

# Package ‘ParallelPC’

October 12, 2022

**Type** Package

**Title** Parallellised Versions of Constraint Based Causal Discovery Algorithms

**Version** 1.2

**Date** 2015-10-30

**Author** Thuc Duy Le, Tao Hoang, Shu Hu, and Liang Zhang

**Maintainer** Thuc Duy Le <Thuc.Le@unisa.edu.au>

**Imports** methods

**Suggests** bnlearn, pcalg, parallel, Rgraphviz

**Description** Parallellise constraint based causality discovery and causal inference methods. The parallelised algorithms in the package will generate the same results as that of the 'pcalg' package but will be much more efficient.

**License** GPL (>= 2)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-12-31 08:37:51

## R topics documented:

cor2	2
fci_parallel	3
fci_stable	5
IDA_parallel	7
IDA_stable	9
jointIDA_direct	10
jointIDA_parallel	11
mccor	12
mcmig	13
mczf	14
mig	15
migsh	16
pcSelect_parallel	17

pcSelect_stable . . . . .	18
pc_parallel . . . . .	19
pc_stable . . . . .	21
rfci_parallel . . . . .	23
rfci_stable . . . . .	25
skeleton_parallel . . . . .	27
skeleton_stable . . . . .	29
smccor . . . . .	30
smcmig . . . . .	31
smczf . . . . .	32
zf . . . . .	33

<b>Index</b>	<b>34</b>
--------------	-----------

---

cor2	<i>The Pearson's correlation test</i>
------	---------------------------------------

---

### Description

Linear correlation: Pearson's linear correlation test.

### Usage

```
cor2(x, y, S, suffStat)
```

### Arguments

<code>x, y, S</code>	It is tested, whether <code>x</code> and <code>y</code> are conditionally independent given the subset <code>S</code> of the remaining nodes. <code>x</code> , <code>y</code> , <code>S</code> all are integers, corresponding to variable or node numbers.
<code>suffStat</code>	the dataset with rows are samples and columns are variables.

### Value

the p-value of the test.

### References

Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

### Examples

```
#####
## Using cor2 as a conditional independence test
#####
library(pcalg)
library(bnlearn)
data("gmG")
```

```
suffStat<-gmG$x
cor2(1,2,3,suffStat)
##Use cor2 with a causal discovery algorithm, e.g. PC
pc_stable(gmG$x, indepTest=cor2, p=ncol(gmG$x), alpha=0.01)
```

fci\_parallel

*Estimate a Partial Ancestral Graph using the FCI\_parallel algorithm***Description**

Estimate a Partial Ancestral Graph (PAG) from observational data, using the FCI\_parallel Algorithm. This is the parallelised version of the FCI algorithm in the pcalg package. The parameters are consistent with the FCI algorithm in pcalg, except the parameter num.cores for specifying the number of cores CPU.

**Usage**

```
fci_parallel(suffStat, indepTest, alpha, labels, p,
  skel.method = c("parallel"), mem.efficient = FALSE, 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, num.cores = detectCores())
```

**Arguments**

suffStat	Sufficient statistics: List containing all necessary elements for the conditional independence decisions in the function indepTest.
indepTest	Predefined function for testing conditional independence. The function is internally called as indepTest(x,y,S,suffStat), and tests conditional independence of x and y given S. Here, x and y are variables, and S is a (possibly empty) vector of variables (all variables are denoted by their column numbers in the adjacency matrix). suffStat is a list containing all relevant elements for the conditional independence decisions. The return value of indepTest is the p-value of the test for conditional independence.
alpha	Significance level for the individual 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, "parallel", uses the parallelised method to build the skeleton of the graph, see skeleton_parallel.
mem.efficient	Uses less amount of memory at any time point while running the algorithm.

type	Character string specifying the version of the FCI algorithm to be used. By default, it is "normal", and so the normal FCI algorithm is called. If set to "anytime", the 'Anytime FCI' is called and m.max needs to be specified. If set to "adaptive", the 'Adaptive Anytime FCI' is called and m.max is not used. For more information, see Details.
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	Maximum size of the conditioning sets that are considered in the conditional independence tests.
pdsep.max	Maximum size of Possible-D-SEP for which subsets are considered as conditioning sets in the conditional independence tests. See pcalg for more details.
rules	Logical vector of length 10 indicating which rules should be used when directing edges. See pcalg for more details.
doPdsep	If TRUE, Possible-D-SEP is computed for all nodes, and all subsets of Possible-D-SEP are considered as conditioning sets in the conditional independence tests, if not defined otherwise in pdsep.max. If FALSE, Possible-D-SEP is not computed, so that the algorithm simplifies to the Modified PC algorithm of Spirtes, Glymour and Scheines (2000, p.84).
biCC	If TRUE, only nodes on paths between nodes x and y are considered to be in Possible-D-SEP(x) when testing independence between x and y. Uses biconnected components, biConnComp from RBGL.
conservative	Logical indicating if the unshielded triples should be checked for ambiguity the second time when v-structures are determined.
maj.rule	Logical indicating if the unshielded triples should be checked for ambiguity the second time when v-structures are determined using a majority rule idea, which is less strict than the standard conservative. For more information, see details.
verbose	If true, more detailed output is provided.
num.cores	Numbers of cores CPU to run the algorithm

### Value

An object of class fciAlgo (see fciAlgo in the pcalg package) containing the estimated graph (in the form of an adjacency matrix with various possible edge marks), the conditioning sets that lead to edge removals (sepset) and several other parameters.

