Package ‘micd’

February 17, 2023

Type Package
Title Multiple Imputation in Causal Graph Discovery
Version 1.1.1
Date 2023-02-09
Maintainer Ronja Foraita <foraita@leibniz-bips.de>
Description Modified functions of the package 'pcalg' and some additional functions to run the PC and the FCI (Fast Causal Inference) algorithm for constraint-based causal discovery in incomplete and multiply imputed datasets.
Depends R (>= 3.0.2), pcalg, mice
Imports RBGL, Rfast, methods, stats, utils
Suggests nnet, ranger, Rgraphviz, testthat (>= 3.0.0)
License GPL (>= 3)
Encoding UTF-8
RoxygenNote 7.2.3
URL https://github.com/bips-hb/micd
BugReports https://github.com/bips-hb/micd/issues
Config/testthat/edition 3
NeedsCompilation no
Author Ronja Foraita [aut, cph, cre] (<https://orcid.org/0000-0003-2216-6653>),
       Janine Witte [aut]
Repository CRAN
Date/Publication 2023-02-17 17:40:02 UTC
**R topics documented:**

- boot.graph .......................................................... 2
- disCItwd .............................................................. 3
- disMItest ............................................................. 5
- fciMI ................................................................. 6
- flexCItest ............................................................ 8
- flexCItwd ............................................................ 9
- flexMItest ........................................................... 10
- gaussCItwd .......................................................... 12
- gaussMItest .......................................................... 13
- getSuff ............................................................... 14
- make.formulas.saturated ........................................... 16
- makeResiduals ........................................................ 17
- mixCItest ............................................................ 19
- mixCItwd ............................................................. 20
- mixMItest ............................................................ 21
- pcMI ................................................................. 23
- skeletonMI ........................................................... 25
- with_graph ........................................................... 26

**Index**

<table>
<thead>
<tr>
<th>boot.graph</th>
<th>Bootstrap Resampling for the PC-MI- and the FCI-MI-algorithm</th>
</tr>
</thead>
</table>

**Description**

Generate R bootstrap replicates for the PC or FCI algorithm for data with missing values.

**Usage**

```r
boot.graph(
  data,
  select = NULL,
  method = c("pcMI", "fciMI"),
  method.mice = NULL,
  args,
  R,
  m = 10,
  args.residuals = NULL,
  seed = NA,
  quickpred = FALSE,
  ...
)
```
**Arguments**

- **data**: Data frame with missing values
- **select**: Variable of integers, indicating columns to select from a data frame; only continuous variables can be included in the model selection
- **method**: Character string specifying the algorithm for causal discovery from the package 'pcalg'.
- **method.mice**: Character string specifying imputation method; see `mice::mice()` for more information.
- **args**: Arguments passed to `method`. NOTE: argument `labels` is set internally and should not be used!
- **R**: A positive integer number of bootstrap replications.
- **m**: Number of chains included in `mice()`.
- **args.residuals**: (Optional) list containing vertices and confounders. May be specified when residuals for vertices should be calculated in each bootstrap data set. See `makeResiduals()` for more information.
- **seed**: A positive integer that is used as argument for `set.seed()`.
- **quickpred**: If true, `mice` uses quickpred to select predictors.
- **...**: Further arguments passed to the imputation function `mice()`.

**Value**

List of objects of class pcalgo (see `pcalg::pcAlgo`) or of fcmialgo (see `pcalg::fciAlgo`).

**Examples**

```r
data(windspeed)
daten <- mice::ampute(windspeed)$amp

bgraph <- boot.graph(data = daten,
                     method = "pcMI",
                     args = "solve.confl = TRUE, alpha = 0.05",
                     R = 5)
```

**Description**

A wrapper for `pcalg::disCItest`, to be used within `pcalg::skeleton`, `pcalg::pc` or `pcalg::fci` when the data contain missing values. Observations where at least one of the variables involved in the test is missing are deleted prior to performing the test (test-wise deletion).
Usage

discItwd(x, y, S = NULL, suffStat)

Arguments

x, y, S (Integer) position of variable X, Y and set of variables S, respectively, in suffStat. It is tested whether X and Y are conditionally independent given the subset S of the remaining variables.

suffStat A list with three elements, "dm", "nlev", "adaptDF": each corresponding to the above arguments. Can be obtained from a data.frame of factor variables using the suffStat function (see example section).

Details

See discItest for details on the G square test. Test-wise deletion is valid if missingness does not jointly depend on X and Y.

Value

A p-value.

