Package 'AncReg'

July 17, 2025

Title Ancestor Regression

Version 1.0.1

Description Causal discovery in linear structural equation models (Schultheiss, and Bühlmann (2023) <doi:10.1093/biomet/asad008>) and vector autoregressive models (Schultheiss, Ulmer, and Bühlmann (2025) <doi:10.1515/jci-2024-0011>) with explicit error control for false discovery, at least asymptotically.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Imports tsutils, Rdpack, methods

Suggests pcalg, MASS, igraph, testthat (>= 3.0.0)

Config/testthat/edition 3

RdMacros Rdpack

URL http://www.markus-ulmer.ch/AncReg/

NeedsCompilation no

Author Christoph Schultheiss [aut],
Markus Ulmer [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-7783-8475>)

Maintainer Markus Ulmer <markus.ulmer@stat.math.ethz.ch>

Repository CRAN

Date/Publication 2025-07-16 15:10:08 UTC

Contents

`AncReg` *Ancestor Regression*

Description

This function performs ancestor regression for linear structural equation models (Schultheiss et al. 2025) and vector autoregressive models (Schultheiss and Bühlmann 2023) with explicit error control for false discovery, at least asymptomatically.

Usage

```
AncReg(x, degree = 0, targets = colnames(x), f = function(x) x^3)
```

Arguments

`x`	A named numeric matrix containing the observational data.
`degree`	An integer specifying the order of the SVAR process to be considered. Default is 0 for no time series.
`targets`	A character vector specifying the variables whose ancestors should be estimated. Default is all variables.
`f`	A function specifying the non-linearity used in the ancestor regression. Default is a cubic function.

Value

An object of class "AncReg" containing:

`z.val`	A numeric matrix of test statistics.
`p.val`	A numeric matrix of p-values.

References

Schultheiss C, Bühlmann P (2023). "Ancestor regression in linear structural equation models." *Biometrika*, **110**(4), 1117-1124. ISSN 1464-3510, doi:10.1093/biomet/asad008, https://academic.oup.com/biomet/article-pdf/110/4/1117/53472115/asad008.pdf.

Schultheiss C, Ulmer M, Bühlmann P (2025). "Ancestor regression in structural vector autoregressive models." doi:10.1515/jci20240011.

See Also

`summary.AncReg`, `instant_graph`, `summary_graph`, `instant_p.val`, `summary_p.val`

Examples

```
##### simulated example for inear structural equation models

# random DAGS for simulation
set.seed(1234)

p <- 5 #number of nodes
DAG <- pcalg::randomDAG(p, prob = 0.5)

B <- matrix(0, p, p) # represent DAG as matrix
for (i in 2:p){
  for(j in 1:(i-1)){
    # store edge weights
    B[i,j] <- max(0, DAG@edgeData@data[[paste(j,"|",i, sep="")]]$weight)
  }
}
colnames(B) <- rownames(B) <- LETTERS[1:p]

# solution in terms of noise
Bprime <- MASS::ginv(diag(p) - B)

n <- 5000
N <- matrix(rexp(n * p), ncol = p)
X <- t(Bprime %*% t(N))
colnames(X) <- LETTERS[1:p]

# fit ancestor regression
fit <- AncReg(X)
# collect ancestral p-values and graph
res <- summary(fit)
res

#compare true and estimated ancestral graph
trueGraph <- igraph::graph_from_adjacency_matrix(recAncestor(B != 0))
ancGraph <- igraph::graph_from_adjacency_matrix(res$graph)

oldpar <- par(mfrow = c(1, 2))
plot(trueGraph, main = 'true ancestral graph', vertex.size = 30)
plot(ancGraph, main = 'Ancestor Regression', vertex.size = 30)

##### SVAR-example with geyser timeseries
geyser <- MASS::geyser
# shift waiting such that it is waiting after erruption
geyser2 <- data.frame(waiting = geyser$waiting[-1], duration = geyser$duration[-nrow(geyser)])

# fit ancestor regression with 6 lags considered
fit2 <- AncReg(as.matrix(geyser2), degree = 6)
res2 <- summary(fit2)
res2

# visualize instantaneous ancestry
instGraph <- igraph::graph_from_adjacency_matrix(res2$inst.graph)
```

```
plot(instGraph, edge.label = round(diag(res2$inst.p.val[1:2, 2:1]), 2),
     main = 'instantaneous effects', vertex.size = 90)

# visualize summary of lagged ancestry
sumGraph <- igraph::graph_from_adjacency_matrix(res2$sum.graph)
plot(sumGraph, edge.label = round(diag(res2$sum.p.val[1:2, 2:1]), 2),
     main = 'summary graph', vertex.size = 90)
par(oldpar)
```

`instant_graph` *Instantaneous graph*

Description

Construct instantaneous graph from p-values and significance level. Recursively constructs all ancestral connections by adding ancestors of ancestors.