### References

1. Diego Colombo, Marloes H Maathuis, Markus Kalisch, Thomas S Richardson, et al. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, 40(1):294-321, 2012.

2. Markus Kalisch, Martin Machler, Diego Colombo, Marloes H Maathuis, and Peter Buhlmann. Causal inference using graphical models with the r package pcalg. *Journal of Statistical Software*, 47(11):1-26, 2012.

## Examples

```
#####
## Using fci_parallel without mem. efficeient
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
fci_parallel(suffStat, indepTest=gaussCItest, p=p, skel.method="parallel", alpha=0.01, num.cores=2)

#####
## Using fci_parallel with mem. efficeient
#####

suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
fci_parallel(suffStat, indepTest=gaussCItest, p=p, skel.method="parallel",
alpha=0.01, num.cores=2, mem.efficient=TRUE)

#####
## Using fci_parallel with mutual information test
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
#' # The first parameter is the dataset
fci_parallel(gmG$x, indepTest=mig, p=p, skel.method="parallel",
alpha=0.01, num.cores=2, mem.efficient=TRUE)
```

---

fci\_stable

*Estimate a PAG, using the FCI\_stable algorithm*

---

## Description

This is the FCI stable version in the pcalg package.

## Usage

```
fci_stable(suffStat, indepTest, alpha, labels, p, skel.method = c("stable",
"original", "stable.fast"), 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**

suffStat	Sufficient statistics: List containing all necessary elements for the conditional independence decisions in the function indepTest.
indepTest	Predefined function for testing conditional independence. The function is internally called as indepTest(x,y,S,suffStat), and tests conditional independence of x and y given S. Here, x and y are variables, and S is a (possibly empty) vector of variables (all variables are denoted by their column numbers in the adjacency matrix). suffStat is a list containing all relevant elements for the conditional independence decisions. The return value of indepTest is the p-value of the test for conditional independence.
alpha	Significance level for the individual 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 skeleton.
type	Character string specifying the version of the FCI algorithm to be used. By default, it is "normal", and so the normal FCI algorithm is called. If set to "anytime", the 'Anytime FCI' is called and m.max needs to be specified. If set to "adaptive", the 'Adaptive Anytime FCI' is called and m.max is not used. For more information, see the FCI function in the pcalg package.
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	Maximum size of the conditioning sets that are considered in the conditional independence tests.
pdsep.max	Maximum size of Possible-D-SEP for which subsets are considered as conditioning sets in the conditional independence tests. See pcalg for more details.
rules	Logical vector of length 10 indicating which rules should be used when directing edges. See pcalg for more details.
doPdsep	If TRUE, Possible-D-SEP is computed for all nodes, and all subsets of Possible-D-SEP are considered as conditioning sets in the conditional independence tests, if not defined otherwise in pdsep.max. If FALSE, Possible-D-SEP is not computed, so that the algorithm simplifies to the Modified PC algorithm of Spirtes, Glymour and Scheines (2000, p.84).
biCC	If TRUE, only nodes on paths between nodes x and y are considered to be in Possible-D-SEP(x) when testing independence between x and y. Uses biconnected components, biConnComp from RBGL.

conservative	Logical indicating if the unshielded triples should be checked for ambiguity the second time when v-structures are determined.
maj.rule	Logical indicating if the unshielded triples should be checked for ambiguity the second time when v-structures are determined using a majority rule idea, which is less strict than the standard conservative. For more information, see details.
verbose	If true, more detailed output is provided.

### Value

An object of class `fciAlgo` (see `fciAlgo` in the `pcalg` package) containing the estimated graph (in the form of an adjacency matrix with various possible edge marks), the conditioning sets that lead to edge removals (`sepset`) and several other parameters.

### References

1. Diego Colombo, Marloes H Maathuis, Markus Kalisch, Thomas S Richardson, et al. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, 40(1):294-321, 2012.
2. Markus Kalisch, Martin Machler, Diego Colombo, Marloes H Maathuis, and Peter Buhlmann. Causal inference using graphical models with the r package `pcalg`. *Journal of Statistical Software*, 47(11):1-26, 2012.

### Examples

```
#####
## Using fci_stable
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
fci_stable(suffStat, indepTest=gaussCItest, p=p, skel.method="stable", alpha=0.01)
```

---

IDA\_parallel

*Estimate Total Causal Effects using the IDA\_parallel Algorithm*

---

### Description

This is the parallelised version of the IDA (stable) algorithm in the `pcalg` package.

### Usage

```
IDA_parallel(datacsv, cause, effect, pcmethod, alpha, num.cores,
  mem.efficient = FALSE)
```

**Arguments**

datacsv	The dataset in csv format.
cause	The number of integer positions of the cause variables in the dataset.
effect	The number of integer positions of the target variables in the dataset.
pcmethod	Character string specifying method; the default, "parallel", will use the parallelised method for learning the skeleton of the graph, see <code>skeleton_parallel</code> .
alpha	significance level (number in (0; 1) for the individual conditional independence tests.
num.cores	The numbers of cores CPU to run the algorithm
mem.efficient	If TRUE, uses less amount of memory at any time point while running the algorithm

**Value**

A matrix that shows the causal effects (minimum of all possible effects) of the causes (columns) on the effects (rows)

**References**

Marloes H Maathuis, Markus Kalisch, Peter Buhlmann, et al. Estimating high-dimensional intervention effects from observational data. *The Annals of Statistics*, 37(6A):3133-3164,2009.

**Examples**

