Package ‘gRc’

October 15, 2024

Version 0.5.1

Title Inference in Graphical Gaussian Models with Edge and Vertex Symmetries

Maintainer Søren Højsgaard <sorenh@math.aau.dk>

Description Estimation, model selection and other aspects of statistical inference in Graphical Gaussian models with edge and vertex symmetries (Graphical Gaussian models with colours). Documentation about 'gRc' is provided in the paper by Hojsgaard and Lauritzen (2007, <doi:10.18637/jss.v023.i06>) and the paper by Hojsgaard and Lauritzen (2008, <doi:10.1111/j.1467-9868.2008.00666.x>).

License GPL

Encoding UTF-8

Depends R (>= 4.2.0), methods, gRbase (>= 2.0.0)

Imports MASS, igraph, stats4, Rcpp (>= 0.11.1)

Suggests microbenchmark, knitr

LinkingTo Rcpp (>= 0.11.1), RcppArmadillo

URL `https://people.math.aau.dk/~sorenh/software/gR/`

Repository CRAN

ByteCompile Yes

RoxygenNote 7.3.2

NeedsCompilation yes

Date/Publication 2024-10-15 07:40:03 UTC

Author Steffen Lauritzen [aut, cph], Søren Højsgaard [aut, cre, cph]

Contents

`add1-drop1` *Add or drop colour classes to RCOX models*

Description

Make a test for adding/dropping all colour classes in scope for an RCOX model.

Usage

```
## S3 method for class 'rcox'
add1(object, scope, details = 0, trace = 0, ...)

## S3 method for class 'rcox'
drop1(object, scope, details = 0, trace = 0, stat = "wald", ...)
```

Arguments

`object`	An RCOX model, an object of class ’rcox’
`scope`	A set of edge colour classes to be considered for addition or deletion, see ’details’.
`details`	Control the amount of output created.
`trace`	For debugging purposes
`...`	Additional arguments, currently unused.
`stat`	Either "wald" for a Wald statistic or "dev" for a deviance statistic.

Value

A list with entries * tab: A dataframe with the test results * cc: A list of colour classes

Note

* Note that the keyword 'stat' is not available for add1 because this function expands the current model and hence the Wald statistic is not available.

* Based on fitting a new model for each new ecc, and may hence be slow.

Author(s)

Søren Højsgaard, `<sorenh@math.aau.dk>`

See Also

`comparecc`, `stepadd1`, `stepdrop1`.

Examples

```
data(math)
gc.sat <- ~me:ve:al:st:an
gc.1   <- ~me+ve+al+st+an

m.sat <- rcox(gm=gc.sat, data=math)
m.1   <- rcox(gm=gc.1,   data=math)

t.sat <- drop1(m.sat)
t.sat$tab
t.sat$cc

t.1   <- add1(m.1)
t.1$tab
t.1$cc
```

`clever_trace` *Calculate trace of various matrix products.*

Description

Calculate trace of various matrix products.

Usage

```
trA(A)

trAW(rA, rW)

trAWB(rA, rW, rB)

trAWBW(rA, rW, rB)

trAWBV(rA, rW, rB, rV)
```

```
trAWBlist(Alist, W, Blist, mode = 0L)

trAWBWlist(Alist, W, Blist, mode = 0L)
```

Arguments

`A`	A matrix
`rA`, `rB`	Square matrices represented as "sparse" matrices. The dimension is inferred from dimension of V,W.
`rW`, `rV`, `W`	Square matrices; must be of same dimension.
`Alist`, `Blist`	Lists of such "sparse" matrices.
`mode`	0 or 1.

