Package 'stcm'

June 29, 2015

Title Tools for Inference with Set-Theoretic Comparative Methods

Version 0.1.1

Date 2015-06-28

Description Provides a number of functions for carrying out inference with set-theoretic comparative methods, including facilities for examining causal paths, assessing the sensitivity of results to measurement and model specification error, and performing Random Forest Comparative Analysis.

Depends R (>= 3.1.1),

License GPL (>= 3)

LazyData yes

Imports QCA, plyr, ggplot2, XML, randomForest, caret, dendextend, magrittr, stats, utils, graphics, grDevices

Maintainer Chris Krogslund <ckrogslund@berkeley.edu>

NeedsCompilation no

Repository CRAN

Author Chris Krogslund [aut, cre]

Date/Publication 2015-06-29 09:52:42

R topics documented:

`caret_infer` *QCA-style Inference with Predictive Models*

Description

`caret_infer` produces QCA-style solutions using models from the `caret` package and methods detailed in Krogslund & Michael (2014) data, case.ids, outcome, inter.func, type, clust.iter.max, clust.nstart, clust.alg, qca.style

Usage

```
caret_infer(data, case.ids, outcome, inter.func, type, clust.iter.max,
clust.nstart, clust.alg, qca.style, ...)
```

Arguments

`data`	a data frame
`case.ids`	a character vector giving the names of case identifiers
`outcome`	a character string giving the name of the outcome
`inter.func`	a function detailing how variable values should be combined to create interaction values
`type`	one of either "c" for classification prediction or "r" for regression prediction
`clust.iter.max`	the maximum number of iterations allowed in `kmeans()`
`clust.nstart`	number of random sets to be chosen in `kmeans()`
`clust.alg`	the clustering algorithm used by `kmeans()`
`qca.style`	logical; if TRUE, return QCA-style printed solutions
`...`	arguments passed to `caret::train()`

Value

Returns a named list containing the following elements:

`model`	a `caret` object of class "train"
`imp.scores`	a vector of unscaled variable importance scores
`solutions`	a vector of QCA-style sufficient conditions

Examples

```
### NOTE: Not Run

# Load data
data(hicks_29)

# QCA-style inference using a Linear Support Vector Machine
#a<-caret_infer(data = hicks_20, case.ids = "Case", outcome = "CON", type = "c", method="svmLinear")

# Examine solutions
#a$solutions

# QCA-style inference using Naive Bayes
#a<-caret_infer(data = hicks_20, case.ids = "Case", outcome = "CON", type = "c", method="nb")

# Examine solutions
#a$solutions

# QCA-style inference using Neural Networks
#a<-caret_infer(data = hicks_20, case.ids = "Case", outcome = "CON", type = "c", method="nnet")

# Examine solutions
#a$solutions
```

---

cond_expand *Create interaction expansion*

Description

cond_expand expands a dataset to include interaction values for all possible combinations of its variables.

Usage

```
cond_expand(data, case.ids, outcome, conditions, inter.func)
```

Arguments

data	a data frame
case.ids	optional character vector containing variable names that identify cases
outcome	a character string containing the name of the outcome variable
conditions	an optional character vector containing the names of the causal factors
inter.func	a function that combines the constituent values of an interaction

Value

Returns a data frame

Examples

```
# Load data
data(hicks_20)

# Combine values via "min" function
cond_expand(data = hicks_20, case.ids = "Case", outcome = "CON")

# Combine values via "mean" function
cond_expand(data = hicks_20, case.ids = "Case", outcome = "CON", inter.func = "mean")
```

---

eqmcc_args *Get* eqmcc() *arguments*

Description

eqmcc_args creates an empty list of arguments taken by eqmcc()

Usage

```
eqmcc_args()
```

Value

Returns a named list of arguments taken by eqmcc()

Examples

```
a<-eqmcc_args()
```

---

eqmcc_options *Simulations using* eqmcc()

Description

eqmcc_options repeatedly executes the eqmcc() function over a number of different argument values and returns its solutions

Usage

```
eqmcc_options(data, eqmcc.options.list, verbose)
```

Arguments

data	a data frame
eqmcc.options.list	a named list containing eqmcc() arguments and the values with which they should be simulated; list should be initially produced by a call to eqmcc_args()
verbose	logical; if TRUE, will print extra execution informaton

Value

Returns a named list containing the following elements:

`opts`	a data frame containing the parameter values used for each execution
`results`	a list of `eqmcc()` solutions

Examples

```
# Load data
data(hicks_20)

# Generate list of eqmcc arguments
arglist<-eqmcc_args()

# Specify values for execution
arglist$outcome<-"CON"
arglist$n.cut<-1:5
arglist$include<-c("1", "?", paste("1", "?", sep=","))

# Run execution
a<-eqmcc_options(data = hicks_20[,-1], eqmcc.options.list = arglist)

# Get data frame of parameter values
a$opts

# Get the solutions for parameter specifications in row 6
a$opts[6,]
a$results[[6]]

# Load data
data(hh)

# Generate list of eqmcc arguments
arglist<-eqmcc_args()

# Specify values for execution
arglist$outcome<-"success"
arglist$incl.cut1<-seq(from = 0.5, to = 1, by = 0.05)
arglist$n.cut<-1:5
arglist$include<-c("1", "?", paste("1", "?", sep=","))

# Run execution
a<-eqmcc_options(data = hh[,-1], eqmcc.options.list = arglist)

# Get data frame of parameter values
a$opts

# Get the solutions for parameter specifications in row 27
a$opts[27,]
a$results[[27]]
```

`fsqca_sim` *fsQCA Sufficiency Inclusion Score & Minimum Frequency Threshold Simulation*

Description

`fsqca_sim` returns QCA results for a range of minimum frequency thresholds across an arbitrarily large set of sufficiency inclusion scores

Usage

```
fsqca_sim(data, outcome, conditions, min.incl.cut,
max.incl.cut, min.n.cut, max.n.cut, reps, plot,
plot.legend, verbose, ...)
```

Arguments

`data`	a data frame
`outcome`	a character string or column index indicating the outcome variable
`conditions`	optional character vector or vector of column indices indicating explanatory variables
`min.incl.cut`	numeric lower bound for sampling of sufficiency inclusion scores
`max.incl.cut`	numeric upper bound for sampling of sufficiency inclusion scores
`min.n.cut`	numeric lower bound for minimum frequency thresholds
`max.n.cut`	numeric upper bound for minimum frequency thresholds
`reps`	number of sufficiency inclusion score pairs to be sampled
`plot`	logical; if TRUE, returns a plot of solutions
`plot.legend`	a character string indicating the type of legend to plot; `solutions` indicates plot legend should contain actual unique solutions; `ids` indicates plot should contain numeric identifiers for unique solutions; `none` indicates plot should not contain a legend
`verbose`	logical; if TRUE, prints additional execution information
`...`	optional arguments passed to `eqmcc()`

Value

Returns a named list with the following components:

`plot`	(if `PLOT==TRUE`Plot of results across `incl.cut` and `n.cut` parameter values
`results`	a data frame containing solutions produced for each set of sampled parameter values
`legend`	a data frame containing QCA solutions and their unique identifiers

Examples

```
# Load data
data(hh)

# Remove case indicator
hh<-hh[,-which(colnames(hh)=="Country")]

# Run function
fsqca_sim(data = hh, outcome = "success", reps=25, plot = TRUE)
```

`fsqca_sim_inclcut` *fsQCA Sufficiency Inclusion Score Simulation*

Description

`fsqca_sim_inclcut` returns QCA results for a range of minimum frequency thresholds across an arbitrarily large set of sufficiency inclusion scores

Usage

```
fsqca_sim_inclcut(data, outcome, conditions, min.incl.cut,
max.incl.cut, n.cut, reps, verbose, ...)
```

Arguments

`data`	a data frame
`outcome`	a character string or column index indicating the outcome variable
`conditions`	optional character vector or vector of column indices indicating explanatory variables
`min.incl.cut`	numeric lower bound for sampling of sufficiency inclusion scores
`max.incl.cut`	numeric upper bound for sampling of sufficiency inclusion scores
`n.cut`	minimum frequency threshold
`reps`	number of sufficiency inclusion score pairs to be sampled
`verbose`	logical; if TRUE, prints additional execution information
`...`	optional arguments passed to `eqmcc()`

Value

Returns adata frame containing solutions produced for each set of sampled parameter values