```
#####
## Using IDA_parallel without mem.efficeient
#####
library(bnlearn)
library(pcalg)
library(parallel)
data("gmI")
datacsv <- cov(gmI$x)
IDA_parallel(datacsv,1:2,3:4,"parallel",0.01, 2)

#####
## Using IDA_parallel with mem.efficeient
#####
library(bnlearn)
library(pcalg)
library(parallel)
data("gmI")
datacsv <- cov(gmI$x)
IDA_parallel(datacsv,1:2,3:4,"parallel",0.01, 2, TRUE)
```



---

IDA_stable	<i>Estimate Total Causal Effects</i>
------------	--------------------------------------

---

**Description**

This the stable version (using stable-PC for structure learning) of the IDA algorithm in the pcalg package.

**Usage**

```
IDA_stable(datacsv, cause, effect, pmethod, alpha)
```

**Arguments**

datacsv	The dataset in csv format with rows are samples and columns are variables
cause	The number of integer positions of the cause variables in the dataset
effect	The number of integer positions of the target variables in the dataset.
pmethod	Character string specifying method; the default, "stable", provides an order-independent skeleton. See Colombo, 2014.
alpha	significance level (number in (0; 1) for the individual conditional independence tests.

**Value**

A matrix that shows the causal effects (minimum of all possible effects) of the causes (columns) on the effects (rows).

**References**

1. Marloes H Maathuis, Markus Kalisch, Peter Buhlmann, et al. Estimating high-dimensional intervention effects from observational data. *The Annals of Statistics*, 37(6A):3133-3164,2009.
2. Diego Colombo and Marloes H Maathuis. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research*, 15(1):3741-3782, 2014.

**Examples**

```
#####
## Using IDA_stable
#####
library(pcalg)
data("gmI")
datacsv <- cov(gmI$x)
IDA_stable(datacsv,1:2,3:4,"stable",0.01)
```

---

jointIDA_direct	<i>Estimate Total Causal Effects of Joint Interventions</i>
-----------------	---

---

### Description

This is the parallelised version of the jointIDA (stable) algorithm in the pcalg package.

### Usage

```
jointIDA_direct(datacsv, cause, effect, method = c("min", "max", "median"),
  pcmethod = "stable", alpha, num.cores = 1, mem.efficient = FALSE,
  technique = c("RRC", "MCD"))
```

### Arguments

datacsv	The dataset in the csv format with rows are samples and columns are the variables.
cause	The number of integer positions of the intervention variables in the dataset.
effect	the integer position of the target variable in the dataset.
method	the method of calculating the final effect from multiple possible effects, e.g. min, max, median
pcmethod	Character string specifying the method of the PC algorithm, e.g. stable for stable-PC, and parallel for parallel-PC.
alpha	significance level (number in (0; 1) for the conditional independence tests.
num.cores	The numbers of cores CPU to run the algorithm
mem.efficient	If TRUE, uses less amount of memory at any time point while running the algorithm
technique	The character string specifying the technique that will be used to estimate the total joint causal effects in the pcalg package. RRC for Recursive regression for causal effects MCD for Modifying the Cholesky decomposition

### Value

A matrix that shows the direct causal effects (minimum of all possible effects) of the (first) cause (columns) on the effects (rows)

---

jointIDA\_parallel      *Estimate Total Causal Effects of Joint Interventions*

---

## Description

This is the parallelised version of the IDA (stable) algorithm in the pcalg package.

## Usage

```
jointIDA_parallel(datacsv, cause, effect, pcmethod = "stable", alpha,
  num.cores = 1, mem.efficient = FALSE, technique = c("RRC", "MCD"))
```

## Arguments

datacsv	The dataset in csv format with rows are samples and columns are variables.
cause	The number of integer positions of the intervention variables in the dataset.
effect	the integer position of the target variable in the dataset.
pcmethod	Character string specifying the method of the PC algorithm, e.g. stable for stable-PC, and parallel for parallel-PC.
alpha	significance level (number in (0; 1) for the conditional independence tests.
num.cores	The numbers of cores CPU to run the algorithm
mem.efficient	If TRUE, uses less amount of memory at any time point while running the algorithm
technique	The character string specifying the technique that will be used to estimate the total joint causal effects in the pcalg package. RRC for Recursive regression for causal effects MCD for Modifying the Cholesky decomposition

## Value

A matrix that shows the causal effects of the causes (rows) on the effect. Different columns show different possible causal effect values.

## Examples

```
#####
## Using IDA_parallel without mem.efficeient
#####
library(bnlearn)
library(pcalg)
library(parallel)
data("gmI")
datacsv <- cov(gmI$x)
jointIDA_parallel(datacsv,1:2,3, pcmethod="parallel",0.01, 2, technique="RRC")

#####
## Using IDA_parallel with mem.efficeient
```

```
#####
library(bnlearn)
library(pcalg)
library(parallel)
data("gmI")
datacsv <- cov(gmI$x)
jointIDA_parallel(datacsv,1:2,3, pcmethod="parallel",0.01, 2, TRUE, technique="RRC")
```

---

mccor

*The Monte Carlo permutation test (mc-cor)*


---

### Description

The Monte Carlo permutation test for Pearson's chi-square. See bnlearn package for details.

### Usage

```
mccor(x, y, S, suffStat)
```

### Arguments

<code>x, y, S</code>	It is tested, whether <code>x</code> and <code>y</code> are conditionally independent given the subset <code>S</code> of the remaining nodes. <code>x</code> , <code>y</code> , <code>S</code> all are integers, corresponding to variable or node numbers.
<code>suffStat</code>	The dataset in matrix format with rows are samples and columns are variables.

### Value

The p-value of the test.

### References

Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

### Examples