See Also

pcalg::discItest for complete data, discMItest for multiply imputed data

Examples

```r
## load data (200 observations)
data(gmD)
dat <- gmD$x[1:1000,]

## delete some observations of X2 and X3
set.seed(123)
dat[sample(1:1000, 50), 2] <- NA
dat[sample(1:1000, 50), 3] <- NA

## analyse incomplete data
# test-wise deletion =========
sufftwd <- get Suff(dat, test = "discItwd")
discItwd(1, 3, NULL, suffStat = sufftwd)

# list-wise deletion =========
dat2 <- dat[complete.cases(dat), ]
suffStat2 <- get Suff(dat2, test = "discItest", adaptDF = FALSE)
discItest(1, 3, NULL, suffStat = suffStat2)

## use discItwd within pcalg::pc =========
pc.fit <- pc(suffStat = sufftwd, indepTest = discItwd, alpha = 0.1, p = 5)
pc.fit
```
if (requireNamespace("Rgraphviz", quietly = TRUE))
plot(pc.fit)

---

disMItest

### G square Test for (Conditional) Independence between Discrete Variables after Multiple Imputation

**Description**

A modified version of `pcalg::disCItest`, to be used within `pcalg::skeleton`, `pcalg::pc` or `pcalg::fci` when multiply imputed data sets are available. Note that in contrast to `pcalg::disCItest`, the variables must here be coded as factors.

**Usage**

```
disMItest(x, y, S = NULL, suffStat)
```

**Arguments**

- `x, y`, `S` (Integer) position of variable X, Y and set of variables S, respectively, in `suffStat`. It is tested whether X and Y are conditionally independent given the subset S of the remaining variables.
- `suffStat` A list of `data.frame`s containing the multiply imputed data sets. Usually obtained from a `mice::mids` object using `mice::complete` with argument `action="all"`. All variables must be coded as factors. NO warning is issued if the variables are not coded as factors!

**Details**

See `pcalg::disCItest` for details on the G square test. `disMItest` applies this test to each `data.frame` in `suffStat`, then combines the results using the rules in Meng & Rubin (1992). Degrees of freedom are never adapted, and there is no minimum required sample size, while `pcalg::disCItest` requires 10*df observations and otherwise returns a p-value of 1.

**Value**

A p-value.

**Author(s)**

Janine Witte

**References**

See Also

pcalg::disCItest for complete data, disCItd for test-wise deletion

Examples

```r
## load data (200 observations) and factorise
data(gmD)
dat <- gmD$x[1:1000,]
dat[] <- lapply(dat, as.factor)

## delete some observations of X2 and X3
set.seed(123)
dat[sample(1:1000, 40), 2] <- NA
dat[sample(1:1000, 40), 3] <- NA

## impute missing values under model with two-way interactions
form <- make.formulas.saturated(dat, d = 2)
imp <- mice::mice(dat, formulas = form, printFlag = FALSE)
imp <- mice::complete(imp, action = "all")

## analyse imputed data
disMItest(1, 3, NULL, suffStat = imp)

## use disMItest within pcalc::pc
pc.fit <- pc(suffStat = imp, indepTest = disMItest, alpha = 0.01, p = 5)
pc.fit

if(require("Rgraphviz", character.only = TRUE, quietly = TRUE)){
  plot(pc.fit)
}
```

fciMI

Estimate a PAG by the FCI-MI Algorithm for Multiple Imputed Data Sets of Continuous Data

Description

This function is a modification of pcalc::fci() to be used for multiple imputation.

Usage

```r
fciMI(
data, alpha, labels, p,
skel.method = c("stable", "original"),
```
fciMI

```r
type = c("normal", "anytime", "adaptive"),
fixedGaps = NULL,
fixedEdges = NULL,
NAdelete = TRUE,
m.max = Inf,
pdsep.max = Inf,
rules = rep(TRUE, 10),
doPdsep = TRUE,
biCC = FALSE,
conservative = FALSE,
maj.rule = FALSE,
verbose = FALSE
)
```

Arguments

- **data**: An object of type mids, which stands for 'multiply imputed data set', typically created by a call to function mice()
- **alpha**: Significance level (number in (0,1) for the conditional independence tests
- **labels**: (Optional) character vector of variable (or "node") names. Typically preferred to specifying p.
- **p**: (Optional) number of variables (or nodes). May be specified if labels are not, in which case labels is set to 1:p.
- **skel.method**: Character string specifying method; the default, "stable" provides an order-independent skeleton, see `pcalg::skeleton()` for details.
- **type**: Character string specifying the version of the FCI algorithm to be used. See `pcalg::fci()` for details.
- **fixedGaps**: See `pcalg::fci()` for details.
- **fixedEdges**: See `pcalg::fci()` for details.
- **NAdelete**: See `pcalg::fci()` for details.
- **m.max**: Maximum size of the conditioning sets that are considered in the conditional independence tests.
- **pdsep.max**: See `pcalg::fci()` for details.
- **rules**: Logical vector of length 10 indicating which rules should be used when directing edges. The order of the rules is taken from Zhang (2008).
- **doPdsep**: See `pcalg::fci()` for details.
- **biCC**: See `pcalg::fci()` for details.
- **conservative**: See `pcalg::fci()` for details.
- **maj.rule**: See `pcalg::fci()` for details.
- **verbose**: If true, more detailed output is provided.

Value

See `pcalg::fci()` for details.
Author(s)

Original code by Diego Colombo, Markus Kalisch, and Joris Mooij. Modifications by Ronja Foraita.

