Package 'VMinR'

February 7, 2018

Type Package

Title The package contains a set of functions which are required in a ValueManager study

Version 1.1.0

Author Maximilian Rausch

Maintainer Maximilian Rausch <maximilian.rausch@tns-infratest.com>

Description The package contains a set of functions which are required in a ValueManager study. Importing data, simulating shares, etc.

License file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Depends R (>= 2.10)

Imports data.table, corrgram, AlgDesign, ggplot2, foreign

Suggests testthat

R topics documented:

beer_data *VM test data - ValuePricer "beer study""*

Description

Dataset containing the variable contents (data model) settings to perform input validation.

Format

A list with the data of the beer study.

dat A matrix of the imported dat-file

utils_mat A matrix including the utilities from the dat file

utils_list A list including the individual utilities from the dat file. One list element per respondent

iaw A matrix including the individual awareness factors from the dat file.

idis A matrix including the individual distribution factors from the dat file.

seg A matrix containing the segment data

weight A vector containing the weight per respondent

def A list containing the values from `VD.read_def`

pricemat_tested A matrix containing the prices used in the model

pricerange_tested A vector containing the minimal and maximal price used in the model

pr_range_mat A matrix containing the minimal and maximal prices per SKU used in the model

SKUs A vector containing the SKU labels from the def-file.

nlev A vector containing the number of levels per attribute.

ID A vector containing the ID per respondent

Examples

```
data(beer_data)
str(beer_data)
```

calibEXE *Calib EXE*

Description

Calibrates the utilities based on purchase intention questions (calibEXE)

Usage

```
calibEXE(BWconcepts = NULL, PI = NULL, utils = NULL, cut = 42,
  nlev = NULL)
```

Arguments

BWconcepts	A matrix/data.frame containing the best and worst concepts per respondents.
PI	A matrix containing the answer to the purchase intention questions for the best/worst concepts.
utils	A matrix containing the utilities per respondent of the model.
cut	An integer value indicating the maximal value for utilities after calibration.
nlev	A vector indicating the number of levels per attribute

Value

A list including 13 elements

BWconcepts	A matrix/data.frame containing the best and worst concepts per respondents which have been passed to the function
utils_calib	A matrix containing the CALIBRATED utilities per respondent of the model.
utils	A matrix containing the original utilities per respondent of the model.
check_order_BW	A vector containing boolean values indicating if the utilities for the best concept have been greater than the ones of the worst concept.
check_order_PI	A vector containing boolean values indicating if the purchase intentions for the best concept have been greater than the ones of the worst concept.
utl_sum	A matrix containing the corrected utility sums for the best and worst concept. Values in the wrong order are set to be equal.
PurchaseInt	A matrix containing the corrected purchase intentions for the best and worst concept. (1 = .95, 2 = .5, 3 = .3, 4 = .15, 5 = .05) Values in the wrong order are set to be equal.
utl_sum_ORIG	A matrix containing the original (uncorrected) utility sums for the best and worst concept.
PurchaseInt_ORIG	A matrix containing the original (uncorrected) purchase intentions for the best and worst concept.
a	The slope of the linear function used for the calibration.
b	The intercept of the linear function used for the calibration.
nlev	A vector indicating the number of levels per attribute
cut	An integer indicating the cutoff value used for the maximal value for utilities after calibration.

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
calibData <- calibEXE(BWconcepts = BWconcepts[, c(paste0("B_Att_", sequence(natt)),
                                                  paste0("W_Att_", sequence(natt)))],
PI = BWconcepts[, c("PI_B", "PI_W")],
utils = dat_input$utils_mat,
cut = 42, nlev = nlev)
```

```
## End(Not run)
```

`ConceptOpt_ISBC` *Concept optimization with ISBC*

Description

Runs the process for "concept optimization with ISBC" studies - Always weighted!

Usage

