)
# Check for the main effects with some interaction.
fractional_factorial_efficiency(~ condition + technical * provenance, fractional)
```

`full_factorial` *Full Factorial Design*

Description

Generates the full factorial design with all the factors coded using standardised orthogonal contrast coding.

Usage

```
full_factorial(attributes_list)
```

Arguments

`attributes_list`	A named list: giving the variable name and the levels as characters. The levels should start from the base of either "0" or "1" and go up in integer values.

Value

a ‘data.frame‘ with the full factorial design and factors coded using standardised orthogonal contrast coding.

References

Kuhfeld, W. F. Marketing Research Methods in SAS Experimental Design, Choice, Conjoint, and Graphical Techniques 2010.

Jörg Suckut (https://stats.stackexchange.com/users/237455/j

Examples

```
# See step 1 of the Practical Introduction to ExpertChoice vignette.
attrshort  = list(condition = c("0", "1", "2"),
technical =c("0", "1", "2"),
provenance = c("0", "1"))
full_factorial(attrshort)
```

---

modulo_method *Modulo Method - Described by Street et al.*

Description

Modulo Method - Described by Street et al.

Usage

```
modulo_method(fractional_fatorial, generators)
```

Arguments

`fractional_fatorial`	The usual.
`generators`	a list of generators

Value

a choiceset list.

Examples

```
# See step 6 of the Practical Introduction to ExpertChoice vignette.

# Step 1
attrshort  = list(condition = c("0", "1", "2"),
technical =c("0", "1", "2"),
provenance = c("0", "1"))

#Step 2
# ff stands for "full fatorial"
 ff  <-  full_factorial(attrshort)
 af  <-  augment_levels(ff)
# af stands for "augmented factorial"

# Step 3
# Choose a design type: Federov or Orthogonal. Here an Orthogonal one is used.
nlevels <- unlist(purrr::map(ff, function(x){length(levels(x))}))
fractional_factorial <- DoE.base::oa.design(nlevels = nlevels, columns = "min34")

# Step 4
# The functional draws out the rows from the original augmented full factorial design.
colnames(fractional_factorial) <- colnames(ff)
fractional <- search_design(ff, fractional_factorial)
# Step 5 - Skipped, but important, see vignette.

# Step 6! -- The modulo_method function
# Two modulators c(1,1,1) and c(0,1,1) are specified.
dce_modulo <- modulo_method(
fractional,
list(c(1,1,1),c(0,1,1))
)
dce_modulo
```

---

search_design *Search Full Factorial for Fractional Factorial Design*

Description

Returns a consistent fractional factorial design from the input fractional factorial design. The key advantage of this function is that it ensures factors are coded and enchances the attributes of the output.

Usage

```
search_design(full_factorial, fractional_factorial_design)
```

Arguments

`full_factorial`	a ‘data.table‘ generated by the ‘full_factorial‘ function
`fractional_factorial_design`	a means of creating a fractional design using either orthogonal arrays or Federov. See the tutorial for examples.

Value

a ‘data.frame‘ with only the rows of your chosen fractional factorial design.

Examples

```
# The use of this function depends on what the input to the argument fractional_factorial_design
# will be. See Step 4 of Practical Introduction to ExpertChoice vignette.

# Step 1
attrshort  = list(condition = c("0", "1", "2"),
technical =c("0", "1", "2"),
provenance = c("0", "1"))

#Step 2
# ff stands for "full fatorial"
```

```
 ff  <-  full_factorial(attrshort)
 af  <-  augment_levels(ff)
# af stands for "augmented factorial"

# Step 3
# Choose a design type: Federov or Orthogonal. Here an Orthogonal one is used.
nlevels <- unlist(purrr::map(ff, function(x){length(levels(x))}))
fractional_factorial <- DoE.base::oa.design(nlevels = nlevels, columns = "min34")

# Step 4! - The search_design function.
# The functional draws out the rows from the original augmented full factorial design.
colnames(fractional_factorial) <- colnames(ff)
fractional <- search_design(ff, fractional_factorial)
```

Index