Examples

daten <- windspeed[,1]
for(i in 2:ncol(windspeed)) daten <- c(daten, windspeed[,i])
 daten[sample(1:length(daten), 260)] <- NA
daten <- matrix(daten, ncol = 6)

## Impute missing values
imp <- mice(daten, printFlag = FALSE)
fci.res <- fciMI(data = imp, label = colnames(imp$data), alpha = 0.01)

if (requireNamespace("Rgraphviz", quietly = TRUE))
plot(fc.res)

flexCItest

Wrapper for gaussCItest, disCItest and mixCItest

Description

A plug-in conditional independence test for `pcalg::skeleton()`, `pcalg::pc()` or `pcalg::fci()` when multiply imputed data sets are available. `flexCItest()` detects whether variables are continuous, discrete or mixed, and automatically switches between `gaussCItest()` (continuous only), `disCItest()` (discrete only) and `mixCItest()` (mixed variables).

Usage

flexCItest(x, y, S = NULL, suffStat)

Arguments

x, y, S  (integer) position of variable X, Y and set of variables S, respectively, in the dataset. It is tested whether X and Y are conditionally independent given the subset S of the remaining variables.

suffStat  a list generated using `getSuff()` with test="flexMItest". See below for details.
Details

suffStat needs to be a list with four elements named datlist, corlist, conpos and dispos. datlist is the list of imputed datasets. corlist is a list with M+1 elements, where M is the number of imputed datasets. For i=1,...,M, the i-th element of corlist is the correlation matrix of the continuous variables in the i-th imputed dataset; the (M+1)-th element is the number of rows in each imputed dataset. conpos is a vector containing the integer positions of the continuous variables in the original dataset. dispos is a vector containing the integer positions of the discrete variables in the original dataset.

Value

A p-value.

See Also

gaussCItest(), disCItest() and mixCItest().

Examples

```r
# load data (numeric and factor variables)
dat <- toenail2[1:400, ]
# obtain correct input 'suffStat' for 'flexCItest'
suff <- get Suff(dat, test="flexCItest")
flexCItest(2,3,NULL, suffStat = suff)
```

Description

A plug-in conditional independence test for pcalg::skeleton, pcalg::pc or pcalg::fci when the data contain missing values. Observations where at least one of the variables involved in the test is missing are deleted prior to performing the test (test-wise deletion). The function flexCItd detects whether variables are continuous, discrete or mixed, and automatically switches between gaussCItd (continuous only), link{disCItd} (discrete only) and mixCItd (mixed).

Usage

flexCItd(x, y, S = NULL, data)

Arguments

x, y, S  (Integer) position of variable X, Y and set of variables S, respectively, in each correlation matrix in suffStat. It is tested whether X and Y are conditionally independent given the subset S of the remaining variables.

data  A data frame
Example

```r
## load data (numeric and factor variables)
dat <- toenail2[1:400, ]

## delete some observations
set.seed(123)
dat[sample(400, 20), 2] <- NA
dat[sample(400, 30), 4] <- NA

## obtain correct input 'suffStat' for 'flexMItest'
suff <- getSuff(imp, test="flexCItwd")

## analyse data
# continuous variables only
flexCItwd(4, 5, NULL, dat)

# discrete variables only
flexCItwd(2, 3, NULL, dat)

# mixed variables
flexCItwd(2, 3, 4, dat)
```
flexMItest

Details

dsuffStat needs to be a list with four elements named datlist, corlist, conpos and dispos. datlist is the list of imputed datasets. corlist is a list with M+1 elements, where M is the number of imputed datasets. For i=1,...,M, the the i-th element of corlist is the correlation matrix of the continuous variables in the i-th imputed dataset; the (M+1)-th element is the number of rows in each imputed dataset. conpos is a vector containing the integer positions of the continuous variables in the original dataset. dispos is a vector containing the integer positions of the discrete variables in the original dataset.

Value

A p-value.

See Also

gaussMItest, disMItest and mixMItest

Examples

```r
## load data (numeric and factor variables)
library(ranger)
dat <- toenail2[1:400, ]

## delete some observations
set.seed(123)
dat[sample(400, 20), 2] <- NA
dat[sample(400, 30), 4] <- NA

## impute missing values using random forests
imp <- mice::mice(dat, method = "rf", m = 3, printFlag = FALSE)

## obtain correct input 'suffStat' for 'flexMItest'
suff <- getSuff(imp, test="flexMItest")

## analyse data
# continuous variables only
flexMItest(4,5,NULL, suffStat = suff)
implist <- complete(imp, action="all")
gaussSuff <- c(lapply(implist, function(i){cor(i[ ,c(4,5)])}), n = 400)
gaussMItest(1,2,NULL, suffStat = gaussSuff)
flexCItd(4, 5, NULL, dat)

# discrete variables only
flexMItest(2,3,NULL, suffStat = suff)
disMItest(2,3,NULL, suffStat = complete(imp, action="all"))
flexCItd(2,3,NULL, dat)

# mixed variables
flexMItest(2,3,4, suffStat = suff)
mixMItest(2,3,4, suffStat = complete(imp, action="all"))
flexCItd(2,3,4, dat)
```
Description

A wrapper for `pcalg::gaussCItest`, to be used within `pcalg::skeleton`, `pcalg::pc` or `pcalg::fci` when the data contain missing values. Observations where at least one of the variables involved in the test is missing are deleted prior to performing the test (test-wise deletion).