```
#####
## Using mccor
#####
library(bnlearn)
library(pcalg)
data("gmG")
suffStat<-gmG$x
mccor(1,2,3,suffStat)
```

---

`mcmig`*The Monte Carlo permutation test (mc-mi-g)*

---

**Description**

The Monte Carlo permutation test for mutual information. See bnlearn package for more details.

**Usage**

```
mcmig(x, y, S, suffStat)
```

**Arguments**

<code>x, y, S</code>	It is tested, whether <code>x</code> and <code>y</code> are conditionally independent given the subset <code>S</code> of the remaining nodes. <code>x</code> , <code>y</code> , <code>S</code> all are integers, corresponding to variable or node numbers.
<code>suffStat</code>	The dataset in matrix format with rows are samples and columns are variables.

**Value**

the p-value of the test.

**References**

Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

**Examples**

```
#####  
## Using mcmig  
#####  
library(bnlearn)  
library(pcalg)  
data("gmG")  
suffStat<-gmG$x  
mcmig(1,2,3,suffStat)
```

---

mcpf	<i>The Monte Carlo permutation test for Gaussian conditional independence test</i>
------	--

---

### Description

The Monte Carlo permutation test for Gaussian conditional independence test. See the mc-zf function in the bnlearn package for more details.

### Usage

```
mcpf(x, y, S, suffStat)
```

### Arguments

<code>x, y, S</code>	It is tested, whether <code>x</code> and <code>y</code> are conditionally independent given the subset <code>S</code> of the remaining nodes. <code>x</code> , <code>y</code> , <code>S</code> all are integers, corresponding to variable or node numbers.
<code>suffStat</code>	The dataset in matrix format with rows are samples and columns are variables.

### Value

the p-value of the test.

### References

Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

### Examples

```
#####
## Using mcpf
#####
library(bnlearn)
library(pcalg)
data("gmG")
suffStat<-gmG$x
mcpf(1,2,3,suffStat)
```

---

mig	<i>Mutual information test</i>
-----	--------------------------------

---

## Description

Mutual information test. See function `mi-g` in `bnlearn` package for more details.

## Usage

```
mig(x, y, S, suffStat)
```

## Arguments

<code>x, y, S</code>	It is tested, whether <code>x</code> and <code>y</code> are conditionally independent given the subset <code>S</code> of the remaining nodes. <code>x</code> , <code>y</code> , <code>S</code> all are integers, corresponding to variable or node numbers.
<code>suffStat</code>	The dataset in matrix format with rows are samples and columns are variables.

## Value

The p-value of the test.

## References

Marco Scutari (2010). Learning Bayesian Networks with the `bnlearn` R Package. *Journal of Statistical Software*, 35(3), 1-22.

## Examples

```
#####  
## Using mig  
#####  
library(bnlearn)  
library(pcalg)  
data("gmG")  
suffStat<-gmG$x  
mig(1,2,3,suffStat)
```

---

 migsh

*Shrinkage estimator for the mutual information (mi-g-sh)*


---

**Description**

Shrinkage estimator for the mutual information. See bnlearn package for more details.

**Usage**

```
migsh(x, y, S, suffStat)
```

**Arguments**

<code>x, y, S</code>	It is tested, whether <code>x</code> and <code>y</code> are conditionally independent given the subset <code>S</code> of the remaining nodes. <code>x</code> , <code>y</code> , <code>S</code> all are integers, corresponding to variable or node numbers.
<code>suffStat</code>	The dataset in matrix format with rows are samples and columns are variables.

**Value**

The p-value of the test.

**References**

Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

**Examples**

```
#####
## Using migsh
#####
library(bnlearn)
library(pcalg)
data("gmG")
suffStat<-gmG$x
migsh(1,2,3,suffStat)
```



---

pcSelect_parallel	<i>Estimate subgraph around a response variable using pcSelect_parallel.</i>
-------------------	--

---

### Description

This is the parallelised version of the pcSelect (stable) function in the pcalg package. Assume that we have a fixed target variable, the algorithm will test the dependency between each variable and the target variable conditioning on combinations of other variables.

### Usage

```
pcSelect_parallel(y, dm, method = c("parallel"), mem.efficient = FALSE,
  num_workers, alpha, corMethod = "standard", verbose = FALSE,
  directed = FALSE)
```

### Arguments

y	The target (response) variable.
dm	Data matrix with rows are samples and columns are variables.
method	Character string specifying method; the default, "parallel" provides an parallelised method to implement all the conditional independence tests.
mem.efficient	If TRUE, uses less amount of memory at any time point while running the algorithm
num_workers	The numbers of cores CPU to run the algorithm
alpha	Significance level of individual partial correlation tests.
corMethod	"standard" or "Qn" for standard or robust correlation estimation
verbose	Logical or in {0,1,2}; FALSE, 0: No output, TRUE, 1: Little output, 2: Detailed output. Note that such output makes the function very much slower.
directed	Logical; should the output graph be directed?

### Value

G A logical vector indicating which column of dm is associated with y.

zMin The minimal z-values when testing partial correlations between y and each column of dm. The larger the number, the more consistent is the edge with the data.

**Examples**

```
#####
## Using pcSelect_parallel without mem.efficeient
#####
library(pcalg)
library(parallel)
p <- 10
set.seed(101)
myDAG <- randomDAG(p, prob = 0.2)
n <- 1000
d.mat <- rmvDAG(n, myDAG, errDist = "normal")
pcSelect_parallel(d.mat[,10],d.mat[,-10], alpha=0.05,num_workers=2)