Value

A number

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

Examples

```
## Turn list into "sparse" matrix
to_sp <- function(x){
  ans <- do.call("rbind", x)
  storage.mode(ans)<-"double"
  return(ans)
}

## Turn "sparse" matrix into dense matrix
##
to_de <- function(A,d){
  if (nrow(A) < 1) stop("need non-empty matrix\n")
  ans <- matrix(0,nr=d,nc=d)
  for (i in 1:nrow(A)){
    e <- A[i,]
    if (length(e) == 1){
      ans[e,e] <- 1
    } else {
      ans[e[1],e[2]] <-   ans[e[2],e[1]] <- 1
    }
  }
  return(ans)
}

d <- 5
W <- matrix(rnorm(d * d), nr=d, nc=d);
V <- matrix(rnorm(d * d), nr=d, nc=d);
```

```
W <- W + t(W)
V <- V + t(V)


A1.lst <- list(c(1,2),c(1,3))
A2.lst <- list(1,3,5)

A1.sp <- to_sp(A1.lst)
A2.sp <- to_sp(A2.lst)

A1.de <- to_de(A1.sp, d)
A2.de <- to_de(A2.sp, d)

trAW(A1.sp, W)
sum(diag(A1.de %*% W))

trAW(A2.sp, W)
sum(diag(A2.de %*% W))

trAWB(A1.sp, W, A2.sp)
trAWB(A2.sp, W, A1.sp)
sum(diag(A1.de %*% W %*% A2.de))

trAWBW(A1.sp, W, A2.sp)
trAWBW(A2.sp, W, A1.sp)
sum(diag(A1.de %*% W %*% A2.de %*% W))

trAWBV(A1.sp, W, A2.sp, V)
trAWBV(A2.sp, W, A1.sp, V)
sum(diag(A1.de %*% W %*% A2.de %*% V))
```

`coerce_parm` *Coerce parameters in RCOX models.*

Description

Coerce parameters in RCOX models. These functions should not be called directly by the user.

Usage

```
theta2K(object, theta, scale = "original")

K2theta(object, K, scale = "original")

## S3 method for class 'rcon'
theta2K(object, theta, scale = "original")

## S3 method for class 'rcor'
```

```
theta2K(object, theta, scale = "original")

## S3 method for class 'rcon'
K2theta(object, K, scale = "original")

## S3 method for class 'rcor'
K2theta(object, K, scale = "original")
```

Arguments

`object`	Model object.
`theta`	Parameter.
`scale`	On which scales are the parameters computed.
`K`	Concentration matrix.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

comparecc — *Compare colour classes of an RCOX model*

Description

A general function for pairwise comparisons of colour classes in an RCOX model, i.e. for testing whether the corresponding parameters are significantly different

Usage

```
comparecc(
  object,
  cc1 = NULL,
  cc2 = NULL,
  type = "ecc",
  stat = "wald",
  details = 1
)
```

Arguments

`object`	An RCOX model, an object of class 'rcox'
`cc1, cc2`	Lists of colour classes of type 'type', see 'details' for an explanation of the defaults.
`type`	Either "ecc" for edge colour classes or "vcc" for vertex colour classes
`stat`	Base the comparison on either "wald" for a Wald statistic or "dev" for a deviance statistic
`details`	Control the amount of output created.

Details

All colour classes specified in `cc1` are compared with all those given in `cc2` (duplicate entries are not compared). If `cc2=NULL` (the default) then all colour classes specified in `cc1` are compared with all colour classes in the model except those specified in `cc1`. If `cc1=NULL` (the default) and `cc2=NULL` then all pairwise comparisons are made.

Value

A list with entries:

`tab`	A data frame with the test results
`cc1`, `cc2`	Lists of colour classes

Author(s)

Søren Højsgaard, sorenh@math.aau.dk

See Also

`add1.rcox`, `drop1.rcox`, `stepadd1`, `stepdrop1`, `join1`, `split1`, `stepjoin1`, `stepsplit1`

Examples

```
data (math)

gm  = ~al:an:st
vcc = list(~me+st, ~ve+an, ~al)
ecc = list(~me:ve+me:al, ~ve:al+al:st)

m1 <- rcox(gm=gm, vcc=vcc, ecc=ecc, data=math)
m1

comparecc(m1, type="vcc")
comparecc(m1, type="ecc")
```

`fit-main` *Fit rcox model*

Description

This is a general function for fitting RCOX models (i.e.\ RCON and RCOR models) using different estimation algorithms.