Usage

`gaussCItest(x, y, S = NULL, suffStat)`

Arguments

- `x, y, S` (integer) position of variable X, Y and set of variables S, respectively, in each correlation matrix in `suffStat`. It is tested whether X and Y are conditionally independent given the subset S of the remaining variables.
- `suffStat` data.frame containing the raw data.

Value

See `pcalg::gaussCItest` for details on Fisher's z-test. Test-wise deletion is valid if missingness does not jointly depend on X and Y. A p-value.

See Also

`pcalg::condIndFisherZ()` for complete data, `gaussCItestMI()` for multiply imputed data

Examples

```r
## load data (numeric variables)
dat <- as.matrix(windspeed)

## delete some observations
set.seed(123)
dat[sample(1:length(dat), 260)] <- NA

## analyse data
# complete data:
suffcomplete <- getSuff(windspeed, test="gaussCItest")
gaussCItest(1, 2, c(4,5), suffStat = suffcomplete)
```
# test-wise deletion: =========
gaussCItwd(1, 2, c(4,5), suffStat = dat)

# list-wise deletion: =========
sufflwd <- getSuff(dat[complete.cases(dat), ], test="gaussCItest")
gaussCItest(1, 2, c(4,5), suffStat = sufflwd)

## use gaussCItwd within pcalg::pc
pc.fit <- pc(suffStat = dat, indepTest = gaussCItwd, alpha = 0.01, p = 6)
pc.fit

gaussMItest

Test Conditional Independence of Gaussians via Fisher’s Z Using Multiple Imputations

Description

A modified version of pcalg::gaussCItest, to be used within pcalg::skeleton, pcalg::pc or pcalg::fci when multiply imputed data sets are available.

Usage

gaussMItest(x, y, S, suffStat)

gaussCItestMI(x, y, S = NULL, data)

Arguments

x, y, S (Integer) position of variable X, Y and set of variables S, respectively, in the adjacency matrix. It is tested, whether X and Y are conditionally independent given the subset S of the remaining nodes.

suffStat A list of length m+1, where m is the number of imputations; the first m elements are the covariance matrices of the m imputed data sets, the m-th element is the sample size. Can be obtained from a mids object by getSuff(mids, test="gaussMItest")

data An object of type mids, which stands for ’multiply imputed data set’, typically created by a call to function mice()

Details

gaussMItest is faster, as it uses pre-calculated covariance matrices.

Value

A p-value.
Examples

```r
## load data (numeric variables)
dat <- as.matrix(windspeed)

## delete some observations
set.seed(123)
dat[sample(1:length(dat), 260)] <- NA

## Impute missing values under normal model
imp <- mice(dat, method = "norm", printFlag = FALSE)

## analyse data
# complete data:
suffcomplete <- getSuff(windspeed, test = "gaussCItest")
gaussCItest(1, 2, c(4,5), suffStat = suffcomplete)
# multiple imputation:
suffMI <- getSuff(imp, test = "gaussMItest")
gaussMItest(1, 2, c(4,5), suffStat = suffMI)
gaussCItestMI(1, 2, c(4,5), data = imp)
# test-wise deletion:
gaussCItwd(1, 2, c(4,5), suffStat = dat)
# list-wise deletion:
dat2 <- dat[complete.cases(dat), ]
sufflwd <- getSuff(dat2, test = "gaussCItest")
gaussCItest(1, 2, c(4,5), suffStat = sufflwd)

## use gaussMItest or gaussCItestMI within pcalg::pc
(pc.fit <- pc(suffStat = suffMI, indepTest = gaussMItest, alpha = 0.01, p = 6))
(pc.fit <- pc(suffStat = imp, indepTest = gaussCItestMI, alpha = 0.01, p = 6))
```

---

**getSuff**

*Obtain 'suffStat' for conditional independence testing*

**Description**

A convenience function for transforming a multiply imputed data set into the 'suffStat' required by `pcalg::gaussCItest()`, `pcalg::disCItest()`, `mixCItest()`, `flexCItest()`, `gaussMItest()`, `disMItest()`, `mixMItest()` and `flexMItest()`.

**Usage**

```r
getSuff(  
  X,
  test = c("gaussCItest", "gaussMItest", "disCItest", "disMItest", "disCItwd",  
            "mixCItest", "mixMItest", "flexMItest", "flexCItest"),
  adaptDF = NULL,
  nlev = NULL
)
```
Arguments

X
  For ‘test=xxxCItest’: a data.frame or matrix; for ‘test=xxxMItest’: an object of
class mice::mids, or a list of data.frames containing the multiply imputed data
sets.

test
  one of gaussCItest(), gaussMItest(), disCItest(), disMItest(), mixCItest(),
mixMItest(), flexCItest(), flexMItest().

adaptDF
  for discrete variables: logical specifying if the degrees of freedom should be
  lowered by one for each zero count. The value for the degrees of freedom cannot
  go below 1.

nlev
  (Optional) for discrete variables: vector with numbers of levels for each variable
  in the data.

