Package ‘cherry’

October 12, 2022

Version 0.6-14

Date 2021-05-07

Title Multiple Testing Methods for Exploratory Research

Author Jelle Goeman, Aldo Solari, Rosa Meijer

Maintainer Jelle Goeman <`j.j.goeman@lumc.nl`>

Depends methods

Imports bitops, lpSolve, hommel

Suggests MASS, multtest

Description Provides an alternative approach to multiple testing by calculating a simultaneous upper confidence bounds for the number of true null hypotheses among any subset of the hypotheses of interest, using the methods of Goeman and Solari (2011) <doi:10.1214/11-STS356>.

License GPL (>= 2)

Collate closed.R shortcuts.R Meinshausen.R regionmethod.R DAGmethod.R DAGstructure.R structuredHolm.R

LazyLoad yes

Repository CRAN

NeedsCompilation no

Date/Publication 2021-05-07 15:50:05 UTC

R topics documented:

`Adjusted` *Adjusted p-values for the number of true hypotheses.*

Description

Calculates adjusted p-values for the number of true hypotheses on the basis of the closed testing procedure.

Usage

```
adjusted (closure, reject, n=0)
```

Arguments

`closure`	An object of class `closure`, typically created through a call to `closed` .
`reject`	A character vector identifying the hypotheses to be rejected. Must be a subvector of `hypotheses(closure)`.
`n`	The maximum number of false null hypotheses allowed.

Details

The function `pick` calculates adjusted p-values for intersection hypotheses of interest.

Value

The function returns a p-value (numeric).

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

Examples

```
# Example: the birthwt data set from the MASS library
# We want to find variables associated with low birth weight
library(MASS)
fullfit <- glm(low~age+lwt+race+smoke+ptl+ht+ui+ftv, family=binomial, data=birthwt)
hypotheses <- c("age", "lwt", "race", "smoke", "ptl", "ht", "ui", "ftv")

# Define the local test to be used in the closed testing procedure
mytest <- function(hyps) {
  others <- setdiff(hypotheses, hyps)
  form <- formula(paste(c("low~",  paste(c("1", others), collapse="+"))))
  anov <- anova(glm(form, data=birthwt, family=binomial), fullfit, test="Chisq")
  res <- anov$"Pr("[2]                   # for R >= 2.14.0
  if (is.null(res)) res <- anov$"P("[2]   # earlier versions
  res
}

# Perform the closed testing with ajdusted p-values
cl <- closed(mytest, hypotheses, alpha=NA)

# What is the adjusted p-value of the intersection of the following hypotheses?
adjusted(cl, c("ht", "lwt", "smoke", "ui"))

# From what confidence level would we conclude
# that more than 2 of the following hypotheses would be false?
adjusted(cl, c("ht", "lwt", "smoke", "ui"), n=2)
```

`Closed Testing` *Closed Testing for Exploratory Research*

Description

Performs the closed testing procedure for user-specified local test.

Usage

```
closed (test, hypotheses, alpha = 0.05, adjust=FALSE)
```

Arguments

`test`	A function that performs the local test. The function should accept a subvector of the `hypotheses` argument as input, and return a p-value.
`hypotheses`	Identifiers of the collection of elementary hypotheses.
`alpha`	The significance level of the test procedure. If set to `NA`, the function calculates adjusted p-values for each hypothesis in the closure.
`adjust`	Whether adjusted p-values should be calculated.

Details

The function `closed` performs the closed testing procedure on the collection `hypotheses`, testing all their intersection hypotheses and controlling the familywise error rate.

Value

The function `closed` returns an object of class `closure`.

Note

The number of intersection hypotheses is exponential in the number of elementary hypotheses. The number of elementary hypotheses is therefore limited to `log2(.Machine$integer.max+1)` (typically 31) for computational reasons.

It is possible to set both `adjust` to `TRUE` and specify `alpha`. In that case, adjusted p-values are calculated up to a value `alpha`; all higher p-values are set to 1.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

References

Goeman and Solari (2011) Statistical Science 26 (4) 584-597.

Examples

```
# Example: the birthwt data set from the MASS library
# We want to find variables associated with low birth weight
library(MASS)
fullfit <- glm(low~age+lwt+race+smoke+ptl+ht+ui+ftv, family=binomial, data=birthwt)
hypotheses <- c("age", "lwt", "race", "smoke", "ptl", "ht", "ui", "ftv")

# Define the local test to be used in the closed testing procedure
mytest <- function(hyps) {
  others <- setdiff(hypotheses, hyps)
  form <- formula(paste(c("low~",  paste(c("1", others), collapse="+"))))
  anov <- anova(glm(form, data=birthwt, family=binomial), fullfit, test="Chisq")
  res <- anov$"Pr("[2]                    # for R >= 2.14.0
  if (is.null(res)) res <- anov$"P("[2]   # earlier versions
  res
}

# perform the closed testing
cl <- closed(mytest, hypotheses)
cl

# how many variables among a chosen set are associated with the response?
pick(cl, c("ht", "lwt", "smoke", "ui"))

# adjusted p-values and a confidence distribution
cl <- closed(mytest, hypotheses, alpha=NA)
```

```
pick(cl, c("ht", "lwt", "smoke", "ui"))
```

`Closure class` *Class "closure" for storing the result of the closed testing procedure*

Description

The class closure is the output of a call to `closed`. It stores the information needed to calculate confidence sets for the number of true and/or false hypotheses among a selected set.