Usage

```
instant_graph(lin.anc, alpha = 0.05, verbose = FALSE, corr = TRUE)
```

Arguments

`lin.anc`	output from AncReg()
`alpha`	significance level
`verbose`	should information be printed?
`corr`	should multiplicity correction be applied?

Value

A list containing:

`rec.ancs`	A boolean matrix indicating whether one variable affects another instantaneously
`alpha`	The significance level to avoid cycles

See Also

AncReg, instant_p.val

Examples

```
# random DAGS for simulation
set.seed(1234)

p <- 5 #number of nodes
DAG <- pcalg::randomDAG(p, prob = 0.5)

B <- matrix(0, p, p) # represent DAG as matrix
```

```
for (i in 2:p){
  for(j in 1:(i-1)){
    # store edge weights
    B[i,j] <- max(0, DAG@edgeData@data[[paste(j,"|",i, sep="")]]$weight)
  }
}
colnames(B) <- rownames(B) <- LETTERS[1:p]

# solution in terms of noise
Bprime <- MASS::ginv(diag(p) - B)

n <- 500
N <- matrix(rexp(n * p), ncol = p)
X <- t(Bprime %*% t(N))
colnames(X) <- LETTERS[1:p]

# fit ancestor regression
fit <- AncReg(X)

# generate instantaneous graph
instant_graph(fit, alpha = 0.01, verbose = TRUE)
```

`instant_p.val` *P-values for instantaneous graph*

Description

Collect p-values for instantaneous graph.

Usage

```
instant_p.val(lin.anc)
```

Arguments

`lin.anc`	output from AncReg()

Value

A numeric matrix of p-values for the instantaneous graph

See Also

AncReg

Examples

```
# random DAGS for simulation
set.seed(1234)

p <- 5 #number of nodes
DAG <- pcalg::randomDAG(p, prob = 0.5)

B <- matrix(0, p, p) # represent DAG as matrix
for (i in 2:p){
  for(j in 1:(i-1)){
    # store edge weights
    B[i,j] <- max(0, DAG@edgeData@data[[paste(j,"|",i, sep="")]]$weight)
  }
}
colnames(B) <- rownames(B) <- LETTERS[1:p]

# solution in terms of noise
Bprime <- MASS::ginv(diag(p) - B)

n <- 500
N <- matrix(rexp(n * p), ncol = p)
X <- t(Bprime %*% t(N))
colnames(X) <- LETTERS[1:p]

# fit ancestor regression
fit <- AncReg(X)

# collect instantaneous p-values
instant_p.val(fit)
```

`recAncestor` *Recursive Ancestor Detection*

Description

This function recursively detects all ancestors of a given set of variables in a matrix. Adds ancestors of ancestors to the output matrix.

Usage

```
recAncestor(pmat)
```

Arguments

`pmat`	A boolean matrix indicating whether a connection was detected.

Value

A boolean matrix indicating whether a connection was detected or constructed.

See Also

AncReg, summary_graph

Examples

```
# random DAGS for simulation
set.seed(1234)

p <- 5 #number of nodes
DAG <- pcalg::randomDAG(p, prob = 0.5)

B <- matrix(0, p, p) # represent DAG as matrix
for (i in 2:p){
  for(j in 1:(i-1)){
    # store edge weights
    B[i,j] <- max(0, DAG@edgeData@data[[paste(j,"|",i, sep="")]]$weight)
  }
}
colnames(B) <- rownames(B) <- LETTERS[1:p]

# edge effects to adjecency matrix
B <- B != 0

# transform adjacency matrix to ancestral matrix
recAncestor(B)
```

---

summary.AncReg *Summary of AncReg*

Description

Summarize the results of AncReg. For models with degree = 0 only the instantaneous graph is returned and for models with degree > 0 the summary graph is returned as well.