Value

An R object that can be used as input to the specified conditional independence test:

Examples

# Example 1: continuous variables, no missing values ===============
data(windspeed)
dat1 <- as.matrix(windspeed)

  ## analyse data
  gaussCItest(1, 2, NULL, suffStat = getSuff(windspeed, test = "gaussCItest"))
mixCItest(1, 2, NULL, suffStat = windspeed)

  ## Example 2: continuous variables, multiple imputation =============
data2 <- mice::ampute(windspeed)$amp

  ## delete some observations
  set.seed(123)

  ## Impute missing values under normal model
  imp2 <- mice(dat2, method = "norm", printFlag = FALSE)

  ## analyse imputed data
  gaussMItest(1, 2, c(4,5), suffStat = getSuff(imp2, test="gaussMItest"))
mixMItest(1, 2, c(4,5), suffStat = getSuff(imp2, test="mixMItest"))
mixMItest(1, 2, c(4,5), suffStat = mice::complete(imp2, action="all"))
  flexMItest(1, 2, c(4,5), suffStat = getSuff(imp2, test="flexMItest"))

  ## Example 3: discrete variables, multiple imputation =============

  # simulate factor variables
  n <- 200
  set.seed(789)
x <- factor(sample(0:2, n, TRUE)) # factor, 3 levels
y <- factor(sample(0:3, n, TRUE)) # factor, 4 levels
z <- factor(sample(0:1, n, TRUE)) # factor, 2 levels
dat3 <- data.frame(x,y,z)
## delete some observations of z

```r
dat3[sample(1:n, 40), 3] <- NA
```

## impute missing values under saturated model

```r
form <- make.formulas.saturated(dat3)
imp3 <- mice::mice(dat3, method = "logreg", formulas = form, printFlag = FALSE)
```

## analyse imputed data

```r
disMItest(1, 3, 2, suffStat = getSuff(imp3, test="disMItest"))
disMItest(1, 3, 2, suffStat = mice::complete(imp3, action = "all"))
mixMItest(1, 3, 2, suffStat = getSuff(imp3, test="mixMItest"))
mixMItest(1, 3, 2, suffStat = mice::complete(imp3, action = "all"))
flexMItest(1, 3, 2, suffStat = getSuff(imp3, test="flexMItest"))
```

### Example 4: mixed variables, multiple imputation

```r
dat4 <- toenail2[1:400,]
sample.seed(123)
dat4[sample(400, 20), 2] <- NA
dat4[sample(400, 30), 4] <- NA
```

## impute missing values using random forests

```r
imp4 <- mice(dat4, method="rf", m = 3, printFlag = FALSE)
mixMItest(2, 3, 5, suffStat = getSuff(imp4, test="mixMItest"))
mixMItest(2, 3, 5, suffStat = mice::complete(imp4, action="all"))
flexMItest(2, 3, 5, suffStat = getSuff(imp4, test="flexMItest"))
```

---

### make.formulas.saturated

*Creates a formulas Argument*

#### Description

This helper function creates a valid formulas object. The formulas object is an argument to the `mice::mice` function. It is a list of formulas that specifies the target variables and the predictors by means of the standard ~ operator. In contrast to `mice::make.formulas`, which creates main effects formulas, `make.formulas.saturated` creates formulas including interaction effects.

#### Usage

```r
make.formulas.saturated(
data, blocks = mice::make.blocks(data), predictorMatrix = NULL, d = NULL)
```
**Arguments**

- `data` : A `data.frame` with the source data.
- `blocks` : An optional specification for blocks of variables in the rows. The default assigns each variable in its own block.
- `predictorMatrix` : A `predictorMatrix` specified by the user.
- `d` : maximum depth of interactions to be considered (1=no interactions, 2=two-way interactions, etc.)

**Value**

A list of formulas.

**Note**

A modification of `mice::make.formulas` by Stef van Buuren et al.

**See Also**

`mice::make.formulas`

**Examples**

```r
## main effects model:
data(nhanes)
f1 <- make.formulas(nhanes)
f1

## saturated model:
f2 <- make.formulas.saturated(nhanes)
f2
```

---

**Description**

Generate residuals based on variables in imputed data sets

**Usage**

```r
makeResiduals(data, v, confounder, method = c("res", "cc", "pd"))
```
makeResiduals

Arguments

- **data**: A data.frame.
- **v**: Vector of integers referring to the location of the variable(s) in the data set.
- **confounder**: Vector of integers referring to the location of the variable(s) in the data set (confounders are not included in the network!)
- **method**: Default method 'res' uses residuals, 'cc' uses complete cases and 'pd' uses pairwise deletion

Value

A data matrix of residuals.