```
ConceptOpt_ISBC(dat = NULL, def = NULL, FrameData = NULL, calib = TRUE,
  nlev = NULL, cut = 42, ID = "Respondent_Serial")
```

Arguments

`dat`	A string indicating the path/filename of the ValueDriver dat-file.
`def`	A string indicating the path/filename of the ValueDriver def-file.
`FrameData`	A string indicating the path/filename of the frame questionnaire data.
`calib`	An boolean value indicating if the utilities should be calibrated on purchase intention - default = `TRUE`.
`nlev`	A vector indicating the number of levels per attribute
`cut`	An integer value indicating the maximal value for utilities after calibration. (required for `calibEXE`)
`ID`	A string indicating the name of teh ID variable in the SPSS dataset - default = `"Respondent_Serial"`.

Value

A list including 13 elements

`call`	A string returning the call including the inputs
`calibData`	A list with the values of the `calibEXE` call. (`NULL` if `calib == FALSE`))
`excel_export`	A data.frame containing the export matrix; concept definition + aggregated shares + sd + individual shares.
`excel_export_calib`	A data.frame containing the calibrated export matrix; concept definition + aggregated shares + sd + individual shares. (`NULL` if `calib == FALSE`))
`dat_file_calib`	A data.frame containing the export information for the calibrated ValueDriver dat-file. (`NULL` if `calib == FALSE`))
`overview`	A data.frame with additional information on the calibExe step. (`NULL` if `calib == FALSE`))
`path`	A string returning the working directory where the export files had been saved to.
`weight`	A vector with the weight used for aggregation

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
nlev <- c(4, 4, 4, 2, 4, 2, 4, 6)
ConcOptTest <- ConceptOpt_ISBC("Innogy.dat",
                               "Innogy.def",
                               FrameData = "./data_calib.SAV",
                               calib = TRUE,
                               nlev = nlev,
                               cut = 42,
                               ID = "Respondent_Serial")
str(ConcOptTest)

## End(Not run)
```

---

convertSSItoDesign *Convert SSI like design file to dummy coding*

Description

Convert SSI like design file to dummy coding.

Usage

```
convertSSItoDesign(df.in, no.output = FALSE, none.col = NULL, nlev = NULL)
```

Arguments

df.in	A matrix/data.frame containg the design file to be recoded.
no.output	not used
none.col	not used
nlev	A string value with the path to the DEF file to import.

Value

A data.frame of the recoded dummy design

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
nlev <- VMinR$VDdata$nlev
basecase <- VMinR$basecase

basecase_dummy_2 <- as.matrix(cbind(convertSSItoDesign(basecase, nlev = nlev), 0))
```

`correlation_HeatMap` *Correlation heatmap*

Description

Calculates correlation heatmap and MDS coordinates for ValuePricer projects.

Usage

```
correlation_HeatMap(input_file = NULL, sepOUT = ";", decOUT = ",",
  usedraws = FALSE, clustered = FALSE)
```

Arguments

`input_file`	the path to the heatmap input file. A specific file which contains all the necessary information. (See VM sharepoint)
`sepOUT`	separator for output csv-files - default = `";"`
`decOUT`	decimal sign for output csv-files - default = `","`
`usedraws`	A boolean variable indicating whether DRAWS should be used as well or not - default = `FALSE`
`clustered`	A boolean variable indicating whether clustered heatmaps should be calculated or not - default = `FALSE`

Value

A list including 23 elements

`input_file`	the path to the heatmap input file.
`cor`	A matrix of the correlations for the scenario specified in the input file.
`cor_draws`	A matrix of the correlations for the scenario specified in the input file based on the DRAWS. If `usedraws == TRUE`
`cor_clustered`	A matrix of the correlations for the scenario specified in the input file for the CLUSTERS. If `clustered == TRUE`
`cor_draws_clustered`	A matrix of the correlations for the scenario specified in the input file for the CLUSTERS based on the DRAWS If `usedraws == TRUE` and `clustered == TRUE`
`base_sim`	A vector with the unweighted aggregated shares for the scenario specified in the input file.
`base_sim_draws`	A vector with the unweighted aggregated shares for the scenario specified in the input file based on the DRAWS. If `usedraws == TRUE`
`draws`	A matrix containing the draws used for the study
`utls`	A matrix containing the utilities used for the study
`Xbeta`	`X * beta` matrix (utility sums).
`Xbeta_draws`	`X * beta` matrix for the draws (utility sums). If `usedraws == TRUE`
`Xbeta_clustered`	`X * beta` matrix (utility sums). If `clustered == TRUE`

`Xbeta_draws_clustered`
: `X * beta` matrix for the draws (utility sums). If `usedraws == TRUE` and `clustered == TRUE`

`brand_list`
: A list with one item per cluster containing the respectiv indicies for the SKUs of the clusters.

`MDS_coord`
: A matrix containing the MDS coordinates

`MDS_coord_draws`
: A matrix containing the MDS coordinates for DRAWS. If `usedraws == TRUE`

`MDS_coord_clustered`
: A matrix containing the clustered MDS coordinates. If `clustered == TRUE`

`MDS_coord_draws_clustered`
: A matrix containing the clustered MDS coordinates for DRAWS. If `usedraws == TRUE` and `clustered == TRUE`

`SKUlabels`
: A vector with the SKU labels passed by the input file.

`ClusterLabels`
: A vector with the cluster labels passed by the input file.

`prices`
: A matrix containing the prices used in the model.

`simPrices`
: A vector containing the prices used for the heatmap calculation.

`simSKUs`
: A vector with the SKU indicies used for the heatmap calculation.

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
heat <- correlation_HeatMap("Input_DUMMY.txt",
                            usedraws=FALSE,
                            clustered = FALSE,
                            sepOUT=",",
                            decOUT=".")

## End(Not run)
```

---

get_DriverData *Import ValueDriver data and definitions*

Description

Imports the dat/def files for a ValueDriver study and extracts the relevant information.