Usage

```
## S3 method for class 'AncReg'
summary(object, alpha = 0.05, verbose = FALSE, corr = TRUE, ...)
```

Arguments

object	output from AncReg()
alpha	significance level for determin whether a connection is significant
verbose	should information be printed?
corr	should multiplicity correction be applied?
...	Further arguments passed to or from other methods.

Value

A list containing: If `degree = 0`:

`p.val`	A numeric matrix of p-values for the instantaneous graph
`graph`	A boolean matrix indicating whether one variable affects another instantaneously
`alpha`	The significance level to avoid cycles

If `degree > 0`:

`inst.p.val`	A numeric matrix of p-values for the instantaneous graph
`inst.graph`	A boolean matrix indicating whether one variable affects another instantaneously
`inst.alpha`	The significance level to avoid cycles
`sum.p.val`	A numeric matrix of p-values for the summary graph
`sum.graph`	A boolean matrix indicating whether one variable affects another

See Also

`AncReg`, `instant_graph`, `summary_graph`, `instant_p.val`, `summary_p.val`

Examples

```
# random DAGS for simulation
set.seed(1234)

p <- 5 #number of nodes
DAG <- pcalg::randomDAG(p, prob = 0.5)

B <- matrix(0, p, p) # represent DAG as matrix
for (i in 2:p){
  for(j in 1:(i-1)){
    # store edge weights
    B[i,j] <- max(0, DAG@edgeData@data[[paste(j,"|",i, sep="")]]$weight)
  }
}
colnames(B) <- rownames(B) <- LETTERS[1:p]

# solution in terms of noise
Bprime <- MASS::ginv(diag(p) - B)

n <- 500
N <- matrix(rexp(n * p), ncol = p)
X <- t(Bprime %*% t(N))
colnames(X) <- LETTERS[1:p]

# fit ancestor regression
fit <- AncReg(X)
# collect ancestral p-values and graph
res <- summary(fit, alpha = 1)
res
```

summary_graph *Summary graph*

Description

Construct summary graph from p-values and significance level. Recursively constructs all ancestral connections by adding ancestors of ancestors.

Usage

```
summary_graph(lin.anc, alpha = 0.05, corr = TRUE)
```

Arguments

lin.anc	output from AncReg()
alpha	significance level
corr	should multiplicity correction be applied?

Value

A boolean matrix indicating whether one variable affects another

See Also

AncReg, summary_p.val

Examples

```
# random DAGS for simulation
set.seed(1234)

p <- 5 #number of nodes
DAG <- pcalg::randomDAG(p, prob = 0.5)

B <- matrix(0, p, p) # represent DAG as matrix
for (i in 2:p){
  for(j in 1:(i-1)){
    # store edge weights
    B[i,j] <- max(0, DAG@edgeData@data[[paste(j,"|",i, sep="")]]$weight)
  }
}
colnames(B) <- rownames(B) <- LETTERS[1:p]

# solution in terms of noise
Bprime <- MASS::ginv(diag(p) - B)

n <- 500
N <- matrix(rexp(n * p), ncol = p)
X <- t(Bprime %*% t(N))
```

```
colnames(X) <- LETTERS[1:p]

# fit ancestor regression
fit <- AncReg(X)

# generate summary graph
summary_graph(fit, alpha = 0.1)
```

---

summary_p.val *P-values for summary graph*

Description

Collect p-values for summary graph.

Usage

```
summary_p.val(lin.anc)
```

Arguments

lin.anc	output from AncReg()

Value

A numeric matrix of p-values for the summary graph

See Also

AncReg

Examples

```
# random DAGS for simulation
set.seed(1234)

p <- 5 #number of nodes
DAG <- pcalg::randomDAG(p, prob = 0.5)

B <- matrix(0, p, p) # represent DAG as matrix
for (i in 2:p){
  for(j in 1:(i-1)){
    # store edge weights
    B[i,j] <- max(0, DAG@edgeData@data[[paste(j,"|",i, sep="")]]$weight)
  }
}
colnames(B) <- rownames(B) <- LETTERS[1:p]

# solution in terms of noise
Bprime <- MASS::ginv(diag(p) - B)
```

```
n <- 500
N <- matrix(rexp(n * p), ncol = p)
X <- t(Bprime %*% t(N))
colnames(X) <- LETTERS[1:p]

# fit ancestor regression
fit <- AncReg(X)

# collect summary p-values
summary_p.val(fit)
```

Index