Examples

```r
data(windspeed)
daten <- mice::ampute(windspeed)$amp

# Impute missing values
imp <- mice(daten, m = 5)

# Build residuals
knoten <- 1:4
confounder <- 5:6

# Residuals based on dataset with missing values
res.pd <- makeResiduals(daten, v = knoten, confounder = confounder, method = "pd")

# Residuals based in multiple imputed data
residuals <- list(data = list(), m = 5)
imp_c <- mice::complete(imp, "all")
for (i in 1:imp$m){
  residuals$data[[i]] <- makeResiduals(imp_c[[i]],
    v = knoten, confounder = confounder)
}

pc.res <- pcMI(data = residuals, p = length(knoten), alpha = 0.05)
fci.res <- fciMI(data = imp, p = length(knoten), alpha = 0.05)

if (requireNamespace("Rgraphviz", quietly = TRUE)){
  oldpar <- par(mfrow = c(1,2))
  plot(pc.res)
  plot(fci.res)
  par(oldpar)
}
```
mixCItest  

Likelihood Ratio Test for (Conditional) Independence between Mixed Variables

Description

A likelihood ratio test for (conditional) independence between mixed (continuous and unordered categorical) variables, to be used within pcalg::skeleton, pcalg::pc or pcalg::fci. It assumes that the variables in the test follow a Conditional Gaussian distribution, i.e. conditional on each combination of values of the discrete variables, the continuous variables are multivariate Gaussian. Each multivariate Gaussian distribution is allowed to have its own mean vector and covariance matrix.

Usage

mixCItest(x, y, S = NULL, suffStat, moreOutput = FALSE)

Arguments

x, y, S  
(Integer) position of variable X, Y and set of variables S, respectively, in suffStat. It is tested whether X and Y are conditionally independent given the subset S of the remaining variables.

suffStat  
A data.frame. Discrete variables must be coded as factors.

moreOutput  
If TRUE, the test statistic and the degrees of freedom are returned in addition to the p-value (only for mixed variables). Defaults to FALSE.

Details

The implementation follows Andrews et al. (2018). The same test is also implemented in TETRAD and in the R-package rcausal, a wrapper for the TETRAD Java library. Small differences in the p-values returned by CGtest and the TETRAD/rcausal equivalent are due to differences in handling sparse or empty cells.

Value

A p-value. If moreOutput=TRUE, the test statistic and the degrees of freedom are returned as well.

Author(s)

Janine Witte

References


**Examples**

```r
# load data (numeric and factor variables)
dat <- toenail2[, -1]

# analyse data
mixCItest(4, 1, NULL, suffStat = dat)
mixCItest(1, 2, 3, suffStat = dat)

## use mixCItest within pcalg::fci
fci.fit <- fci(suffStat = dat, indepTest = mixCItest, alpha = 0.01, p = 4)
if (requireNamespace("Rgraphviz", quietly = TRUE))
  plot(fci.fit)
```

---

### mixCItwd

**Likelihood Ratio Test for (Conditional) Independence between Mixed Variables with Missings**

**Description**

A version of `mixCItest`, to be used within `pcalg::skeleton`, `pcalg::pc` or `pcalg::fci` when the data contain missing values. Observations where at least one of the variables involved in the test is missing are deleted prior to performing the test (test-wise deletion).

**Usage**

```r
mixCItwd(x, y, S = NULL, suffStat)
```

**Arguments**

- `x, y, S` (Integer) position of variable X, Y and set of variables S, respectively, in `suffStat`. It is tested whether X and Y are conditionally independent given the subset S of the remaining variables.
- `suffStat` `data.frame`. Discrete variables must be coded as factors.

**Details**

See `mixCItest` for details on the assumptions of the Conditional Gaussian likelihood ratio test. Test-wise deletion is valid if missingness does not jointly depend on X and Y.

**Value**

A p-value.
mixMItest

**See Also**

mixCItest() for complete data, mixMItest() for multiply imputed data

**Examples**

```r
## load data (numeric and factor variables)
data(toenail2)
dat <- toenail2[, -1]

## delete some observations
set.seed(123)
dat[sample(2000, 20), 1] <- NA

## analyse data
# complete data: =========
mixCItest(1, 2, 4, suffStat=toenail2)
# test-wise deletion: =========
mixCItwd(1, 2, 4, suffStat = dat)
# list-wise deletion: =========
dat2 <- dat[complete.cases(dat), ]
mixCItest(1, 2, 4, suffStat = dat2)

## use mixCItwd within pcalg::pc
pc.fit <- pc(suffStat = dat, indepTest = mixCItwd, alpha = 0.01, p = 4)
```

---

**Description**

A modified version of mixCItest, to be used within pcalg::skeleton, pcalg::pc or pcalg::fci when multiply imputed data sets are available.