Usage

```
get_DriverData(dat_file = NULL, def_file = NULL, nlev = NULL,
  none = TRUE)
```

Arguments

dat_file	A string value with the path to the DAT file to import.
def_file	A string value with the path to the DEF file to import.
nlev	A vector indicating the number of levels per attribute
none	A boolean variable indicating whether NONE is included in the dat file or not - default = TRUE

Value

A list including elements

dat	A matrix of the imported dat-file
utils_mat	A matrix including the utilities from the dat file
utils_list	A list including the individual utilities from the dat file. One list element per respondent
seg	A matrix containing the segment data
weight	A vector containing the weight per respondent
def	A list containing the values from VD.read_def
nlev	A vector containing the number of levels per attribute.
nseg	A variable indicating the number of segments in the dat file
ID	A vector containing the ID per respondent
RLH	A vector containing the root likelihood (RLH) from the HB estiomation per respondent.
check1	check if the extracted info is consistent with the dat file (number of segments)
check2	check if the extracted info is consistent with the dat file (number of total levels)

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
VDdata <- get_DriverData(dat_file = "TEST_timtim_gew.dat",
                         def_file = "TEST_timtim_gew.def",
                         nlev = c(4, 6, 4, 2, 2, 2, 2, 2, 2, 2, 2, 5, 5, 5, 5, 5, 5),
                         none = TRUE)
str(VDdata)

## End(Not run)
```

`get_PricerData` *Import ValuePricer data and definitions*

Description

Imports the dat/def files for a ValuePricer study and extracts the relevant information.

Usage

```
get_PricerData(dat_file = NULL, def_file = NULL, none = TRUE)
```

Arguments

`dat_file`	A string value with the path to the DAT file to import.
`def_file`	A string value with the path to the DEF file to import.
`none`	A boolean variable indicating whether NONE is included in the dat file or not - default = `TRUE`

Value

A list including elements

`dat`	A matrix of the imported dat-file
`utils_mat`	A matrix including the utilities from the dat file
`utils_list`	A list including the individual utilities from the dat file. One list element per respondent
`iaw`	A matrix including the individual awareness factors from the dat file.
`idis`	A matrix including the individual distribution factors from the dat file.
`seg`	A matrix containing the segment data
`weight`	A vector containing the weight per respondent
`def`	A list containing the values from `VP.read_def`
`pricemat_tested`	A matrix containing the prices used in the model
`pricerange_tested`	A vector containing the minimal and maximal price used in the model
`pr_range_mat`	A matrix containing the minimal and maximal prices per SKU used in the model
`SKUs`	A vector containing the SKU labels from the def-file.
`nlev`	A vector containing the number of levels per attribute.
`ID`	A vector containing the ID per respondent

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
beer_data <- get_PricerData(dat_file = "beer_study.dat",
                            def_file = "beer_study.def",
                            none = TRUE)

str(beer_data)

## End(Not run)
```

`hello` *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

`prodAcceptance` *Calculate Product Acceptance (Buying Rate)*

Description

Calculates the Productacceptance/Buying Rate for a given concept.

Usage

```
prodAcceptance(conc, utils)
```

Arguments

`conc`	A binary vector containing 0s or 1s for each level of the model.
`utils`	A matrix containing the utilities per respondent of the model. Remove the NONE column if present.

Value

A vector contaning the prodct acceptance/buying rate values of the concept for each respondent.

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
...