Slots

These slots are not meant to be directly accessed by the user.

`adjusted`**:** Object of class "numeric". Sores the adjusted p-values.

`defining`**:** Object of class "integer". Stores the identifiers of the intersection hypotheses that are the defining rejections of the closed testing procedure. The identifiers should be read as binary booleans, i.e. 19 = 10011 (binary) is the intersection of the first, second and fifth elementary hypothesis.

`hypotheses`**:** Object of class "character". Holds the names of the elementary hypotheses.

`alpha`**:** Object of class "numeric". The type I error level chosen.

`max.alpha`**:** Object of class "numeric". The largest value for which adjusted p-values have been calculated.

Methods

show (closure): Prints a brief description of the test results, including the upper bound of the number of true hypotheses and the corresponding lower bound to the number of false hypotheses among the full set.

summary (closure): Prints the test results (as `show`) plus the defining rejections.

defining (closure): Extracts the defining rejections as a list.

shortlist (closure): Extracts the shortlist as a list.

hypotheses (closure): Extracts the hypotheses slot.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

See Also

`closed`, `pick`.

Examples

```
# Example: the birthwt data set from the MASS library
# We want to find variables associated with low birth weight
library(MASS)
fullfit <- glm(low~age+lwt+race+smoke+ptl+ht+ui+ftv, family=binomial, data=birthwt)
hypotheses <- c("age", "lwt", "race", "smoke", "ptl", "ht", "ui", "ftv")

# Define the local test to be used in the closed testing procedure
mytest <- function(hyps) {
  others <- setdiff(hypotheses, hyps)
  form <- formula(paste(c("low~",  paste(c("1", others), collapse="+"))))
  anov <- anova(glm(form, data=birthwt, family=binomial), fullfit, test="Chisq")
  res <- anov$"Pr("[2]                    # for R >= 2.14.0
  if (is.null(res)) res <- anov$"P("[2]   # earlier versions
  res
}

# perform the closed testing procedure
cl <- closed(mytest, hypotheses)
summary(cl)
defining(cl)
shortlist(cl)
hypotheses(cl)

# how many variables among a chosen set are associated with the response?
pick(cl, c("ht", "lwt", "smoke", "ui"))
```

`Construct the DAG from a given collection of sets.`

Constructing a DAG.

Description

Constructs a DAG from a given collection of sets. Filters duplicates, and checks whether the eventual DAG structure has twoway logical relationships.

Usage

```
construct (sets)
```

Arguments

`sets`	A list of sets/hypotheses for which you want to construct a DAG according to the underlying subset relations.

Value

The function `construct` returns an object of class `DAGstructure`.

Author(s)

Rosa Meijer: `<r.j.meijer@lumc.nl>`

See Also

`DAGstructure`

Examples

```
#Generate data, where the response Y is associated with two (out of 4) covariates
set.seed(1)
n=100
p=4
X <- matrix(rnorm(n*p),n,p)
beta <- c(0,0.5,0.5,0)
Y <- X %*% beta + rnorm(n)

# Let us assume we have the following sets that we want to test:
sets <- list(c(1,2,3,4), c(1,2), c(2,3,4), c(2,3), 1, 2, 3, 4)
names(sets) <- c(1234, 12, 234, 23, 1, 2, 3, 4)

# Start by making the corresponding graph structure
struct <- construct(sets)

# Check whether the DAG has toway logical relations:
istwoway(struct)
```

`DAG class` *Class "DAG" for storing the result of the DAG multiple testing method*

Description

The class DAG is the output of a call to `DAGmethod`. It stores which hypotheses have been rejected and can be used to calculate confidence sets for the number of true and/or false hypotheses among a selected set of hypotheses.

Slots

These slots are not meant to be directly accessed by the user.

`sets`: Object of class "list". Stores unique original sets that are to be tested.

`method`: Object of class "character". Stores whether the any-parent, all-parents or structuredHolm method has been used.

`isadjusted`: Object of class "logical". Stores whether adjusted p-values are calculated.

`allpvalues`: Object of class "numeric". Stores (adjusted) p-values for all hypotheses. Has value NA if adjusted p-value is larger than alpha.

- `implications`**:** Object of class "logical". Stores whether hypotheses are implications at chosen alpha-level
- `alpha`**:** Object of class "numeric". The type I error level chosen.
- `rejected`**:** Object of class "logical". Stores for each hypothesis wether this hypothesis has been rejected
- `leaf_based_sets`**:** Object of class "list". Stores sets expressed in the indices of their corresponding leaf nodes (which are sets itself).
- `twoway`**:** Object of class "logical". Is TRUE if the final DAG structure has twoway logical relationships.

Methods

- **show** (DAG): Prints how many hypotheses there are in total and how many of them were rejected.
- **summary** (DAG): Prints the test results (as `show`).
- **alpha** (DAG): Retrieves the maximal alpha_value from the DAG object.
- **implications** (DAG): Retrieves the implications from a given DAG object.
- **pvalue** (DAG,indicator): Retrieves pvalues for all possible hypotheses (as specified by indicator) from the DAG object.

Author(s)

Rosa Meijer: `<r.j.meijer@lumc.nl>`

See Also

`DAGmethod`, `DAGpick`, `structuredHolm`.