#####
## Using pcSelelct_parallel with mem.efficeient
#####
library(pcalg)
library(parallel)
p <- 10
set.seed(101)
myDAG <- randomDAG(p, prob = 0.2)
n <- 1000
d.mat <- rmvDAG(n, myDAG, errDist = "normal")
pcSelect_parallel(d.mat[,10],d.mat[,-10], alpha=0.05,mem.efficient=TRUE,num_workers=2)
```

---

pcSelect\_stable

*Estimate subgraph around a response variable using pcSelect*


---

**Description**

This is the stable version (order independent version) of the pcSelect function (pc-Simple algorithm) in the pcalg package.

**Usage**

```
pcSelect_stable(y, dm, alpha, corMethod = "standard", method = "stable",
  verbose = FALSE, directed = FALSE)
```

**Arguments**

y	The target (response) variable.
dm	Data matrix with rows are samples and columns are variables.
alpha	Significance level of individual partial correlation tests.
corMethod	"standard" or "Qn" for standard or robust correlation estimation
method	Character string specifying method; the default, "stable" provides an Order-independent version.

verbose Logical or in {0,1,2};  
 FALSE, 0: No output,  
 TRUE, 1: Little output,  
 2: Detailed output.  
 Note that such output makes the function very much slower.

directed Logical; should the output graph be directed?

### Value

G A logical vector indicating which column of dm is associated with y.

zMin The minimal z-values when testing partial correlations between y and each column of dm.  
 The larger the number, the more consistent is the edge with the data.

### Examples

```
#####
## Using pcSelect_stable
#####
library(pcalg)
library(parallel)
p <- 10
set.seed(101)
myDAG <- randomDAG(p, prob = 0.2)
n <- 1000
d.mat <- rmvDAG(n, myDAG, errDist = "normal")
pcSelect_stable(d.mat[,10],d.mat[,-10], alpha=0.05)
```

---

pc_parallel	<i>Estimate the Equivalence Class of a DAG using the PC_parallel Algorithm</i>
-------------	--

---

### Description

Estimate the equivalence class of a directed acyclic graph (DAG) from observational data, using the PC\_parallel algorithm.

### Usage

```
pc_parallel(suffStat, indepTest, alpha, labels, p, fixedGaps = NULL,
  fixedEdges = NULL, NDelete = TRUE, m.max = Inf, u2pd = c("relaxed",
  "rand", "retry"), skel.method = c("parallel"), mem.efficient = FALSE,
  conservative = FALSE, maj.rule = FALSE, solve.confl = FALSE,
  verbose = FALSE, num.cores = detectCores())
```

**Arguments**

<code>suffStat</code>	A list of sufficient statistics, containing all necessary elements for the conditional independence decisions in the function <code>indepTest</code> .
<code>indepTest</code>	A function for testing conditional independence. It is internally called as <code>indepTest(x,y,S,suffStat)</code> , and tests conditional independence of $x$ and $y$ given $S$ . Here, $x$ and $y$ are variables, and $S$ is a (possibly empty) vector of variables (all variables are denoted by their column numbers in the adjacency matrix). <code>suffStat</code> is a list, see the argument above. The return value of <code>indepTest</code> is the p-value of the test for conditional independence.
<code>alpha</code>	significance level (number in (0,1) for the individual conditional independence tests.
<code>labels</code>	(optional) character vector of variable (or "node") names. Typically preferred to specifying <code>p</code> .
<code>p</code>	(optional) number of variables (or nodes). May be specified if <code>labels</code> are not, in which case <code>labels</code> is set to <code>1:p</code> .
<code>fixedGaps</code>	A logical matrix of dimension $p \times p$ . If entry <code>[i,j]</code> or <code>[j,i]</code> (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.
<code>fixedEdges</code>	A logical matrix of dimension $p \times p$ . If entry <code>[i,j]</code> or <code>[j,i]</code> (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.
<code>NAdelete</code>	If <code>indepTest</code> returns NA and this option is TRUE, the corresponding edge is deleted. If this option is FALSE, the edge is not deleted.
<code>m.max</code>	Maximal size of the conditioning sets that are considered in the conditional independence tests.
<code>u2pd</code>	String specifying the method for dealing with conflicting information when trying to orient edges (see <code>pcalg</code> for details).
<code>skel.method</code>	Character string specifying method; the default, "parallel", <code>skeleton_parallel</code> for learning the causal structure.
<code>mem.efficient</code>	If TRUE, uses less amount of memory at any time point while running the algorithm.
<code>conservative</code>	Logical indicating if the conservative PC is used. In this case, only option <code>u2pd = "relaxed"</code> is supported. Note that therefore the resulting object might not be extendable to a DAG. See <code>pcalg</code> for details.
<code>maj.rule</code>	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 <code>pcalg</code> .
<code>solve.confl</code>	If TRUE, the orientation of the $v$ -structures and the orientation rules work with lists for candidate sets and allow bi-directed edges to resolve conflicting edge orientations. See <code>pcalg</code> for details.
<code>verbose</code>	If TRUE, detailed output is provided.
<code>num.cores</code>	The numbers of cores CPU to run the algorithm.

**Value**

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

**Examples**

```
#####
## Using pc_parallel without mem.efficient
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
pc_parallel(suffStat, indepTest=gaussCItest, p=p, skel.method="parallel", alpha=0.01, num.cores=2)