Usage

```
## S3 method for class 'rcox'
fit(
  object,
  Kstart = object$Kstart,
  method = object$method,
  control = object$control,
  details = object$details,
  trace = object$trace,
  returnModel = TRUE,
  ...
)

matching(object, control = object$control, trace = object$trace)

ipm(object, K0, control = object$control, trace = object$trace)

scoring(
  object,
  K0,
  control = object$control,
  maxit = control$maxouter,
  trace = object$trace
)
```

Arguments

`object`	An RCOX model object (an object of class 'rcox')
`Kstart`	An initial value for the concentration matrix.
`method`	The specific estimation method. Can be either "scoring", (a modified Fisher scoring algorithm), "ipm" (iterative partial maximization), "matching" (score matching) or "user" (currently not used).
`control`	A list controlling the fitting algorithms. See the 'details' section.
`details`	The amount of details printed on the screen. 0 means no details at all.
`trace`	Controls various diagnostics print outs. A debugging feature not intended for the user.
`returnModel`	If TRUE the model object m is returned with fitting info added to it. If FALSE only the fitting info is returned.
`...`	Additional arguments; currently not used.
`K0`	Initial value for concentration matrix.
`maxit`	Maximum number of iterations.

Details

The fitted parameters etc. can be extracted using 'fitInfo(m)'. The control argument is a list with named entries. Most important are the entries 'maxouter' and 'maxinner' (which both defaults to 25) for controlling the estimation algorithms. For other components please refer to the code.

Value

An RCOX model object.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

rcox, update.rcox

Examples

```
data(math)
gm  = ~al:an:st
vcc = list(~me + st, ~ve + an, ~al)
ecc = list(~me:ve + me:al, ~ve:al + al:st)

m1 <- rcox(gm=gm, vcc=vcc, ecc=ecc, data=math, fit=FALSE)

fit(m1, method="matching")
fit(m1, method="scoring")
fit(m1, method="ipm")
```

---

generating-class *Generating class for RCON / RCOR models*

Description

Implementation of generating classes for RCOX models. These functions are not intended to be called by the user.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

get-slot — *Get slots from RCOX model object.*

Description

Get slots from RCOX model object.

Usage

```
getSlot(object, slot)

dimension(object)

logL(object)

getSlot(object, slot)

dataRep(object, slot = NULL)

intRep(object, slot = NULL)

fitInfo(object, slot = NULL)

getcc(object, type)

getecc(object)

getvcc(object)

getedges(object, complement = FALSE)
```

Arguments

`object`	RCOX model object.
`slot`	slot.
`type`	Type of colour class.
`complement`	If FALSE, the edges of the model is returned. If TRUE, the edges not in the model is returned.

Author(s)

Søren Højsgaard, `<sorenh@math.aau.dk>`

Examples

```
data(math)
gm  = ~al:an:st
vcc = list(~me+st, ~ve+an, ~al)
ecc = list(~me:ve+me:al, ~ve:al+al:st)

m1 <- rcox(gm=gm, vcc=vcc, ecc=ecc, data=math)
getecc(m1)

getSlot(m1,"type")
fitInfo(m1)
fitInfo(m1,"K")
```

`grc-summary` *The package 'gRc': summary information*

Description

This package is for statistical inference in RCOX models. That is, graphical Gaussian models where specific entries of the inverse covariance matrix or partial correlation matrix have been restricted to being equal. Entries which are restricted to being identical are displayed with identical colours in the independence graph. Hence the name of the package gRc: The "c" stands for colours.

Details

#' @author Søren Højsgaard, `<sorenh@math.aau.dk>`

Details

The function for specifying RCOX models is rcox, and we refer to the help page for that function for examples.