Examples

```
#Generate data, where the response Y is associated with two (out of 4) covariates
set.seed(1)
n=100
p=4
X <- matrix(rnorm(n*p),n,p)
beta <- c(0,0.5,0.5,0)
Y <- X %*% beta + rnorm(n)

# Let us assume we have the following sets that we want to test:
sets <- list(c(1,2,3,4), c(1,2), c(2,3,4), c(2,3), 1, 2, 3, 4)
names(sets) <- c(1234, 12, 234, 23, 1, 2, 3, 4)

# Start by making the corresponding graph structure
struct <- construct(sets)

# Check whether the DAG has toway logical relations:
istwoway(struct)

# Define the local test to be used in the closed testing procedure.
```

```
# This test expects a set as input.
mytest <- function(set)
{
  X <- X[,set,drop=FALSE]
  lm.out <- lm(Y ~ X)
  x <- summary(lm.out)
  return(pf(x$fstatistic[1],x$fstatistic[2],x$fstatistic[3],lower.tail=FALSE))
}

# Perform the DAG procedure (default is all-parents method).
DAG <- DAGmethod(struct, mytest, isadjusted=TRUE)
summary(DAG)

# What are the smallest sets that are found to be significant? If the sets have names,
# as in our example, the implications function will return the names
# of the implying sets, together with their (adjusted) p-value.
# If no names are provided, indices will be used instead of the names.
implications(DAG)

# What is the adjusted p-value of the null-hypothesis corresponding to the fourth set,
# which is set c(2,3)?
# To look up the pvalue, the function uses the index or name of the set
# in the list of sets stored in the DAGstructure.
# (Note that, if there were duplicate sets in the original list, this index can be different from
# the one in the original list given to \code{construct})
pvalue(DAG,4)
pvalue(DAG, "23") #as above, but while using names

# How many of the elementary hypotheses (the last 4 sets) have to be false
# with probability 1-alpha?
# Sets (don't have to be elementary hypotheses in general) must be specified
# by their index or name.
DAGpick(DAG, 5:8)
DAGpick(DAG, c("1","2","3","4")) #as above, but while using names
```

`DAG Testing` *Testing of hypotheses, forming a DAG.*

Description

Tests all hypotheses in a given DAG while controlling the FWER, using a user-specified local test.

Usage

```
DAGmethod (DAGstructure, test, alpha_max = 0.05, method = "all", isadjusted = FALSE,
      optimization = "none", degree = "group", pvalues = NULL, verbose = FALSE)
```

Arguments

`DAGstructure`	DAGstructure object, as returned by the function `construct`.
`test`	A function that performs the local test. The function should have a set as input and return a p-value.
`alpha_max`	The significance level of the test procedure.
`method`	Type of DAG procedure that is chosen. "all" gives the all-parents procedure, "any" the any-parent procedure.
`isadjusted`	If set to TRUE, adjusted p-values will be calculated. Otherwise, the p-values of all rejected hypotheses will equal alpha_max.
`optimization`	Can be, in ascending order of accuracy and computational costs: "none", "LP" (linear programming) or "ILP" (integer linear programming).
`degree`	Can be "group" or "individual". If "group" is chosen, optimization is done on a group level, otherwise on an individual level (more accurate, but more time-consuming).
`pvalues`	Optional (in case of stored p-values): a vector in which the raw p-values of the exact sets as found in the DAGstructure argument are stored (in the same order). If the test function is provided, this argument is not necessary.
`verbose`	If set to TRUE, while running the method, a counter will indicate how many hypotheses are already rejected.

Details

The function `DAGmethod` tests all possible hypotheses within a given DAG structure, while controlling the familywise error rate.

Value

The function `DAGmethod` returns an object of class `DAG`.

Author(s)

Rosa Meijer: `<r.j.meijer@lumc.nl>`

References

Meijer and Goeman (2015) Biometrical Journal 57 (1) 123-143.

See Also

`DAG`, `DAGstructure`, `construct`, `DAGpick`.

Examples

```
#Generate data, where the response Y is associated with two (out of 4) covariates
set.seed(1)
n=100
```

```
p=4
X <- matrix(rnorm(n*p),n,p)
beta <- c(0,0.5,0.5,0)
Y <- X %*% beta + rnorm(n)

# Let us assume we have the following sets that we want to test:
sets <- list(c(1,2,3,4), c(1,2), c(2,3,4), c(2,3), 1, 2, 3, 4)
names(sets) <- c(1234, 12, 234, 23, 1, 2, 3, 4)

# Start by making the corresponding graph structure
struct <- construct(sets)

# Check whether the DAG has toway logical relations:
istwoway(struct)

# Define the local test to be used in the closed testing procedure.
# This test expects a set as input.
mytest <- function(set)
{
  X <- X[,set,drop=FALSE]
  lm.out <- lm(Y ~ X)
  x <- summary(lm.out)
  return(pf(x$fstatistic[1],x$fstatistic[2],x$fstatistic[3],lower.tail=FALSE))
}

# Perform the DAG procedure (default is all-parents method).
DAG <- DAGmethod(struct, mytest, isadjusted=TRUE)
summary(DAG)

# What are the smallest sets that are found to be significant?
implications(DAG)

# What is the adjusted p-value of the null-hypothesis corresponding to the fourth set,
# which is set c(2,3)?
# To look up the pvalue, the function uses the index or name of the set
# in the list of sets stored in the DAGstructure.
# (Note that, if there were duplicate sets in the original list, this index can be different from
# the one in the original list given to \code{construct})
pvalue(DAG,4)
pvalue(DAG, "23") #as above, but while using names

# How many of the elementary hypotheses (the last 4 sets) have to be false
# with probability 1-alpha?
# Sets (don't have to be elementary hypotheses in general) must be specified
# by their index or name.
DAGpick(DAG, 5:8)
DAGpick(DAG, c("1","2","3","4")) #as above, but while using names
```

DAGpick *Confidence limits for the number of false hypotheses in a given family of sets.*

Description

Calculates confidence limits for the number of false hypotheses on the basis of the DAG procedure within a family of sets.