**Usage**

mixMItest(x, y, S = NULL, suffStat, moreOutput = FALSE)

**Arguments**

- `x, y, S` (integer) position of variable X, Y and set of variables S, respectively, in suffStat. It is tested whether X and Y are conditionally independent given the subset S of the remaining variables.
**mixMItest**

suffStat  A list of `data.frame`s containing the multiply imputed data sets. Usually obtained from a `mice::mids` object using `mice::complete` with argument `action="all"`. Discrete variables must be coded as factors.

moreOutput (only for mixed of discrete variables) If TRUE, the test statistic, its main components and the degrees of freedom are returned in addition to the p-value. Defaults to FALSE.

**Details**

See `mixCItest` for details on the assumptions of the Conditional Gaussian likelihood ratio test. `CGtestMI` applies this test to each `data.frame` in `suffStat`, then combines the results using the rules in Meng & Rubin (1992).

**Value**

A p-value. If `moreOutput=TRUE`, the test statistic, its main components and the degrees of freedom are returned as well.

**Author(s)**

Janine Witte

**References**


**Examples**

```r
## load data (numeric and factor variables)
data(toenail2)
dat <- toenail2[1:1000,]

## delete some observations
set.seed(123)
dat[sample(1000, 20), 2] <- NA
dat[sample(1000, 30), 4] <- NA

## impute missing values using random forests (because of run time we just impute 2 chains)
imp <- mice(dat, method = "rf", m = 2, printFlag = FALSE)

## analyse data
# complete data:
mixCItest(2, 3, 5, suffStat = toenail2[1:1000,])
# multiple imputation:
suffMI <- complete(imp, action = "all")
mixMItest(2, 3, 5, suffStat = suffMI)
# test-wise deletion:
mixCItwd(2, 3, 5, suffStat = dat)
# list-wise deletion:
```
Estimate the Equivalence Class of a DAG Using the PC-MI Algorithm for Multiple Imputed Data Sets

Description

This function is a modification of `pcalg::pc()` to be used for multiple imputation.

Usage

```r
pcMI(
  data,
  alpha,
  labels,
  p,
  fixedGaps = NULL,
  fixedEdges = NULL,
  NAdelete = TRUE,
  m.max = Inf,
  u2pd = c("relaxed", "rand", "retry"),
  skel.method = c("stable", "original"),
  conservative = FALSE,
  maj.rule = FALSE,
  solve.conf1 = FALSE,
  verbose = FALSE
)
```

Arguments

- `data`: An object of type mids, which stands for 'multiply imputed data set', typically created by a call to function mice()
- `alpha`: Significance level (number in (0,1) for the conditional independence tests
- `labels`: (Optional) character vector of variable (or "node") names. Typically preferred to specifying p.
- `p`: (Optional) number of variables (or nodes). May be specified if labels are not, in which case labels is set to 1:p.
fixedGaps  A logical matrix of dimension p*p. If entry [i, j] or [j, i] (or both) are TRUE, the edge i-j is removed before starting the algorithm. Therefore, this edge is guaranteed to be absent in the resulting graph.

fixedEdges  A logical matrix of dimension p*p. If entry [i, j] or [j, i] (or both) are TRUE, the edge i-j is never considered for removal. Therefore, this edge is guaranteed to be present in the resulting graph.

NAdelete  If indepTest returns NA and this option is TRUE, the corresponding edge is deleted. If this option is FALSE, the edge is not deleted.

m.max  Maximal size of the conditioning sets that are considered in the conditional independence tests.

u2pd  String specifying the method for dealing with conflicting information when trying to orient edges (see details below).

skel.method  Character string specifying method; the default, "stable" provides an order-independent skeleton, see pcalg::skeleton() for details.

conservative  Logical indicating if the conservative PC is used. See pcalg::pc() for details.

maj.rule  Logical indicating that the triples shall be checked for ambiguity using a majority rule idea, which is less strict than the conservative PC algorithm. For more information, see pcalg::pc().

solve.confl  See pcalg::pc() for more details.

verbose  If TRUE, detailed output is provided.

Details
An object of class "pcAlgo" (see pcAlgo) containing an estimate of the equivalence class of the underlying DAG.

Value
See pcalg::pc() for more details.

Note
This is a modified function of pcalg::pc() from the package 'pcalg' (Kalisch et al., 2012; http://www.jstatsoft.org/v47/i11/).

Author(s)
Original code by Markus Kalisch, Martin Maechler, and Diego Colombo. Modifications by Ronja Foraita.

Examples

daten <- mice::ampute(windspeed)$amp

## Impute missing values
imp <- mice(daten)
pcMI(data = imp, label = colnames(imp$data), alpha = 0.01)
skeletonMI

Estimate (Initial) Skeleton of a DAG using the PC Algorithm for Multiple Imputed Data Sets of Continuous Data

Description

This function is a modification of `pcalg::skeleton()` to be used for multiple imputation.