Examples

```
# Load data
data(hh)

# Remove case indicator
hh<-hh[,-which(colnames(hh)=="Country")]

# Run function
a<-fsqca_sim_inclcut(data = hh, outcome = "success", n.cut=2)

# Get table of solutions (note: truncated to first 50 rows)
a[1:50,]

# Increase the number of replications
#'a<-fsqca_sim(data = hh, outcome = "success", reps=1000, plot = TRUE)
```

`get_qca_datasets` *Get QCA datasets*

Description

`get_qca_datasets` downloads a list of QCA datasets from the COMPASSS.org database

Usage

```
get_qca_datasets(csQCA, mvQCA, fsQCA)
```

Arguments

`csQCA`	logical; if TRUE, download all available csQCA datasets
`mvQCA`	logical; if TRUE, download all available mvQCA datasets
`fsQCA`	logical; if TRUE, download all available fsQCA datasets

Value

Returns a named list of QCA datasets with author identifiters

Examples

```
# Get all csQCA datasets (not run)

# a<-get_qca_datasets(csQCA = TRUE)

# Get all csQCA and fsQCA datasets (not run)

# a<-get_qca_datasets(csQCA = TRUE, fsQCA = TRUE)
```

hh *Replication from Hussain & Howard*

Description

A dataset containing fuzzy-set membership scores for 20 countries in several sets, including average income, wealth inequality, level of unemployment, population age, digital connectivity, censorship sophistication, fuel dependence of the economy, regime fragility, and social movement success.

Usage

```
hh
```

Format

A data frame with 20 rows and 11 variables

Source

http://philhoward.org/wp-content/uploads/2012/11/International-Studies-Review-Replication-Data.csv

References

Muzammil M. Hussain and Philip N. Howard. 2013. "What Best Explains Successful Protest Cascades? ICTs and the Fuzzy Causes of the Arab Spring." International Studies Review (15) 1, pp 48-66.

hicks_20 *Replication Data from Hicks et al. (1995)*

Description

A dataset containing crisp-set membership scores for 15 countries in 1920 in several sets, including patriarchal statism, unitary democracy, working- class mobilization, liberal government, catholic government, and the consolidation of income security programs.

Usage

```
hicks_20
```

Format

A data frame with 15 rows and 7 variables

Source

http://www.compasss.org/bibliography/data/HicksEtal1995set1.csv

References

Hicks, Alexander, Joya Misra, and Tang Nah Ng. 1995. "The Programmatic Emergence of the Social Security State." American Sociological Review 60 (3):329-49.

`hicks_29` *Replication Data from Hicks et al. (1995)*

Description

A dataset containing crisp-set membership scores for 15 countries in 1929 in several sets, including patriarchal statism, unitary democracy, working- class mobilization, liberal government, catholic government, and the consolidation of income security programs.

Usage

```
hicks_29
```

Format

A data frame with 15 rows and 7 variables

Source

http://www.compasss.org/bibliography/data/HicksEtal1995set2.csv

References

Hicks, Alexander, Joya Misra, and Tang Nah Ng. 1995. "The Programmatic Emergence of the Social Security State." American Sociological Review 60 (3):329-49.

`meas_error` *Add measurement error to datasets*

Description

`meas_error` creates a list of datasets that contain a specified type and level of measurement error

Usage

```
meas_error(data, case.ids, outcome, conditions, type,
 error.level, fuzzy.range, reps, analyze, max.cond.plot, ...)
```

Arguments

data	a data frame
case.ids	optional character vector containing variable names that identify cases
outcome	a character string containing the name of the outcome variable
conditions	optional character vector indicating explanatory variables
type	a character string indicating the type of set membership scores in the data; takes either "crisp" for crisp-set data, or "fuzzy" for fuzzy-set data
error.level	a numeric vector indicating the proportion of data points that should be perturbed with every iteration
fuzzy.range	bandwidth around a data point used for sampling fuzzy-set membership scores from a uniform distribution
reps	number of perturbed datasets to create
analyze	logical; if TRUE, carry out frequency analysis of conditions
max.cond.plot	maximum number of conditions to plot for frequency analysis
...	optional arguments passed to eqmcc()

Value

Returns a named list containing the following components:

parameter.specs	a data frame containing parameter specifications used to generate each subset of the data frames list
datasets	a list of data frames containing measurement error
result.freq	a data frame containing counts of condition appearances
plot	a plot of condition frequencies (per solution)