#####
## Using pc_parallel with mem.efficient
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
pc_parallel(suffStat, indepTest=gaussCItest, p=p, skel.method="parallel",
alpha=0.01, num.cores=2, mem.efficient=TRUE)

#####
## Using pc_parallel with mutual information test
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
#The first parameter is the dataset rather than suffStat
pc_parallel(gmG$x, indepTest=mig, p=p, skel.method="parallel",
alpha=0.01, num.cores=2, mem.efficient=TRUE)
```

---

pc\_stable

---

*Estimate the Equivalence Class of a DAG using the PC\_stable Algorithm*


---

**Description**

Estimate the equivalence class of a directed acyclic graph (DAG) from observational data, using the PC\_stable algorithm.

**Usage**

```
pc_stable(suffStat, indepTest, alpha, labels, p, fixedGaps = NULL,
  fixedEdges = NULL, NAdelate = TRUE, m.max = Inf, u2pd = c("relaxed",
  "rand", "retry"), skel.method = c("stable", "original", "stable.fast"),
  conservative = FALSE, maj.rule = FALSE, solve.confl = FALSE,
  verbose = FALSE)
```

**Arguments**

suffStat	A list of sufficient statistics, containing all necessary elements for the conditional independence decisions in the function indepTest.
indepTest	A function for testing conditional independence. It is internally called as indepTest(x,y,S,suffStat), and tests conditional independence of x and y given S. Here, x and y are variables, and S is a (possibly empty) vector of variables (all variables are denoted by their column numbers in the adjacency matrix). suffStat is a list, see the argument above. The return value of indepTest is the p-value of the test for conditional independence.
alpha	significance level (number in (0,1) for the individual 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.
NAdelate	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 pcalg for details).
skel.method	Character string specifying method; the default, "stable" provides an order-independent skeleton.
conservative	Logical indicating if the conservative PC is used. In this case, only option u2pd = "relaxed" is supported. See pcalg for more information.
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 the pcalg package.
solve.confl	If TRUE, the orientation of the v-structures and the orientation rules work with lists for candidate sets and allow bi-directed edges to resolve conflicting edge

orientations. In this case, only option `u2pd = relaxed` is supported. Note, that therefore the resulting object might not be a CPDAG because bi-directed edges might be present. See details for more information.

`verbose` If TRUE, detailed output is provided.

### Value

An object of class "pcAlgo" (see `pcAlgo` in the `pcalg` package) containing an estimate of the equivalence class of the underlying DAG.

### Examples

```
#####
## Using pc_stable
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
pc_stable(suffStat, indepTest=gaussCItest, p=p, skel.method="stable", alpha=0.01)
```

---

rfci\_parallel

*Estimate a PAG fast using the RFCI\_parallel Algorithm*

---

### Description

This is the parallelised version of the RFCI algorithm in the `pcalg` package.

### Usage

```
rfci_parallel(suffStat, indepTest, alpha, labels, p,
  skel.method = c("parallel"), mem.efficient = FALSE, fixedGaps = NULL,
  fixedEdges = NULL, NDelete = TRUE, m.max = Inf, rules = rep(TRUE,
  10), conservative = FALSE, maj.rule = FALSE, verbose = FALSE,
  num.cores = detectCores())
```

### Arguments

`suffStat` Sufficient statistics: List containing all necessary elements for the conditional independence decisions in the function `indepTest`.

`indepTest` Predefined function for testing conditional independence. The function is internally called as `indepTest(x,y,S,suffStat)`, and tests conditional independence of `x` and `y` given `S`. Here, `x` and `y` are variables, and `S` is a (possibly empty) vector of variables (all variables are denoted by their column numbers in the adjacency matrix). `suffStat` is a list containing all relevant elements for the conditional independence decisions. The return value of `indepTest` is the p-value of the test for conditional independence.

alpha	Significance level for the individual 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, "parallel" provides an efficient skeleton, see skeleton_parallel.
mem.efficient	Uses less amount of memory at any time point while running the algorithm
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	Maximum size of the conditioning sets that are considered in the conditional independence tests.
rules	Logical vector of length 10 indicating which rules should be used when directing edges. See the pcalg package for details.
conservative	Logical indicating if the unshielded triples should be checked for ambiguity the second time when v-structures are determined. For more information, see pcalg.
maj.rule	Logical indicating if the unshielded triples should be checked for ambiguity the second time when v-structures are determined using a majority rule idea, which is less strict than the standard conservative. For more information, see pcalg.
verbose	If true, more detailed output is provided.
num.cores	The numbers of cores CPU to run the algorithm

### Value

An object of class fciAlgo (see fciAlgo in the pcalg package) containing the estimated graph (in the form of an adjacency matrix with various possible edge marks), the conditioning sets that lead to edge removals (sepset) and several other parameters.

### References

1. Diego Colombo, Marloes H Maathuis, Markus Kalisch, Thomas S Richardson, et al. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, 40(1):294-321, 2012.
2. Markus Kalisch, Martin Machler, Diego Colombo, Marloes H Maathuis, and Peter Buhlmann. Causal inference using graphical models with the r package pcalg. *Journal of Statistical Software*, 47(11):1-26, 2012.



**Examples**

```
#####
## Using rfci_parallel without mem.efficient
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
rfci_parallel(suffStat, indepTest=gaussCItest, p=p, skel.method="parallel", alpha=0.01, num.cores=2)

#####
## Using rfci_parallel with mem.efficient
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
rfci_parallel(suffStat, indepTest=gaussCItest, p=p, skel.method="parallel",
alpha=0.01, num.cores=2, mem.efficient=TRUE)

