Package 'seqICP'

October 14, 2022

Title Sequential Invariant Causal Prediction

Version 1.1

Author Niklas Pfister and Jonas Peters

Maintainer Niklas Pfister <pfister@stat.math.ethz.ch>

Description Contains an implementation of invariant causal prediction for sequential data. The main function in the package is 'seqICP', which performs linear sequential invariant causal prediction and has guaranteed type I error control. For non-linear dependencies the package also contains a non-linear method 'seqICPnl', which allows to input any regression procedure and performs tests based on a permutation approach that is only approximately correct. In order to test whether an individual set S is invariant the package contains the subroutines 'seqICP.s' and 'seqICPnl.s' corresponding to the respective main methods.

Depends R (>= 3.2.3)

License GPL-3

Encoding UTF-8

LazyData true

Imports dHSIC, mgcv, stats

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-25 14:54:20 UTC

R topics documented:

`seqICP-package` *Sequential Invariant Causal Prediction*

Description

Contains an implementation of invariant causal prediction for sequential data. The main function in the package is 'seqICP', which performs linear sequential invariant causal prediction and has guaranteed type I error control. For non-linear dependencies the package also contains a non-linear method 'seqICPnl', which allows to input any regression procedure and performs tests based on a permutation approach that is only approximately correct. In order to test whether an individual set S is invariant the package contains the subroutines 'seqICP.s' and 'seqICPnl.s' corresponding to the respective main methods.

Details

The DESCRIPTION file:

Package:	seqICP
Title:	Sequential Invariant Causal Prediction
Version:	1.1
Author:	Niklas Pfister and Jonas Peters
Maintainer:	Niklas Pfister <pfister@stat.math.ethz.ch>
Description:	Contains an implementation of invariant causal prediction for sequential data. The main function in the packs
Depends:	R (>= 3.2.3)
License:	GPL-3
Encoding:	UTF-8
LazyData:	true
Imports:	dHSIC, mgcv, stats
RoxygenNote:	6.0.1

Index of help topics:

```
seqICP                  Sequential Invariant Causal Prediction
seqICP-package          Sequential Invariant Causal Prediction
seqICP.s                Sequential Invariant Causal Prediction for an
                        individual set S
seqICPnl                Non-linear Invariant Causal Prediction
seqICPnl.s              Non-linear Invariant Causal Prediction for an
                        individual set S
summary.seqICP          summary function
summary.seqICPnl        summary function
```

Author(s)

Niklas Pfister and Jonas Peters

Maintainer: Niklas Pfister <pfister@stat.math.ethz.ch>

References

Pfister, N., P. Bühlmann and J. Peters (2017). Invariant Causal Prediction for Sequential Data. ArXiv e-prints (1706.08058).

Peters, J., P. Bühlmann, and N. Meinshausen (2016). Causal inference using invariant prediction: identification and confidence intervals. Journal of the Royal Statistical Society, Series B (with discussion) 78 (5), 947–1012.

`seqICP` *Sequential Invariant Causal Prediction*

Description

Estimates the causal parents S of the target variable Y using invariant causal prediction and fits a linear model of the form
Y = a X^S + N.

Usage

```
seqICP(X, Y, test = "decoupled", par.test = list(grid = c(0,
  round(nrow(X)/2), nrow(X)), complements = FALSE, link = sum, alpha = 0.05, B =
  100, permutation = FALSE), model = "iid", par.model = list(pknown = FALSE,
  p = 0, max.p = 10), max.parents = ncol(X), stopIfEmpty = TRUE,
  silent = TRUE)
```

Arguments

`X`	matrix of predictor variables. Each column corresponds to one predictor variable.
`Y`	vector of target variable, with length(Y)=nrow(X).
`test`	string specifying the hypothesis test used to test for invariance of a parent set S (i.e. the null hypothesis H0_S). The following tests are available: "decoupled", "combined", "trend", "variance", "block.mean", "block.variance", "block.decoupled", "smooth.mean", "smooth.variance", "smooth.decoupled" and "hsic".
`par.test`	parameters specifying hypothesis test. The following parameters are available: `grid`, `complements`, `link`, `alpha`, `B` and `permutation`. The parameter `grid` is an increasing vector of gridpoints used to construct enviornments for change point based tests. If the parameter `complements` is 'TRUE' each environment is compared against its complement if it is 'FALSE' all environments are compared pairwise. The parameter `link` specifies how to compare the pairwise test statistics, generally this is either max or sum. The parameter `alpha` is a numeric value in (0,1) indicting the significance level of the hypothesis test. The parameter `B` is an integer and specifies the number of Monte-Carlo samples (or permutations) used in the approximation of the null distribution. If the parameter `permutation` is 'TRUE' a permuatation based approach is used to approximate the null distribution, if it is 'FALSE' the scaled residuals approach is used.