Usage

```r
skeletonMI(
  data,
  alpha,
  labels,
  p,
  method = c("stable", "original"),
  m.max = Inf,
  fixedGaps = NULL,
  fixedEdges = NULL,
  NAdelete = TRUE,
  verbose = FALSE
)
```

Arguments

- `data` An object of type `mids`, which stands for 'multiply imputed data set', typically created by a call to function `mice()`
- `alpha` Significance level
- `labels` (Optional) character vector of variable (or "node") names. Typically preferred to specifying `p`
- `p` (Optional) number of variables (or nodes). May be specified if `labels` are not, in which case `labels` is set to `1:p`.
- `method` Character string specifying method; the default, "stable" provides an order-independent skeleton, see `pcalg::pc()` for details.
- `m.max` Maximal size of the conditioning sets that are considered in the conditional independence tests.
- `fixedGaps` Logical symmetric matrix of dimension `p*p`. If entry `[i,j]` is true, the edge `i-j` is removed before starting the algorithm. Therefore, this edge is guaranteed to be absent in the resulting graph.
- `fixedEdges` A logical symmetric matrix of dimension `p*p`. If entry `[i,j]` is true, the edge `i-j` is never considered for removal. Therefore, this edge is guaranteed to be present in the resulting graph.
- `NAdelete` Logical needed for the case `indepTest(*)` returns `NA`. If it is true, the corresponding edge is deleted, otherwise not.
- `verbose` If `TRUE`, detailed output is provided.
with_graph

Value

See `pcalg::skeleton()` for more details.

Note

This is a modified function of `pcalg::skeleton()` from the package 'pcalg' (Kalisch et al., 2012; http://www.jstatsoft.org/v47/i11/).

Author(s)

Original code by Markus Kalisch, Martin Maechler, Alain Hauser, and Diego Colombo. Modifications by Ronja Foraita.

Examples

```r
data(gmG)
n <- nrow(gmG$x)
V <- colnames(gmG$x) # labels aka node names
## estimate Skeleton
data_mids <- mice(gmG$x, printFlag = FALSE)
(skel.fit <- skeletonMI(data = data_mids, alpha = 0.01, labels = V, verbose = FALSE))
```

---

**with_graph**

*Evaluate Causal Graph Discovery Algorithm in Multiple Imputed Data sets*

Description

Evaluate Causal Graph Discovery Algorithm in Multiple Imputed Data sets

Usage

```r
with_graph(data, algo = c("pc", "fci", "fciPlus", "ges"), args, score = FALSE)
```

Arguments

- `data`: An object of type mids, which stands for 'multiply imputed data set', typically created by a call to function mice()
- `algo`: An algorithm for causal discovery from the package 'pcalg' (see details).
- `args`: Additional arguments passed to the algo. Must be a string vector starting with comma, i.e. ", ..."
- `score`: Logical indicating whether a score-based or a constrained-based algorithm is applied.

Value

A list object of S3 class `mice::mira-class`. 
Examples

data(windspeed)
dat <- as.matrix(windspeed)

## delete some observations
set.seed(123)
dat[sample(1:length(dat), 260)] <- NA

## Impute missing values under normal model
imp <- mice(dat, method = "norm", printFlag = FALSE)
mylabels <- names(imp$imp)
out.fci <- with_graph(data = imp,
        algo = "fciPlus",
        args = ", indepTest = gaussCItest, verbose = FALSE,
        labels = mylabels, alpha = 0.01")

out.ges <- with_graph(data = imp, algo = "ges", arg = NULL, score = TRUE)

if (requireNamespace("Rgraphviz", quietly = TRUE)){
  oldpar <- par(mfrow = c(1,2))
  plot(out.fci$res[[1]])
  plot(out.ges$res[[1]]$essgraph)
  par(oldpar)
}
Index

boot.graph, 2
complete, 5, 22
discItest, 3–6
discItest(), 8, 9, 15
discItwd, 3, 6
discMItest, 4, 5, 11
discMItest(), 14, 15

factor, 5
fci, 3, 5, 9, 10, 12, 13, 19–21
fciMI, 6
flexItest, 8
flexItest(), 8, 14, 15
flexItwd, 9
flexMItest, 10
flexMItest(), 14, 15

gaussItest, 12, 13
gaussItest(), 8, 9, 15
gaussItestMI (gaussMItest), 13
gaussItestMI(), 12
gaussItwd, 9, 12
gaussMItest, 10, 11, 13
gaussMItest(), 14, 15
getSuff, 10, 14

make.formulas, 16, 17
make.formulas.saturated, 16
makeResiduals, 17
makeResiduals(), 3
mice, 16
mice::mice(), 3
mids, 5, 15, 22
mixItest, 19, 20–22
mixItest(), 8, 9, 14, 15, 21
mixItwd, 9, 20
mixMItest, 10, 11, 21
mixMItest(), 14, 15, 21

pc, 3, 5, 9, 10, 12, 13, 19–21
pcalg::condIndFisherZ(), 12
pcalg::discItest(), 14
pcalg::fci(), 6–8
pcalg::fciAlgo, 3
pcalg::gaussItest(), 14
pcalg::pc(), 8, 23–25
pcalg::pcAlgo, 3
pcalg::skeleton(), 7, 8, 24–26
pcMI, 23

skeleton, 3, 5, 9, 10, 12, 13, 19–21
skeletonMI, 25

with_graph, 26