Examples

```
# Load data
data(hicks_29)

# Run function, get measurement error in 1, 5, and 10% of the data points
a<-meas_error(data = hicks_29, case.ids = "Case", outcome = "CON", type = "crisp",
error.level = c(0.01, 0.05, 0.1), reps = 200, analyze=FALSE)

# Get parameter specifications
a$parameter.specs

# Isolate list of datasets with 5% measurement error
a$datasets[[2]]

# Load data
data(hh)

# Run function, get measurement error in 1, 5, and 10% of the data points
```

```
a<-meas_error(data = hh, case.ids = "Country", outcome = "success", type = "fuzzy",
error.level = c(0.01, 0.05, 0.1), reps = 200, analyze=FALSE)

# Get parameter specifications
a$parameter.specs

# Isolate list of datasets with 10% measurement error
a$datasets[[3]]
```

`mod_error` *Add model error to datasets*

Description

`mod_error` creates a list of datasets that contain a specified type and level of measurement error

Usage

```
mod_error(data, case.ids, outcome, conditions, type,
number, reps, analyze, max.cond.plot, ...)
```

Arguments

`data`	a data frame
`case.ids`	optional character vector containing variable names that identify cases
`outcome`	a character string containing the name of the outcome variable
`conditions`	optional character vector indicating explanatory variables
`type`	a character string indicating the type of set membership scores in the data; takes either "crisp" for crisp-set data, or "fuzzy" for fuzzy-set data
`number`	a numeric vector indicating the number of random variables to be added to the data
`reps`	number of perturbed datasets to create
`analyze`	logical; if TRUE, carry out frequency analysis of conditions
`max.cond.plot`	maximum number of conditions to plot for frequency analysis
`...`	optional arguments passed to `eqmcc()`

Value

Returns a named list containing the following components:

`datasets`	a list of data frames containing model specification error
`result.freq`	a data frame containing counts of condition appearances
`plot`	a plot of condition frequencies (per solution)

Examples

```
# Load data
data(hicks_29)

# Run function, get three random variables
a<-mod_error(data = hicks_29, case.ids="Case", outcome="CON", type = "crisp",
number = 3, reps = 200, analyze=FALSE)

# Load data
data(hh)

# Run function, get two random variables
a<-mod_error(data = hh, case.ids="Country", outcome="success", type = "fuzzy",
number = 2, reps = 200, analyze=FALSE)
```

---

paths_summary *Summarize an STCM dataset*

Description

paths_summary provides descriptions of an STCM dataset related to cases and causal paths.

Usage

```
paths_summary(data, case.ids, outcome, conditions, plot, verbose,
size.warn)
```

Arguments

data	a data frame
case.ids	optional character vector containing variable names that identify cases
outcome	a character string containing the name of the outcome variable
conditions	an optional character vector containing the names of the causal factors
plot	logical; if TRUE, a plot of causal pathways is returned
verbose	logical; if TRUE, information on code execution will be printed
size.warn	logical; if TRUE, prompts the user to authorize execution when number of causal paths >=1000

Value

Returns a named list with the following components:

n.cases	number of cases in the dataset
n.cases.table	a data frame containing the number of unique cases for each variable in case.ids
n.paths.possible	number of possible paths to the outcome

`n.paths.observed`
: number of observed unique paths to the outcome

`n.paths.percent`
: percent of possible paths to the outcome observed

`paths.obs`
: a data frame containing the observed unique paths to the outcome

`paths.unobs`
: a data frame containing the observed unique paths to the outcome

`freq.paths.obs`
: a data frame containing the observed unique paths to the outcome, the number of cases corresponding to those paths, and their `case.id` values

`var.diversity`
: a data frame containing a measure of diversity in the observed values for each variable; ranges from 0 (observed values are either all 0 or all 1) to 1 (observed values are equally divided between 0s and 1s)

`paths.plot`
: if `plot==TRUE`, a path diagram; solid lines indicate observed paths, while dotted lines indicate unobserved paths; the outcome values associated with each path are also given

`multicollinearity.df`
: a data frame bivariate correlations and their absolute values for all causal factors

`multicollinearity.avg`
: the average absolute bivariate correlation

Examples

```
# Load data
data(hicks_20)

# Run function
a<-paths_summary(data = hicks_20, case.ids = "Case", outcome = "CON", plot = TRUE)

# Get number of cases
a$n.cases

# Get number of paths to the outcome observed
a$n.paths.observed

# Get percent of possible paths observed
a$n.paths.percent

# Get multicollinearity
a$multicollinearity.df.avg

# View paths plot
a$paths.plot
```

`paths_summary_fuzzy` *Summarize a fuzzy STCM dataset*

Description

`paths_summary_fuzzy` provides descriptions of an STCM dataset related to cases and causal paths.