#####
## Using fci_parallel with mutual information test
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)

# The first parameter is the dataset
rfci_parallel(gmG$x, indepTest=mig, p=p, skel.method="parallel",
alpha=0.01, num.cores=2, mem.efficient=TRUE)
```

---

rfci\_stable

---

*Estimate a PAG using the RFCI\_stable Algorithm*


---

**Description**

This is the RFCI stable version in the pcalg package.

**Usage**

```
rfci_stable(suffStat, indepTest, alpha, labels, p, skel.method = c("stable",
"original", "stable.fast"), fixedGaps = NULL, fixedEdges = NULL,
NAdelete = TRUE, m.max = Inf, rules = rep(TRUE, 10),
conservative = FALSE, maj.rule = FALSE, verbose = FALSE)
```

**Arguments**

suffStat	Sufficient statistics: List containing all necessary elements for the conditional independence decisions in the function indepTest.
indepTest	Predefined function for testing conditional independence. The function is internally called as indepTest(x,y,S,suffStat), and tests conditional independence of x and y given S. Here, x and y are variables, and S is a (possibly empty) vector of variables (all variables are denoted by their column numbers in the adjacency matrix). suffStat is a list containing all relevant elements for the conditional independence decisions. The return value of indepTest is the p-value of the test for conditional independence.
alpha	significance level (number in (0,1) for the individual 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 skeleton.
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	Maximum size of the conditioning sets that are considered in the conditional independence tests.
rules	Logical vector of length 10 indicating which rules should be used when directing edges. See the pcalg package for details.
conservative	Logical indicating if the unshielded triples should be checked for ambiguity after the skeleton has been found, similar to the conservative PC algorithm.
maj.rule	Logical indicating if the unshielded triples should be checked for ambiguity after the skeleton has been found using a majority rule idea, which is less strict than the conservative.
verbose	If true, more detailed output is provided.

**Value**

An object of class fciAlgo (see fciAlgo in the pcalg package) containing the estimated graph (in the form of an adjacency matrix with various possible edge marks), the conditioning sets that lead to edge removals (sepset) and several other parameters.

## References

1. Diego Colombo, Marloes H Maathuis, Markus Kalisch, Thomas S Richardson, et al. Learning high-dimensional directed acyclic graphs with latent and selection variables. *The Annals of Statistics*, 40(1):294-321, 2012.
2. Markus Kalisch, Martin Machler, Diego Colombo, Marloes H Maathuis, and Peter Buhlmann. Causal inference using graphical models with the r package pcalg. *Journal of Statistical Software*, 47(11):1-26, 2012.

## Examples

```
#####
## Using rfci_stable
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
rfci_stable(suffStat, indepTest=gaussCItest, p=p, skel.method="stable", alpha=0.01)
```

---

skeleton\_parallel      *Estimate (Initial) Skeleton of a DAG.*

---

## Description

This is the parallelised version of the skeleton function in the pcalg package.

## Usage

```
skeleton_parallel(suffStat, indepTest, alpha, labels, p,
  method = c("parallel"), mem.efficient = FALSE, workers, num_workers,
  m.max = Inf, fixedGaps = NULL, fixedEdges = NULL, NAdelete = TRUE,
  verbose = FALSE)
```

## Arguments

suffStat	Sufficient statistics: List containing all necessary elements for the conditional independence decisions in the function indepTest.
indepTest	Predefined function for testing conditional independence. The function is internally called as indepTest(x,y,S,suffStat) and tests conditional independence of x and y given S. Here, x and y are variables, and S is a (possibly empty) vector of variables (all variables are denoted by their column numbers in the adjacency matrix). suffStat is a list containing all relevant elements for the conditional independence decisions. The return value of indepTest is the p-value of the test for conditional independence.
alpha	significance level (number in (0; 1) for the individual 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.
method	Character string specifying method; the default, "parallel" provides an efficient skeleton, see skeleton_parallel.
mem.efficient	Uses less amount of memory at any time point while running the algorithm
workers	Creates a set of copies of R running in parallel and communicating over sockets.
num_workers	The numbers of cores CPU as numbers of workers to run the algorithm
m.max	Maximal size of the conditioning sets that are considered in the conditional independence tests.
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	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.

### Value

An object of class "pcAlgo" (see pcAlgo in the pcalg package) containing an estimate of the skeleton of the underlying DAG, the conditioning sets (sepset) that led to edge removals and several other parameters.

### Examples

```
#####
## Using skeleton_parallel without mem.efficeient
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
skeleton_parallel(suffStat,indepTest=gaussCItest,p=p,method="parallel",alpha=0.01,num_workers=2)

#####
## Using skeleton_parallel with mem.efficeient
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
skeleton_parallel(suffStat,indepTest=gaussCItest,p=p,method="parallel",
alpha=0.01,num_workers=2,mem.efficient=TRUE)
```

---

skeleton\_stable      *Estimate (Initial) Skeleton of a DAG using the PC\_stable Algorithm*

---

### Description

This is the skeleton (stable) function in the pcalg package. It is copied here to localise the parallel functions.

### Usage