model	string specifying the underlying model class. Either "iid" if Y consists of independent observations or "ar" if Y has a linear time dependence structure.
par.model	parameters specifying model. The following parameters are available: pknown, p and max.p. If pknown is 'FALSE' the number of lags will be determined by comparing all fits up to max.p lags using the AIC criterion. If pknown is 'TRUE' the procedure will fit p lags.
max.parents	integer specifying the maximum size for admissible parents. Reducing this below the number of predictor variables saves computational time but means that the confidence intervals lose their coverage property.
stopIfEmpty	if 'TRUE', the procedure will stop computing confidence intervals if the empty set has been accepted (and hence no variable can have a signicificant causal effect). Setting to 'TRUE' will save computational time in these cases, but means that the confidence intervals lose their coverage properties for values different to 0.
silent	If 'FALSE', the procedure will output progress notifications consisting of the currently computed set S together with the p-value resulting from the null hypothesis H0_S

Details

The function can be applied to two types of models
(1) a linear model (model="iid")
Y_i = a X_i^S + N_i
with iid noise N_i and
(2) a linear autoregressive model (model="ar")
Y_t = a_0 X_t^S + ... + a_p (Y_(t-p),X_(t-p)) + N_t
with iid noise N_t.

For both models the invariant prediction procedure is applied using the hypothesis test specified by the test parameter to determine whether a candidate model is invariant. For further details see the references.

Value

object of class 'seqICP' consisting of the following elements

parent.set	vector of the estimated causal parents.
test.results	matrix containing the result from each individual test as rows.
S	list of all the sets that were tested. The position within the list corresponds to the index in the first column of the test.results matrix.
p.values	p-value for being not included in the set of true causal parents. (If a p-value is smaller than alpha, the corresponding variable is a member of parent.set.)
coefficients	vector of coefficients resulting from a regression based on the estimated parent set.
stopIfEmpty	a boolean value indicating whether computations stop as soon as intersection of accepted sets is empty.

modelReject	a boolean value indicating if the whole model was rejected (the p-value of the best fitting model is too low).
pknown	a boolean value indicating whether the number of lags in the model was known. Only relevant if model was set to "ar".
alpha	significance level at which the hypothesis tests were performed.
n.var	number of predictor variables.
model	either "iid" or "ar" depending on which model was selected.

Author(s)

Niklas Pfister and Jonas Peters

References

Pfister, N., P. Bühlmann and J. Peters (2017). Invariant Causal Prediction for Sequential Data. ArXiv e-prints (1706.08058).

Peters, J., P. Bühlmann, and N. Meinshausen (2016). Causal inference using invariant prediction: identification and confidence intervals. Journal of the Royal Statistical Society, Series B (with discussion) 78 (5), 947–1012.

See Also

The function `seqICP.s` allows to perform hypothesis test for individual sets S. For non-linear models the functions `seqICPnl` and `seqICPnl.s` can be used.

Examples

```
set.seed(1)

# environment 1
na <- 140
X1a <- 0.3*rnorm(na)
X3a <- X1a + 0.2*rnorm(na)
Ya <- -.7*X1a + .6*X3a + 0.1*rnorm(na)
X2a <- -0.5*Ya + 0.5*X3a + 0.1*rnorm(na)

# environment 2
nb <- 80
X1b <- 0.3*rnorm(nb)
X3b <- 0.5*rnorm(nb)
Yb <- -.7*X1b + .6*X3b + 0.1*rnorm(nb)
X2b <- -0.5*Yb + 0.5*X3b + 0.1*rnorm(nb)

# combine environments
X1 <- c(X1a,X1b)
X2 <- c(X2a,X2b)
X3 <- c(X3a,X3b)
Y <- c(Ya,Yb)
Xmatrix <- cbind(X1, X2, X3)
```

```
# Y follows the same structural assignment in both environments
# a and b (cf. the lines Ya <- ... and Yb <- ...).
# The direct causes of Y are X1 and X3.
# A linear model considers X1, X2 and X3 as significant.
# All these variables are helpful for the prediction of Y.
summary(lm(Y~Xmatrix))

# apply seqICP to the same setting
seqICP.result <- seqICP(X = Xmatrix, Y,
par.test = list(grid = seq(0, na + nb, (na + nb)/10), complements = FALSE, link = sum,
alpha = 0.05, B =100), max.parents = 4, stopIfEmpty=FALSE, silent=FALSE)
summary(seqICP.result)
# seqICP is able to infer that X1 and X3 are causes of Y
```