## End(Not run)
```

---

readCHO *Read Sawtooth CHO file*

Description

Reads the Sawtooth CHO file

Usage

```
readCHO(fileIN, progress = TRUE)
```

Arguments

fileIN	A string with the file name to be imported incl. path (if necessary)
progress	A boolean variable indicating if progress bar should be displayed - default TRUE; set to FALSE if less than 50 cases to read.

Value

A list including elements

fileIN	A string which returns the input file name
choIN	A matrix containing the raw imported cho-file
ind_info	A matrix (one line per respondent) containing the info of the first line per respondent of the Sawtooth cho-file
nconc	A list (one list element per respondent) containing vectors (length: ntasks) of the numbers of concepts per task. (Can vary per task, e.g. ACBC)
choice	A list (one list element per respondent) of vectors (length: ntasks) containing the choices per task.
design	A matrix containing the plain experimental design; no version, task or concept information included
design_out	A matrix: design to be used in e.g. writeCHO; 1st column: sequential version number; 2nd column: ID; 3rd column: task; 4th column: concept; 5th column ++: design
ID	A vector containing the IDs

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
choIN <- readCHO("example.cho")

## End(Not run)
```

`readDAT` *Read Sawtooth DAT file*

Description

Reads the Sawtooth DAT file

Usage

```
readDAT(inFILE, exportCOMPLETES = FALSE, exportUNIQUE = TRUE,
  ID_var = "r", out_unique = NULL, out_COMP = NULL, progress = TRUE)
```

Arguments

`inFILE`	A string with the file name to be imported incl. path (if necessary)
`exportCOMPLETES`	A boolean variable indicating if a dat file with the COMPLETE cases should be exported. default `FALSE`
`exportUNIQUE`	A boolean variable indicating if a dat file with the UNIQUE cases should be exported. default `TRUE`
`ID_var`	A string variable giving the variable name of the ID variable. default = "r"
`out_unique`	A string variable in case the outfile should be specifically labeled. If `NULL` label is set to fileIN_UNIQUE.dat
`out_COMP`	A string variable in case the outfile should be specifically labeled. If `NULL` label is set to fileIN_Complete.dat
`progress`	A boolean variable indicating if progress bar should be displayed - default `TRUE`; set to FALSE if less than 50 cases to read.

Value

A list including elements

`inFILE`	A string which returns the input file name
`dat_file`	A character-vector containing the raw imported dat-file
`dat_table`	A data.frame containing the information from the dat file
`dat_completes`	A character-vector containing the raw dat-file including status = "terminate" only
`dat_unique`	A character-vector containing the raw dat-file with unique and complete cases only

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
dat_file <- readDAT("example.dat")

## End(Not run)
```

`summary_calibEXE` *Summary of calibEXE*

Description

Runs a summary for the calibration of utilities using the calibEXE function.

Usage

```
summary_calibEXE(calibData = calibData, outfile = "summary.txt",
  infile = "INFILE")
```

Arguments

`calibData`	A list containing the results of `calibEXE`
`outfile`	A text value indicating the label of the output file. Default `"summary.txt"`.
`infile`	A text value indicating the label input file for `calibEXE` function. Default `"INFILE"`.

Value

No output generated. Summary will be exported to the indicated file.

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
summary_calibEXE(calibData, "summary.txt", infile = dat_input$def$file_in)

## End(Not run)
```

`VD.computeShares` *Compute ValueDriver shares*

Description

Calculates the shares (first choice or preference share) for a ValueDriver like study.

Usage

```
VD.computeShares(design, utils, nlev, weight = NULL, FC = FALSE,
  dummy = TRUE)
```

Arguments

`design`	A matrix containing the design including the concepts to simulate - in case it is not dummy coded use `dummy = TRUE` to ally dummy coding via `convertSSItoDesign`
`utils`	A matrix containing the utilities
`nlev`	A vector indicating the number of levels per attribute
`weight`	A vector with the weights (one per respondent)
`FC`	A boolean variable indicating if first choice simulation should be used (`FALSE` indicates preference share simulation) - default `TRUE`
`dummy`	A boolean variable indicating if the design file needs to be dummy-coded using `convertSSItoDesign`. - default `TRUE`

Value

A list including elements

`meanShares`	aggregated shares accross all respondents
`indShares`	individual shares for each respondent

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
utils_mat <- VMinR$VDdata$utils_mat
nlev <- VMinR$VDdata$nlev
weight <- VMinR$VDdata$weight
basecase <- as.matrix(VMinR$basecase_dummy_2)

sim_BaseR <- VD.computeShares(design = basecase,
                              utils = utils_mat,
                              nlev = nlev,
                              weight = weight,
                              FC = FALSE)

round(sim_BaseR$meanShares, 3)
```

VD.read_def *Read ValueDriver definitions file*

Description

Reads the ValueDriver def file containing the definition (e.g. labels, prices)

Usage

```
VD.read_def(file, nlev)
```

Arguments

file	A string value with the path to the DEF file to import.
nlev	A vector indicating the number of levels per attribute

Value

A list including elements

att_List	A list containing one element per attribute which contains a vector of the levels
nlev	A vector indicating the number of levels per attribute
natt	A variable returning the number of attributes
def	A list containg the labels of the segment data (variable names and levels)
nseg	A variable returning the number of segments
file_in	A string value with the path to the DEF file which was passed to the function.

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
VD.read_def(file = "data/TEST_timtim_5seg_2_gew_gew2.def",
            nlev = c(4, 6, 4, 2, 2, 2, 2, 2, 2, 2, 2, 5, 5, 5, 5, 5, 5))

## End(Not run)
```

VMinR *VM test data - timtim*

Description

Dataset containing the variable contents (data model) settings to perform input validation.