```
skeleton_stable(suffStat, indepTest, alpha, labels, p, method = c("stable",
  "original", "stable.fast"), m.max = Inf, fixedGaps = NULL,
  fixedEdges = NULL, NAdelete = TRUE, verbose = FALSE)
```

### Arguments

suffStat	Sufficient statistics: List containing all necessary elements for the conditional independence decisions in the function indepTest.
indepTest	Predefined function for testing conditional independence. The function is internally called as indepTest(x,y,S,suffStat) and tests conditional independence of x and y given S. Here, x and y are variables, and S is a (possibly empty) vector of variables (all variables are denoted by their column numbers in the adjacency matrix). suffStat is a list containing all relevant elements for the conditional independence decisions. The return value of indepTest is the p-value of the test for conditional independence.
alpha	significance level (number in (0,1) for the individual 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.
method	Character string specifying method; the default, "stable" provides an order-independent skeleton, see 'Details' below.
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.

**Value**

An object of class "pcAlgo" (see pcAlgo in the pcalg package) containing an estimate of the skeleton of the underlying DAG, the conditioning sets (sepset) that led to edge removals and several other parameters.

**Examples**

```
#####
## Using skeleton_stable
#####
library(pcalg)
library(parallel)
data("gmG")
p<-ncol(gmG$x)
suffStat<-list(C=cor(gmG$x),n=nrow(gmG$x))
skeleton_stable(suffStat, indepTest=gaussCItest, p=p, method="stable", alpha=0.01)
```

---

smccor

---

*The sequential Monte Carlo permutation test (smc-cor)*


---

**Description**

The sequential Monte Carlo permutation test. See bnlearn package for details.

**Usage**

```
smccor(x, y, S, suffStat)
```

**Arguments**

x, y, S	It is tested, whether x and y are conditionally independent given the subset S of the remaining nodes. x, y, S all are integers, corresponding to variable or node numbers.
suffStat	The dataset in matrix format with rows are samples and columns are variables.

**Value**

The p-value of the test.

**References**

Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

**Examples**

```
#####
## Using smccor
#####
library(bnlearn)
library(pcalg)
data("gmG")
suffStat<-gmG$x
smccor(1,2,3,suffStat)
```

---

smcmig

*The sequential Monte Carlo permutation test (smc-mi-g)*


---

**Description**

The sequential Monte Carlo permutation test. See bnlearn package for more details.

**Usage**

```
smcmig(x, y, S, suffStat)
```

**Arguments**

<code>x, y, S</code>	It is tested, whether <code>x</code> and <code>y</code> are conditionally independent given the subset <code>S</code> of the remaining nodes. <code>x</code> , <code>y</code> , <code>S</code> all are integers, corresponding to variable or node numbers.
<code>suffStat</code>	The data matrix with rows are samples and columns are variables.

**Value**

The p-value of the test.

**References**

Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

**Examples**

```
#####
## Using smcmig
#####
library(bnlearn)
library(pcalg)
data("gmG")
suffStat<-gmG$x
smcmig(1,2,3,suffStat)
```

---

smczf	<i>The sequential Monte Carlo permutation test for Gaussian conditional independence test.</i>
-------	--

---

### Description

The sequential Monte Carlo permutation test for Gaussian conditional independence test. See the smc-zf function in the bnlearn package for more details.

### Usage

```
smczf(x, y, S, suffStat)
```

### Arguments

x, y, S	It is tested, whether x and y are conditionally independent given the subset S of the remaining nodes. x, y, S all are integers, corresponding to variable or node numbers.
suffStat	The data matrix with rows are samples and columns are variables.

### Value

The p-value of the test.

### References

Marco Scutari (2010). Learning Bayesian Networks with the bnlearn R Package. Journal of Statistical Software, 35(3), 1-22.

### Examples

```
#####
## Using smczf
#####
library(bnlearn)
library(pcalg)
data("gmG")
suffStat<-gmG$x
smczf(1,2,3,suffStat)
```



---

zf *Gaussian conditional independence test*

---

**Description**

Gaussian conditional independence test. See the *zf* function in the *bnlearn* package for more details.

**Usage**

```
zf(x, y, S, suffStat)
```

**Arguments**

<code>x, y, S</code>	It is tested, whether <i>x</i> and <i>y</i> are conditionally independent given the subset <i>S</i> of the remaining nodes. <i>x</i> , <i>y</i> , <i>S</i> all are integers, corresponding to variable or node numbers.
<code>suffStat</code>	the data matrix with rows are samples and columns are the variables.

**Value**

The p-value of the test.

**References**

Marco Scutari (2010). Learning Bayesian Networks with the *bnlearn* R Package. *Journal of Statistical Software*, 35(3), 1-22.

**Examples**

```
#####
## Using zf
#####
library(bnlearn)
library(pcalg)
data("gmG")
suffStat<-gmG$x
zf(1,2,3,suffStat)
```

# Index

[cor2](#), [2](#)

[fci\\_parallel](#), [3](#)

[fci\\_stable](#), [5](#)

[IDA\\_parallel](#), [7](#)

[IDA\\_stable](#), [9](#)

[jointIDA\\_direct](#), [10](#)

[jointIDA\\_parallel](#), [11](#)

[mccor](#), [12](#)

[mcmig](#), [13](#)

[mczf](#), [14](#)

[mig](#), [15](#)

[migsh](#), [16](#)

[pc\\_parallel](#), [19](#)

[pc\\_stable](#), [21](#)

[pcSelect\\_parallel](#), [17](#)

[pcSelect\\_stable](#), [18](#)

[rfci\\_parallel](#), [23](#)

[rfci\\_stable](#), [25](#)

[skeleton\\_parallel](#), [27](#)

[skeleton\\_stable](#), [29](#)

[smccor](#), [30](#)

[smcmig](#), [31](#)

[smczf](#), [32](#)

[zf](#), [33](#)