`seqICP.s` *Sequential Invariant Causal Prediction for an individual set S*

Description

Tests whether the conditional distribution of Y given X^S is invariant across time, by assuming a linear dependence model.

Usage

```
seqICP.s(X, Y, S, test = "decoupled", par.test = list(grid = c(0,
  round(nrow(X)/2), nrow(X)), complements = FALSE, link = sum, alpha = 0.05, B =
  100, permutation = FALSE), model = "iid", par.model = list(pknown = FALSE,
  p = 0, max.p = 10))
```

Arguments

`X`	matrix of predictor variables. Each column corresponds to one predictor variable.
`Y`	vector of target variable, with length(Y)=nrow(X).
`S`	vector containing the indicies of predictors to be tested
`test`	string specifying the hypothesis test used to test for invariance of a parent set S (i.e. the null hypothesis H0_S). The following tests are available: "decoupled", "combined", "trend", "variance", "block.mean", "block.variance", "block.decoupled", "smooth.mean", "smooth.variance", "smooth.decoupled" and "hsic".
`par.test`	parameters specifying hypothesis test. The following parameters are available: `grid`, `complements`, `link`, `alpha`, `B` and `permutation`. The parameter `grid` is an increasing vector of gridpoints used to construct enviornments for change point based tests. If the parameter `complements` is 'TRUE' each environment is compared against its complement if it is 'FALSE' all environments are compared pairwise. The parameter `link` specifies how to compare the pairwise test statistics, generally this is either max or sum. The parameter `alpha` is a numeric value in (0,1) indicting the significance level of the hypothesis test. The

parameter B is an integer and specifies the number of Monte-Carlo samples (or permutations) used in the approximation of the null distribution. If the parameter permutation is 'TRUE' a permuatation based approach is used to approximate the null distribution, if it is 'FALSE' the scaled residuals approach is used.

`model` string specifying the underlying model class. Either "iid" if Y consists of independent observations or "ar" if Y has a linear time dependence structure.

`par.model` parameters specifying model. The following parameters are available: pknown, p and max.p. If pknown is 'FALSE' the number of lags will be determined by comparing all fits up to max.p lags using the AIC criterion. If pknown is 'TRUE' the procedure will fit p lags.

Details

The function can be applied to two types of models
(1) a linear model (model="iid")
Y_i = a X_i^S + N_i
with iid noise N_i and
(2) a linear autoregressive model (model="ar")
Y_t = a_0 X_t^S + ... + a_p (Y_(t-p),X_(t-p)) + N_t
with iid noise N_t.

For both models the hypothesis test specified by the test parameter is used to test whether the set S leads to an invariant model. For futher details see the references.

Value

list containing the following elements

`test.stat` value of the test statistic.

`crit.value` critical value computed using a Monte-Carlo simulation of the null distribution.

`p.value` p-value.

`p` number of lags that were used.

`model.fit` 'lm' object of linear model fit.

Author(s)

Niklas Pfister and Jonas Peters

References

Pfister, N., P. Bühlmann and J. Peters (2017). Invariant Causal Prediction for Sequential Data. ArXiv e-prints (1706.08058).

Peters, J., P. Bühlmann, and N. Meinshausen (2016). Causal inference using invariant prediction: identification and confidence intervals. Journal of the Royal Statistical Society, Series B (with discussion) 78 (5), 947–1012.

See Also

To estimate the set of causal parents use the function seqICP. For non-linear models use the corresponding functions seqICPnl and seqICPnl.s.