Usage

```
DAGpick (DAG, indicators, optimization = "ILP")
```

Arguments

`DAG`	DAG object, as returned by the function `DAGmethod`.
`indicators`	The names or indices of the sets (as specified in the DAGstructure object) for which you want to know the confidence limits. Note that, if there were duplicate sets in the original list, the index can be different from the one in the original list given to `construct`.
`optimization`	Can be, in ascending order of accuracy and computational costs: "LP" (linear programming) or "ILP" (integer linear programming).

Value

The function `DAGpick` returns the lower bound of a 1-alpha confidence set for the number of false sets. For the moment, the function can only be used on DAG objects that correspond to a DAG with two-way logical relationships.

Author(s)

Rosa Meijer: `<r.j.meijer@lumc.nl>`

See Also

`DAGmethod`, `DAG`.

Examples

```
#Generate data, where the response Y is associated with two (out of 4) covariates
set.seed(1)
n=100
p=4
X <- matrix(rnorm(n*p),n,p)
beta <- c(0,0.5,0.5,0)
Y <- X %*% beta + rnorm(n)

# Let us assume we have the following sets that we want to test:
```

```
sets <- list(c(1,2,3,4), c(1,2), c(2,3,4), c(2,3), 1, 2, 3, 4)
names(sets) <- c(1234, 12, 234, 23, 1, 2, 3, 4)

# Start by making the corresponding graph structure
struct <- construct(sets)

# Define the local test to be used in the closed testing procedure.
# This test expects a set as input.
mytest <- function(set)
{
  X <- X[,set,drop=FALSE]
  lm.out <- lm(Y ~ X)
  x <- summary(lm.out)
  return(pf(x$fstatistic[1],x$fstatistic[2],x$fstatistic[3],lower.tail=FALSE))
}

# Perform the DAG procedure (default is all-parents method).
DAG <- DAGmethod(struct, mytest, isadjusted=TRUE)
summary(DAG)

# How many of the elementary hypotheses (the last 4 sets) have to be false
# with probability 1-alpha?
# Sets (don't have to be elementary hypotheses in general) must be specified
# by their index or name.
DAGpick(DAG, 5:8)
DAGpick(DAG, c("1","2","3","4")) #as above, but while using names
```

`DAGstructure class` *Class "DAGstructure" for storing the result of the construct method that constructs the DAG.*

Description

The class DAGstructure is the output of a call to `construct`. It stores the DAG structure that is induced by the given sets.

Slots

These slots are not meant to be directly accessed by the user.

`parents`**:** Object of class "list". Stores the parents of each set, indicated by indices.

`children`**:** Object of class "list". Stores the children of each set, indicated by indices.

`sets`**:** Object of class "list". Stores the sets that are used for the DAG construction.

`twoway`**:** Object of class "logical". Is TRUE if the final DAG structure has twoway logical relationships.

Methods

istwoway (DAGstructure): Indicates whether given DAGstructure has twoway relationships.

Author(s)

Rosa Meijer: <`r.j.meijer@lumc.nl`>

See Also

`DAGmethod`

Examples

```
# Let us assume we have the following sets that we want to test:
sets <- list(c(1,2,3,4), c(1,2), c(2,3,4), c(2,3), 1, 2, 3, 4)
# The sets need to have names in order to be able to look up their p-values later
names(sets) <- c(1234, 12, 234, 23, 1, 2, 3, 4)

# Start by making the corresponding graph structure
struct <- construct(sets)

# Check whether the DAG has toway logical relations:
istwoway(struct)
```

`hommelFast` *Calculates adjusted p-values of Hommel's method efficiently.*

Description

Calculates adjusted p-values of Hommel's method efficiently.

Usage

```
hommelFast (pvalues, simes = TRUE)
```

Arguments

`pvalues`	A vector of p-values.
`simes`	If `TRUE`, a Simes' test is used, if `FALSE` Hommel's test is used.

Value

Returns a `hommel` object.

Author(s)

Rosa Meijer: <`r.j.meijer@lumc.nl`>

Examples

```
#Generate a vector of pvalues
set.seed(1)
n <- 1000
pvalues <- c(runif(0.50*n,0,0.001), runif(0.50*n,0,1))

#Create an hommel object in which the adjusted p-values are stored, based on a Simes'test
#(or Hommel's test, by choosing simes = FALSE):
hom <- hommelFast(pvalues, simes = TRUE)

#Retrieve the first 10 adjusted p-values by using the \code{p.adjust} method
# from the hommel package. Note that they are not sorted
hommel::p.adjust(hom)[1:10]
```

`NAEP` *National Assessment of Educational Progress (NAEP) p-values*

Description

P-values for the null hypothesis of no change in the average eighth-grade mathematics achievement scores for the 34 states that participated in both the 1990 and the 1992 National Assessment of Educational Progress (NAEP) Trial State Assessment.