Usage

```
paths_summary_fuzzy(data, case.ids, outcome, conditions,
 inter.func, sig.digits, size.warn)
```

Arguments

data	a data frame
case.ids	optional character vector containing variable names that identify cases
outcome	a character string containing the name of the outcome variable
conditions	an optional character vector containing the names of the causal factors
inter.func	a function that combines the constituent values of an interaction
sig.digits	number of significant digits for cutpoints
size.warn	logical; if TRUE, prompts the user to authorize execution when number of causal paths >=1000

Value

Returns a named list with the following components:

n.fuzzy.paths.obs
: a data frame containing the number and percent of possible paths observed for a given binary cutpoint

fuzzy.paths.plot
: a plot of the percent of possible paths observed across a range of binary cutpoints

Examples

```
# Load data
data(hh)

# Run function
a<-paths_summary_fuzzy(data = hh, case.ids = "Country", outcome = "success", sig.digits = 1)

# Get results
a$n.fuzzy.paths.obs

# Get plot
a$fuzzy.paths.plot
```

rfca *Carry out RFCA*

Description

rfca carries out Random Forest Comparative Analysis (RFCA) as detailed in Krogslund & Michael (2014)

Usage

```
rfca(data, outcome, case.ids, fuzzy, ntree, mtry, inter.func,
clust.iter.max, clust.nstart, clust.alg, qca.style, ...)
```

Arguments

data	a data frame
outcome	a character string giving the name of the outcome
case.ids	a character vector giving the names of case identifiers
fuzzy	logical; if TRUE, data will be dichotomized around the median
ntree	the number of trees to be grown
mtry	the number of variables to try at each split
inter.func	a function detailing how variable values should be combined to create interaction values
clust.iter.max	the maximum number of iterations allowed in stats::kmeans()
clust.nstart	number of random sets to be chosen in stats::kmeans()
clust.alg	the clustering algorithm used by stats::kmeans()
qca.style	logical; if TRUE, return QCA-style printed solutions
...	arguments passed to randomForest()

Value

Returns a vector of variable names

Examples

```
# Load data
data(hicks_29)

# Run RFCA
a<-rfca(data = hicks_29, outcome = "CON", case.ids = "Case", ntree = 100, mtry = 5)

# Increase the number of trees grown and variables tried at each split
a<-rfca(data = hicks_29, outcome = "CON", case.ids = "Case", ntree = 100, mtry = 5)

# Load data
```

```
data(hh)

# Run RFCA
a<-rfca(data = hh, outcome = "success", case.ids = "Country", fuzzy = TRUE,
ntree = 100, mtry = 5)
```

`rfca_args` *Get* `rfca()` *arguments*

Description

`rfca_args` creates an empty list of arguments taken by `rfca()`

Usage

```
rfca_args()
```

Value

Returns a named list of arguments taken by `rfca()`

Examples

```
a<-rfca_args()
```

`rfca_options` *Simulations using* `rfca()`

Description

`rfca_options` repeatedly executes the `rfca()` function over a number of different argument values and returns its solutions

Usage

```
rfca_options(data, rfca.options.list, verbose)
```

Arguments

`data`	a data frame
`rfca.options.list`	a named list containing `rfca()` arguments and the values with which they should be simulated; list should be initially produced by a call to `rfca_args()`
`verbose`	logical; if TRUE, will print extra execution informaton

Value

Returns a named list containing the following elements:

`opts`	a data frame containing the parameter values used for each execution
`results`	a list of `rfca()` solutions

Examples

```
# Load data
data(hicks_20)

# Generate list of eqmcc arguments
arglist<-rfca_args()

# Specify values for execution
arglist$outcome<-"CON"
arglist$case.ids<-"Case"
arglist$ntree<-c(1,5,100,5000)
arglist$mtry<-c(1,5,10,20)

# Run execution
a<-rfca_options(data = hicks_20, rfca.options.list = arglist)

# Get data frame of parameter values
a$opts

# Get the solutions for parameter specifications in row 6
a$opts[4,]
a$results[[4]]
```

Index