Examples

```
set.seed(1)

# environment 1
na <- 130
X1a <- rnorm(na,0,0.1)
Ya <- 5*X1a+rnorm(na,0,0.5)
X2a <- Ya+rnorm(na,0,0.1)

# environment 2
nb <- 70
X1b <- rnorm(nb,-1,1)
Yb <- 5*X1b+rnorm(nb,0,0.5)
X2b <- rnorm(nb,0,0.1)

# combine environments
X1 <- c(X1a,X1b)
X2 <- c(X2a,X2b)
Y <- c(Ya,Yb)
Xmatrix <- cbind(X1, X2)

# apply seqICP.s to all possible sets - only the true parent set S=1
# is invariant in this example
seqICP.s(Xmatrix, Y, S=numeric(), par.test=list(grid=c(0,50,100,150,200)))
seqICP.s(Xmatrix, Y, S=1, par.test=list(grid=c(0,50,100,150,200)))
seqICP.s(Xmatrix, Y, S=2, par.test=list(grid=c(0,50,100,150,200)))
seqICP.s(Xmatrix, Y, S=c(1,2), par.test=list(grid=c(0,50,100,150,200)))
```

---

seqICPnl *Non-linear Invariant Causal Prediction*

Description

Estimates the causal parents S of the target variable Y using invariant causal prediction and fits a general model of the form
Y = f(X^S) + N.

Usage

```
seqICPnl(X, Y, test = "block.variance", par.test = list(grid = c(0,
  round(nrow(X)/2), nrow(X)), complements = FALSE, link = sum, alpha = 0.05, B =
  100), regression.fun = function(X, Y) fitted.values(lm.fit(X, Y)),
  max.parents = ncol(X), stopIfEmpty = TRUE, silent = TRUE)
```

Arguments

X	matrix of predictor variables. Each column corresponds to one predictor variable.
Y	vector of target variable, with length(Y)=nrow(X).
test	string specifying the hypothesis test used to test for invariance of a parent set S (i.e. the null hypothesis H0_S). The following tests are available: "block.mean", "block.variance", "block.decoupled", "smooth.mean", "smooth.variance", "smooth.decoupled" and "hsic".
par.test	parameters specifying hypothesis test. The following parameters are available: grid, complements, link, alpha and B. The parameter grid is an increasing vector of gridpoints used to construct enviornments for change point based tests. If the parameter complements is 'TRUE' each environment is compared against its complement if it is 'FALSE' all environments are compared pairwise. The parameter link specifies how to compare the pairwise test statistics, generally this is either max or sum. The parameter alpha is a numeric value in (0,1) indicting the significance level of the hypothesis test. The parameter B is an integer and specifies the number of Monte-Carlo samples used in the approximation of the null distribution.
regression.fun	regression function used to fit the function f. This should be a function which takes the argument (X,Y) and outputs the predicted values f(Y).
max.parents	integer specifying the maximum size for admissible parents. Reducing this below the number of predictor variables saves computational time but means that the confidence intervals lose their coverage property.
stopIfEmpty	if 'TRUE', the procedure will stop computing confidence intervals if the empty set has been accepted (and hence no variable can have a signicificant causal effect). Setting to'TRUE' will save computational time in these cases, but means that the confidence intervals lose their coverage properties for values different to 0.
silent	If 'FALSE', the procedure will output progress notifications consisting of the currently computed set S together with the p-value resulting from the null hypothesis H0_S

Details

The function can be applied to models of the form
Y_i = f(X_i^S) + N_i
with iid noise N_i and f is from a specific function class, which the regression procedure given by the parameter regression.fun should be able to approximate.

The invariant prediction procedure is applied using the hypothesis test specified by the test parameter to determine whether a candidate model is invariant. For further details see the references.

Value

object of class 'seqICPnl' consisting of the following elements

parent.set	vector of the estimated causal parents.
test.results	matrix containing results from each individual hypothesis test H0_S as rows. The first column ind links the rows in the test.results matrix to the position in the list of the variable S.
S	list of all the sets that were tested. The position within the list corresponds to the index in the first column of the test.results matrix.
p.values	p-value for being not included in the set of true causal parents. (If a p-value is smaller than alpha, the corresponding variable is a member of parent.set.)
stopIfEmpty	a boolean value indicating whether computations stop as soon as intersection of accepted sets is empty.
modelReject	a boolean value indicating if the whole model was rejected (the p-value of the best fitting model is too low).
alpha	significance level at which the hypothesis tests were performed.
n.var	number of predictor variables.