Usage

```
data(NAEP)
```

Format

A named numeric vector of p-values, uncorrected for multiple testing.

References

Benjamini Y. and Hochberg, Y. 2000. On the adaptive control of the false discovery rate in multiple testing with independent statistics. Journal of Educational and Behavioral Statistics 25 (1) 60-83.

Williams, V.S.L., Jones, L.V. and Tukey, J.W. 1999. Controlling error in multiple comparisons, with examples from state-to-state differences in educational achievement. Journal of Educational and Behavioral Statistics 24 (1) 42-69.

Pick *Confidence limits for the number of true hypotheses.*

Description

Calculates confidence limits for the number of true hypotheses on the basis of the closed testing procedure.

Usage

```
pick (closure, reject, alpha, silent=FALSE, plot=FALSE)
```

Arguments

`closure`	An object of class `closure`, typically created through a call to `closed` .
`reject`	A character vector identifying the hypotheses to be rejected. Must be a subvector of `hypotheses(closure)`.
`alpha`	For closure objects with adjusted p-values, specifies the value of alpha for which confidence limits are to be calculated (optional).
`silent`	If `FALSE`, prints the result to the screen.
`plot`	Whether a a confidence distribution should be plotted. Only available for closure objects with adjusted p-values.

Details

The function `pick` calculates a confidence interval for the number of true hypotheses among a selected set of hypotheses.

Value

The function returns the upper confidence limit for the number of true hypotheses among the set `reject`. The lower confidence limit is always equal to 0. If `closed` was called with `alpha=NA`, a confidence distribution is plotted and returned.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

Examples

```
# Example: the birthwt data set from the MASS library
# We want to find variables associated with low birth weight
library(MASS)
fullfit <- glm(low~age+lwt+race+smoke+ptl+ht+ui+ftv, family=binomial, data=birthwt)
hypotheses <- c("age", "lwt", "race", "smoke", "ptl", "ht", "ui", "ftv")

# Define the local test to be used in the closed testing procedure
```

```
mytest <- function(hyps) {
  others <- setdiff(hypotheses, hyps)
  form <- formula(paste(c("low~",  paste(c("1", others), collapse="+"))))
  anov <- anova(glm(form, data=birthwt, family=binomial), fullfit, test="Chisq")
  res <- anov$"Pr("[2]                    # for R >= 2.14.0
  if (is.null(res)) res <- anov$"P("[2]   # earlier versions
  res
}

# perform the closed testing
cl <- closed(mytest, hypotheses)
summary(cl)

# how many variables among a chosen set are associated with the response?
pick(cl, c("ht", "lwt", "smoke", "ui"))
```

`Plot region objects` *Visualizing of the region hypotheses that could be rejected.*

Description

Visualizes region objects as created through a call to `regionmethod`.

Usage

```
regionplot (region, alpha, color="red")

regionplot2 (region, alpha, color_rej="red", color_unrej="grey")
```

Arguments

`region`	An object of class `region`, typically created through a call to `regionmethod`.
`alpha`	For region objects with adjusted p-values, specifies the value of alpha for which rejections should be plotted (optional).
`color`	Color that is used to indicate rejected region hypotheses.
`color_rej`	Color that is used to indicate rejected region hypotheses.
`color_unrej`	Color that is used to indicate unrejected region hypotheses.

Details

Both plot functions create a graph that visualizes all possible region hypotheses. Each region hypothesis is a node in the graph, and from each region hypothesis two edged connect the hypothesis with its child hypotheses. The `regionplot2` function visualized the graph with its nodes and edges. This function is especially useful for `region` objects with a limited number of elementary hypotheses. The `regionplot` function does not display the nodes and edges separately, but draws a polygon that follows the original graph structure.

Author(s)

Rosa Meijer: <r.j.meijer@lumc.nl>

Examples

```
#generate data, where the response Y is associated with certain groups of covariates
#namely cov 3-6, 9-12, 15-18
set.seed(1)
n=100
p=20
X <- matrix(rnorm(n*p),n,p)
beta <- c(rep(0,2),rep(1,4),rep(0,2),rep(1,4),rep(0,2),rep(1,4),rep(0,2))
Y <- X %*% beta + rnorm(n)

# Define the local test to be used in the closed testing procedure
mytest <- function(left,right)
{
  X <- X[,(left:right),drop=FALSE]
  lm.out <- lm(Y ~ X)
  x <- summary(lm.out)
  return(pf(x$fstatistic[1],x$fstatistic[2],x$fstatistic[3],lower.tail=FALSE))
}

# perform the region procedure
reg <- regionmethod(rep(1,p), mytest, isadjusted=TRUE)
summary(reg)

#what are the smallest regions that are found to be significant?
implications(reg)

#how many covariates within the full region of length 20 are at least associated with the response?
regionpick(reg, list(c(1,p)), alpha=0.05)

#visualize the results by either plotting a polygon corresponding to the underlying graph
regionplot(reg)

#or by plotting the graph itself
regionplot2(reg)
```

`Region class` *Class "region" for storing the result of the region procedure*

Description

The class region is the output of a call to `regionmethod`. It stores which region hypotheses have been rejected and can be used to calculate confidence sets for the number of true and/or false hypotheses among a selected region.