Format

A list with 3 elements.

VDdata A list containing the output of `get_DriverData`

basecase A matrix containing the scenario information in SSI style

basecase_dummy_2 A matrix containing the scenario information in dummy coding

Examples

```
data(VMinR)
str(VMinR)
```

VP.computeShares *Compute ValuePricer shares*

Description

Calculates the shares (first choice or preference share) for a ValuePricer like study.

Usage

```
VP.computeShares(utils, prices, simPrices, simSKUs = NULL, nlev,
  weight = NULL, none = FALSE, iaw = NULL, FC = FALSE)
```

Arguments

`utils`	A matrix containing the utilities
`prices`	A matrix containing the prices used in the interview
`simPrices`	A vector containing the prices to be used in the simulated scenario
`simSKUs`	A vector containing indicies of the SKUs to be used in the simulated scenario
`nlev`	An vector indicating the number of levels per attribute
`weight`	A vector with the weights (one per respondent). Will be set to 1 if `NULL`
`none`	A boolean variable indicating if NONE should be used in the simulated scenario. - default `FALSE`
`iaw`	A matrix of individual awareness factors corresponding to those in the ValuePricer tool. All are set to 1 if `NULL`. In case both individual awareness and distribution factors need to be used the respective matrices need to be multiplied before passing to this function.
`FC`	A boolean variable indicating if first choice simulation should be used (`FALSE` indicates preference share simulation) - default `TRUE`

Value

A list including elements

ind_sim	individual shares for each respondent
Xbeta	X * beta matrix used to calculate the shares (utility sums).
simPrices	respective input object passed through
simSKUs	respective input object passed through
prices	respective input object passed through
iaw	respective input object passed through
none	respective input object passed through
weight	respective input object passed through
simShares	aggregated shares accross all respondents

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
beer_def <- beer_data$def

sim_Beer <- VP.computeShares(beer_data$utils_mat,
                             beer_def$prices,
                             beer_def$prices[,3],
                             simSKUs = NULL,
                             nlev = beer_data$nlev,
                             weight = beer_data$weight,
                             none = FALSE,
                             iaw = NULL,
                             FC = FALSE)

round(sim_Beer$simShares, 3)
```

VP.read_def *Read ValuePricer definitions file*

Description

Reads the ValuePricer def file extracting the labels and prices

Usage

```
VP.read_def(file)
```

Arguments

file	A string value with the path to the DEF file to import.

Value

A list including elements

brands	A vector including the SKU labels
prices	A matrix (nSKUs x nPrices) including the prices per SKU

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
VP.read_def(file = "data/TEST_FILE.def")

## End(Not run)
```

---

writeCHO *Read Sawtooth CHO file*

Description

writes the Sawtooth CHO file; Input is based on the list elements of readCHO.

Usage

```
writeCHO(export_file = "outfile.cho", design_out, ind_info_OUT, nconc, cho,
  progress = TRUE)
```

Arguments

export_file	A string with the file name to be writen to incl. path (if necessary)
design_out	matrix/data.frame: design to be exported: 1st column: sequential version number; needs to be sequential 1 to nversions (no parts missing); 2nd column: ID; 3rd column: task; 4th column: concept; 5th column ++: design;
ind_info_OUT	A matrix (one line per respondent) containing the info of the first line per respondent for the Sawtooth cho-file
nconc	A list (one list element per respondent) of vectors (length: ntasks) containing the numbers of concepts per task. (Can vary per task, e.g. ACBC)
cho	A list (one list element per respondent) of vectors (length: ntasks) containing the choices per task.
progress	A boolean variable indicating if progress bar should be displayed - default TRUE; set to FALSE if less than 50 cases to read.

Value

No output returned. File writen to working directory.

Author(s)

Maximilian Rausch - Maximilian.Rausch@tns-infratest.com

Examples

```
## Not run:
writeCHO(export_file = "outfile.cho", choIN$design_out, choIN$ind_info_OUT, choIN$nconc, choIN$cho)

## End(Not run)
```

Index