Author(s)

Niklas Pfister and Jonas Peters

References

Pfister, N., P. Bühlmann and J. Peters (2017). Invariant Causal Prediction for Sequential Data. ArXiv e-prints (1706.08058).

Peters, J., P. Bühlmann, and N. Meinshausen (2016). Causal inference using invariant prediction: identification and confidence intervals. Journal of the Royal Statistical Society, Series B (with discussion) 78 (5), 947–1012.

See Also

The function seqICPnl.s can be used to perform the individual hypothesis tests H0_S. seqICP and seqICP.s are the corresponding functions for classical sequential invariant causal prediction.

Examples

```
set.seed(2)

# environment 1
na <- 120
X1a <- 0.3*rnorm(na)
X3a <- X1a + 0.2*rnorm(na)
Ya <- 2*X1a^2 + 0.6*sin(X3a) + 0.1*rnorm(na)
X2a <- -0.5*Ya + 0.5*X3a + 0.1*rnorm(na)

# environment 2
nb <- 80
X1b <- 2*rnorm(nb)
X3b <- rnorm(nb)
Yb <- 2*X1b^2 + 0.6*sin(X3b) + 0.1*rnorm(nb)
```

```
X2b <- -0.5*Yb + 0.8*rnorm(nb)

# combine environments
X1 <- c(X1a,X1b)
X2 <- c(X2a,X2b)
X3 <- c(X3a,X3b)
Y <- c(Ya,Yb)
Xmatrix <- cbind(X1, X2, X3)

# use GAM as regression function
GAM <- function(X,Y){
  d <- ncol(X)
  if(d>1){
    formula <- "Y~1"
    names <- c("Y")
    for(i in 1:(d-1)){
      formula <- paste(formula,"+s(X",toString(i),")",sep="")
      names <- c(names,paste("X",toString(i),sep=""))
    }
    data <- data.frame(cbind(Y,X[,-1,drop=FALSE]))
      colnames(data) <- names
    fit <- fitted.values(mgcv::gam(as.formula(formula),data=data))
  } else{
    fit <- rep(mean(Y),nrow(X))
  }
  return(fit)
}

# Y follows the same structural assignment in both environments
# a and b (cf. the lines Ya <- ... and Yb <- ...).
# The direct causes of Y are X1 and X3.
# A GAM model fit considers X1, X2 and X3 as significant.
# All these variables are helpful for the prediction of Y.
summary(mgcv::gam(Y~s(X1)+s(X2)+s(X3)))

# apply seqICP to the same setting
seqICPnl.result <- seqICPnl(X = Xmatrix, Y, test="block.variance",
par.test = list(grid = seq(0, na + nb, (na + nb)/10), complements = FALSE, link = sum,
alpha = 0.05, B =100), regression.fun = GAM,  max.parents = 4, stopIfEmpty=FALSE, silent=FALSE)
summary(seqICPnl.result)
# seqICPnl is able to infer that X1 and X3 are causes of Y
```

`seqICPnl.s` *Non-linear Invariant Causal Prediction for an individual set S*

Description

Tests whether the conditional distribution of Y given X^S is invariant across time, by allowing for arbitrary non-linear additive dependence models.

Usage

```
seqICPnl.s(X, Y, S, test = "block.variance", par.test = list(grid = c(0,
  round(nrow(X)/2), nrow(X)), complements = FALSE, link = sum, alpha = 0.05, B =
  100), regression.fun = function(X, Y) fitted.values(lm.fit(X, Y)))
```

Arguments

X	matrix of predictor variables. Each column corresponds to one predictor variable.
Y	vector of target variable, with length(Y)=nrow(X).
S	vector containing the indicies of predictors to be tested
test	string specifying the hypothesis test used to test for invariance of a parent set S (i.e. the null hypothesis H0_S). The following tests are available: "block.mean", "block.variance", "block.decoupled", "smooth.mean", "smooth.variance", "smooth.decoupled" and "hsic".
par.test	parameters specifying hypothesis test. The following parameters are available: grid, complements, link, alpha and B. The parameter grid is an increasing vector of gridpoints used to construct enviornments for change point based tests. If the parameter complements is 'TRUE' each environment is compared against its complement if it is 'FALSE' all environments are compared pairwise. The parameter link specifies how to compare the pairwise test statistics, generally this is either max or sum. The parameter alpha is a numeric value in (0,1) indicting the significance level of the hypothesis test. The parameter B is an integer and specifies the number of Monte-Carlo samples used in the approximation of the null distribution.
regression.fun	regression function used to fit the function f. This should be a function which takes the argument (X,Y) and outputs the predicted values f(Y).