Slots

These slots are not meant to be directly accessed by the user.

`weights`**:** Object of class "numeric". Stores the weights of elementary hypotheses.

`isadjusted`**:** Object of class "logical". Stores whether adjusted p-values are calculated.

`allpvalues`**:** Object of class "matrix". Stores (adjusted) p-values for all possible region hypotheses. Has value NA if adjusted p-value is larger than alpha. Corresponds to a 0x0 matrix if `all_pvalues` is set to FALSE (by default).

`implications`**:** Object of class "matrix". Stores implications includinge (adjusted) p-values at chosen alpha-level

`alpha`**:** Object of class "numeric". The type I error level chosen.

`totalrejected`**:** Object of class "numeric". Stores the total number of rejected region hypotheses.

Methods

show (region): Prints how many region hypotheses were tested and how many of them were rejected.

summary (region): Prints the test results (as `show`).

alpha (region): Retrieves the maximal alpha_value from the region object.

implications (region,alpha): Retrieves the implications from a given region object. By default, the alpha-level is set to `alpha_max`, but the alpha level can be varied, if the supporting information is present in the region object.

pvalue (region,left,right): Retrieves pvalues for all possible region hypotheses (indicated with left and right bound) from the region object. Only able to return the value if the allpvalues matrix is stored in the region object.

Author(s)

Rosa Meijer: <`r.j.meijer@lumc.nl`>

See Also

`regionmethod`, `regionpick`.

Examples

```
#generate data, where the response Y is associated with certain groups of covariates
#namely cov 3-6, 9-12, 15-18
set.seed(1)
n=100
p=20
X <- matrix(rnorm(n*p),n,p)
beta <- c(rep(0,2),rep(1,4),rep(0,2),rep(1,4),rep(0,2),rep(1,4),rep(0,2))
Y <- X %*% beta + rnorm(n)

# Define the local test to be used in the closed testing procedure
mytest <- function(left,right)
```

```
{
  X <- X[,(left:right),drop=FALSE]
  lm.out <- lm(Y ~ X)
  x <- summary(lm.out)
  return(pf(x$fstatistic[1],x$fstatistic[2],x$fstatistic[3],lower.tail=FALSE))
}

# perform the region procedure
reg <- regionmethod(rep(1,p), mytest, isadjusted=TRUE, all_pvalues=TRUE)
summary(reg)

#what are the smallest regions that are found to be significant?
implications(reg)

#what are the smallest regions that are found to be significant at an alpha-level of 0.03?
implications(reg, alpha=0.03)

#what is the adjusted p-value of the overall null-hypothesis
#(corresponding to the region ranging from 1 to 20)?
pvalue(reg,1,20)

#how many covariates within the full region of length 20 are at least associated with the response?
regionpick(reg, list(c(1,p)), alpha=0.05)

#visualize the results by either plotting a polygon corresponding to the underlying graph
regionplot(reg)

#or by plotting the graph itself
regionplot2(reg)
```

`Region Testing` *Testing of all possible region hypotheses*

Description

Tests all possible region hypotheses in a given region while controlling the FWER, using a user-specified local test.

Usage

```
regionmethod (weights, test, alpha_max = 0.05, all_pvalues = FALSE,
              isadjusted = FALSE, verbose = FALSE)
```

Arguments

`weights`	a vector that indicates the weight each elementary hypotheses should receive in the multiple testing procedure. The length of the vector should equal the number of elementary hypotheses. All values should be strictly positive.

`test`	A function that performs the local test. The function should have the left and rightbound of the region as input (as two separate numbers), and return a p-value.
`alpha_max`	The significance level of the test procedure.
`all_pvalues`	If set to TRUE, the procedure will return a matrix with the p-values of all tested region hypotheses. If set to FALSE, only the p-values of the implications will be returned.
`isadjusted`	If set to TRUE, adjusted p-values will be calculated. Otherwise, the p-values of all rejected hypotheses will equal alpha_max.
`verbose`	If set to TRUE, while running the method, a counter will indicate how many region hypotheses are already rejected.

Details

The function `regionmethod` tests all possible region hypotheses within one main interval, while controlling the familywise error rate.

Value

The function `regionmethod` returns an object of class `region`.

Note

The number of region hypotheses is quadratic in the number of elementary hypotheses. The number of elementary hypotheses should for computational reasons not be too large. For values in between 1000 and 10.000 the region procedure can still be used, but one might consider calculating the individual raw p-values beforehand.

If both `isadjusted` and `all_pvalues` are set to TRUE, afterwards implications can be obtained for all alpha-values smaller or equal to `alpha_max`.

Author(s)

Rosa Meijer: `<r.j.meijer@lumc.nl>`

References

Meijer, Krebs and Goeman (2015) Statistical Applications in Genetics and Molecular Biology 14 (1) 1-19.

Examples

```
#generate data, where the response Y is associated with certain groups of covariates
#namely cov 3-6, 9-12, 15-18
set.seed(1)
n=100
p=20
X <- matrix(rnorm(n*p),n,p)
beta <- c(rep(0,2),rep(1,4),rep(0,2),rep(1,4),rep(0,2),rep(1,4),rep(0,2))
```

```
Y <- X %*% beta + rnorm(n)

# Define the local test to be used in the closed testing procedure
mytest <- function(left,right)
{
  X <- X[,(left:right),drop=FALSE]
  lm.out <- lm(Y ~ X)
  x <- summary(lm.out)
  return(pf(x$fstatistic[1],x$fstatistic[2],x$fstatistic[3],lower.tail=FALSE))
}

# perform the region procedure
reg <- regionmethod(rep(1,p), mytest, isadjusted=TRUE)
summary(reg)

#what are the smallest regions that are found to be significant?
implications(reg)

#at least how many covariates within the full region of length 20
#are associated with the response?
regionpick(reg, list(c(1,p)), alpha=0.05)

#visualize the results by either plotting a polygon corresponding to the underlying graph
regionplot(reg)

#or by plotting the graph itself
regionplot2(reg)
```

`regionpick` *Confidence limits for the number of false hypotheses in a given region.*

Description

Calculates confidence limits for the number of false hypotheses on the basis of the region procedure within one or more regions.

Usage

```
regionpick (region, intervals, alpha, silent = FALSE, ignore_weights = TRUE)
```

Arguments

`region`	An object of class `region`, typically created through a call to `regionmethod`.
`intervals`	A list containing one or more regions, specified by a left and rightbound.
`alpha`	For region objects with adjusted p-values, specifies the value of alpha for which confidence limits are to be calculated (optional).
`silent`	If `FALSE`, prints the result to the screen.

`ignore_weights`	If set to TRUE, a confidence interval for the number of false elementary hypotheses will be computed. If set to FALSE, a confidence interval for the combined weight of false elemenatary hypotheses will be computed.

Details

The function `regionpick` calculates a confidence interval for the number (or weight) of false hypotheses among a selected set of elementary hypotheses.

Value

The function returns the lower confidence limit for the number of false hypotheses (i.e. true findings) among the set of elementary hypotheses as specified by `intervals`. The upper confidence limit is always equal to the size of the set.

Author(s)

Rosa Meijer: `<r.j.meijer@lumc.nl>`

Examples

```
#generate data, where the response Y is associated with certain groups of covariates
#namely cov 3-6, 9-12, 15-18
set.seed(1)
n=100
p=20
X <- matrix(rnorm(n*p),n,p)
beta <- c(rep(0,2),rep(1,4),rep(0,2),rep(1,4),rep(0,2),rep(1,4),rep(0,2))
Y <- X %*% beta + rnorm(n)

# Define the local test to be used in the closed testing procedure
mytest <- function(left,right)
{
  X <- X[,(left:right),drop=FALSE]
  lm.out <- lm(Y ~ X)
  x <- summary(lm.out)
  return(pf(x$fstatistic[1],x$fstatistic[2],x$fstatistic[3],lower.tail=FALSE))
}

# perform the region procedure
reg <- regionmethod(rep(1,p), mytest, isadjusted=TRUE)
summary(reg)

#what are the smallest regions that are found to be significant?
implications(reg)

#how many covariates within the full region of length 20 are at least associated with the response?
regionpick(reg, list(c(1,p)), alpha=0.05)

#how many covariates within the two subregions, (1,5) and (16,20)
#are at least associated with the response?
```

```
regionpick(reg, list(c(1,5),c(16,20)))
```

`Select with shortcuts` *Confidence limits for the number of true hypotheses, with shortcuts.*

Description

Calculates confidence limits for the number of true hypotheses on the basis of the closed testing procedure using specific local tests that allow shortcuts.

Usage

```
pickFisher (p, select = seq_along(p), alpha=0.05, silent=FALSE)

curveFisher (p, select = seq_along(p), order, alpha=0.05, plot = TRUE)

pickSimes (hommel, select, alpha=0.05, silent=FALSE)

curveSimes (hommel, select, order, alpha=0.05, plot = TRUE)

pickMeinshausen (p, PM, select = seq_along(p), alpha=0.05, silent=FALSE)

curveMeinshausen (p, PM, select = seq_along(p), order, alpha=0.05, plot = TRUE)
```

Arguments

`p`	The vector of p-values for all tested hypotheses.
`hommel`	The hommel object, obtained from the hommelFast function.
`PM`	A matrix of permutation p-values. Rows are hypotheses; columns are permutations.
`select`	The indexing vector of the p-values of the hypotheses to be selected. May be any type of appropriate indexing vector (integers, logical, or character).
`order`	The indexing vector specifying the order in which p-values of the hypotheses are to be rejected. May be integer or character.
`alpha`	The significance level of the test procedure.
`silent`	If `FALSE`, prints verbose result to the screen.
`plot`	If `TRUE`, plots the curve of correct rejections versus total rejections.

Details

The results of the `pickFisher`, `pickSimes` and `pickMeinshausen` functions are identical to applying `closed` and `pick`, for specific choices of the local test, but are computationally more efficient. `pickFisher` uses local tests based on Fisher combinations. This local test is only valid if p-values of true hypotheses are independent. `pickSimes` uses a local test based on Simes' inequality. It is valid if p-values of true hypotheses are independent but also under some forms of positive correlations. The Hommel variant of the Simes local test is valid under any dependence structure of the p-values. `pickMeinshausen` is a permutation-based variant of `pickSimes`. See the reference below.

In the curve functions, the user may specify either `select` or `order`. Specifying `order` fixes the precise order in which hypotheses are selected, whereas specifying `select` only specifies which hypotheses are candidates for selection, leaving the order to be chosen by the function to maximize the number of correct rejections.