Details

The function can be applied to models of the form
Y_i = f(X_i^S) + N_i
with iid noise N_i and f is from a specific function class, which the regression procedure given by the parameter regression.fun should be able to approximate.

For both models the hypothesis test specified by the test parameter specifies the hypothesis test used to test whether the set S leads to an invariant model. For futher details see the references.

Value

list containing the following elements

test.stat	value of the test statistic.
crit.value	critical value computed using a Monte-Carlo simulation of the null distribution.
p.value	p-value.

Author(s)

Niklas Pfister and Jonas Peters

References

Pfister, N., P. Bühlmann and J. Peters (2017). Invariant Causal Prediction for Sequential Data. ArXiv e-prints (1706.08058).

Peters, J., P. Bühlmann, and N. Meinshausen (2016). Causal inference using invariant prediction: identification and confidence intervals. Journal of the Royal Statistical Society, Series B (with discussion) 78 (5), 947–1012.

See Also

To estimate the set of causal parents use the function `seqICPnl`. For linear models use the corresponding functions `seqICP` and `seqICP.s`.

Examples

```
set.seed(1)

# environment 1
na <- 130
X1a <- rnorm(na,0,0.1)
Ya <- 5*X1a+rnorm(na,0,0.5)
X2a <- Ya+rnorm(na,0,0.1)

# environment 2
nb <- 70
X1b <- rnorm(nb,-1,1)
Yb <- X1b^2+rnorm(nb,0,0.5)
X2b <- rnorm(nb,0,0.1)

# combine environments
X1 <- c(X1a,X1b)
X2 <- c(X2a,X2b)
Y <- c(Ya,Yb)
Xmatrix <- cbind(X1, X2)

# use GAM as regression function
GAM <- function(X,Y){
  d <- ncol(X)
  if(d>1){
    formula <- "Y~1"
    names <- c("Y")
    for(i in 1:(d-1)){
      formula <- paste(formula,"+s(X",toString(i),")",sep="")
      names <- c(names,paste("X",toString(i),sep=""))
    }
    data <- data.frame(cbind(Y,X[,-1,drop=FALSE]))
      colnames(data) <- names
    fit <- fitted.values(mgcv::gam(as.formula(formula),data=data))
```

```
  } else{
    fit <- rep(mean(Y),nrow(X))
  }
  return(fit)
}

# apply seqICPnl.s to all possible sets using the regression
# function GAM - only the true parent set S=1 is
# invariant in this example
seqICPnl.s(Xmatrix, Y, S=numeric(), par.test=list(grid=c(0,50,100,150,200)), regression.fun=GAM)
seqICPnl.s(Xmatrix, Y, S=1, par.test=list(grid=c(0,50,100,150,200)), regression.fun=GAM)
seqICPnl.s(Xmatrix, Y, S=2, par.test=list(grid=c(0,50,100,150,200)), regression.fun=GAM)
seqICPnl.s(Xmatrix, Y, S=c(1,2), par.test=list(grid=c(0,50,100,150,200)), regression.fun=GAM)
```

`summary.seqICP` *summary function*

Description

Summary functions for 'seqICP' objects.

Usage

```
## S3 method for class 'seqICP'
summary(object, show.past = TRUE, ...)
```

Arguments

`object`	object of class 'seqICP'.
`show.past`	'TRUE' if lagged variables should also be shown.
`...`	additional arguments affecting the summary produced.

Author(s)

Niklas Pfister and Jonas Peters

`summary.seqICPnl` *summary function*

Description

Summary functions for 'seqICPnl' objects.

Usage

```
## S3 method for class 'seqICPnl'
summary(object, show.past = TRUE, ...)
```

Arguments

`object`	object of class 'seqICPnl'.
`show.past`	'TRUE' if lagged variables should also be shown.
`...`	additional arguments affecting the summary produced.

Author(s)

Niklas Pfister and Jonas Peters

Index