Value

For `pickFisher`, `pickSimes` and `pickMeinshausen`, the function returns the lower confidence limit for the number of false hypotheses (correct rejection) among the set `reject`. The upper confidence limit is always equal to the number of rejections made. `curveFisher` and `curveSimes` return the same confidence limit, but for selecting only the first 1,2,3,... hypotheses from the selected set.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Aldo Solari

References

Goeman and Solari (2011) Statistical Science 26 (4) 584-597.

Meinshausen (2006) Scandinavian Journal of Statistics 33 (2), 227-237.

Examples

```
# Fisher's method
data(NAEP)
pickFisher(NAEP, c("NH", "NC", "IA"))
pickFisher(NAEP, 1:7)
curveFisher(NAEP)
curveFisher(NAEP, order=7:1)


# Simes method
hom <- hommelFast(NAEP)
pickSimes(hom, c("NH", "NC", "IA"))
pickSimes(hom, 1:7)
curveSimes(hom)
curveSimes(hom, select=1:7)

# Meinshausen's method
# This example uses data from the multtest package on bioconductor
if(require("multtest")) {
  data(golub)
  smallglb <- golub[1:500,]
  TM<-sapply(1:nrow(smallglb), function(i) {
    mt.sample.teststat(smallglb[i,], golub.cl,test="t.equalvar",B=500)
  })
  PM<-2*(1-pt(abs(TM),df=length(golub.cl)-2)) # permutation matrix
```

```
  # p-values
p<-apply(smallglb,1, function(z) t.test(z[golub.cl==0],z[golub.cl==1],var.equal=TRUE)$p.value)

  pickMeinshausen(p, PM, select=1:100)
  pickMeinshausen(p, PM, select=sort.list(p)[1:100])
  curveMeinshausen(p,PM, select=1:200)
  curveMeinshausen(p,PM, order=1:200)

}
```

`structured Holm` *Testing of hypotheses, forming a DAG, by using a variant of Holm's method.*

Description

Tests all hypotheses in a given DAG while controlling the FWER, using a user-specified local test.

Usage

```
 structuredHolm (DAGstructure, test, alpha_max = 0.05, isadjusted = FALSE,
optimization = "none", pvalues = NULL, verbose = FALSE)
```

Arguments

`DAGstructure`	DAGstructure object, as returned by the function `construct`.
`test`	A function that performs the local test. The function should have a set as input and return a p-value.
`alpha_max`	The significance level of the test procedure.
`isadjusted`	If set to TRUE, adjusted p-values will be calculated. Otherwise, the p-values of all rejected hypotheses will equal alpha_max.
`optimization`	Can be, in ascending order of accuracy and computational costs: "none", "LP" (linear programming) or "ILP" (integer linear programming).
`pvalues`	Optional (in case of stored p-values): a vector in which the raw p-values of the exact sets as found in the DAGstructure argument are stored (in the same order). If the test function is provided, this argument is not necessary.
`verbose`	If set to TRUE, while running the method, a counter will indicate how many hypotheses are already rejected.

Details

The function `structuredHolm` tests all possible hypotheses within a given DAG structure, while controlling the familywise error rate.

Value

The function `structuredHolm` returns an object of class `DAG`.

Author(s)

Rosa Meijer: <`r.j.meijer@lumc.nl`>

References

Meijer and Goeman (2015) Briefings in Bioinformatics, submitted.

See Also

`DAG`, `DAGstructure`, `construct`, `DAGpick`.

Examples

```
#Generate data, where the response Y is associated with two (out of 4) covariates
set.seed(1)
n=100
p=4
X <- matrix(rnorm(n*p),n,p)
beta <- c(0,0.5,0.5,0)
Y <- X %*% beta + rnorm(n)

# Let us assume we have the following sets that we want to test:
sets <- list(c(1,2,3,4), c(1,2), c(2,3,4), c(2,3), 1, 2, 3, 4)
names(sets) <- c(1234, 12, 234, 23, 1, 2, 3, 4)

# Start by making the corresponding graph structure
struct <- construct(sets)

# Check whether the DAG has toway logical relations:
istwoway(struct)

# Define the local test to be used in the closed testing procedure.
# This test expects a set as input.
mytest <- function(set)
{
  X <- X[,set,drop=FALSE]
  lm.out <- lm(Y ~ X)
  x <- summary(lm.out)
  return(pf(x$fstatistic[1],x$fstatistic[2],x$fstatistic[3],lower.tail=FALSE))
}

# Perform the structuredHolm procedure.
DAG <- structuredHolm(struct, mytest, isadjusted=TRUE)
summary(DAG)

# What are the smallest sets that are found to be significant?
implications(DAG)

# What is the adjusted p-value of the null-hypothesis corresponding to the fourth set,
# which is set c(2,3)?
# To look up the pvalue, the function uses the index or name of the set
```

```
# in the list of sets stored in the DAGstructure.
# (Note that, if there were duplicate sets in the original list, this index can be different from
# the one in the original list given to \code{construct})
pvalue(DAG,4)
pvalue(DAG, "23") #as above, but while using names

# How many of the elementary hypotheses (the last 4 sets) have to be false
# with probability 1-alpha?
# Sets (don't have to be elementary hypotheses in general) must be specified
# by their index or name.
DAGpick(DAG, 5:8)
DAGpick(DAG, c("1","2","3","4")) #as above, but while using names
```

Index