See Also

`rcox`

`join-split` *Joining and splitting of colour classes in RCOX models*

Description

Test for joining of two colour classes (of a specific type) by testing if their corresponding parameters are not significantly different. Split a colour class and test how much this changes the fit of the model.

Arguments

`object`	An RCOX model, an object of class RCOX
`scope`	A specification of colour classes which should be considered for joining/splitting. If NULL, then all colour classes are considered.
`type`	Either "ecc" for edge colour classes or "vcc" for vertex colour classes.
`stat`	Either "wald" for a Wald statistic or "dev" for deviance statistic.
`details`	Control the amount of output

Value

A list with entries:

`tab`	A data frame with the test results
`cc`	A list of colour classes

Note

Note that the keyword 'stat' is not available for split1 because this function expands the current and hence the Wald statistic is not available. Note also that join1 is simply a wrapper for comparecc applied to edge colour classes.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

`rcox`, `update`, `comparecc`

Examples

```
data(math)
g1     <- ~me:ve:al+al:st:an
m1     <- rcox(gm=g1, data=math)
join1(m1)

gm  = ~al:an:st
vcc = list(~me+st, ~ve+an)
ecc = list(~me:ve+me:al, ~ve:al+al:st)
m2 <- rcox(gm=gm, vcc=vcc, ecc=ecc, data=math, type="rcon")
split1(m2)
```

plot.rcox *Plot rcox object*

Description

Plot rcox object

Usage

```
## S3 method for class 'rcox'
plot(x, y, ...)
```

Arguments

x	An rcox object
y	Ignored
...	Currently not used

Examples

```
gm  = ~al:an:st
vcc = list(~me+st, ~ve+an, ~al)
ecc = list(~me:ve+me:al, ~ve:al+al:st)
m1 <- rcox(gm=gm, vcc=vcc, ecc=ecc, data=math, method='matching')
plot(m1)
```

rcox *Main function for specifying RCON/RCOR models.*

Description

This is the main function for specifying and fitting RCON/RCOR models in the package along with certain utility functions.

Usage

```
rcox(
  gm = NULL,
  vcc = NULL,
  ecc = NULL,
  type = c("rcon", "rcor"),
  method = "ipm",
  fit = TRUE,
  data = NULL,
  S = NULL,
  n = NULL,
  Kstart = NULL,
  control = list(),
  details = 1,
  trace = 0
)
```

Arguments

`gm`	Generating class for a grapical Gaussian model, see 'Examples' for an illustration
`vcc`	List of vertex colour classes for the model
`ecc`	List of edge colour classes for the model
`type`	Type of model. Default is RCON
`method`	Estimation method; see 'Details' below.
`fit`	Should the model be fitted
`data`	A dataframe
`S`	An empirical covariance matrix (as alternative to giving data as a dataframe)
`n`	The number of observations (which is needed if data is specified as an empirical covariance matrix)
`Kstart`	An initial value for K. Can be omitted.
`control`	Controlling the fitting algorithms
`details`	Controls the amount of output
`trace`	Debugging info

Details

Estimation methods:

* 'ipm' (default) is iterative partial maximization which when finished calculates the information matrix so that approximate variances of the parameters can be obtained using vcov().

* 'ipms' is iterative partial maximization without calculating the information matrix. This is the fastest method.

* 'scoring' is stabilised Fisher scoring.

* 'matching' is score matching followed by one step with Fisher scoring.

* 'hybrid1' is for internal use and should not be called directly

Value

A model object of type 'RCOX'.

Author(s)

Søren Højsgaard, `<sorenh@math.aau.dk>`

Examples

```
data(math)
gm  = ~al:an:st
vcc = list(~me+st, ~ve+an, ~al)
ecc = list(~me:ve+me:al, ~ve:al+al:st)

m1 <- rcox(gm=gm, vcc=vcc, ecc=ecc, data=math, method='matching')
m2 <- rcox(gm=gm, vcc=vcc, ecc=ecc, data=math, method='scoring')
m3 <- rcox(gm=gm, vcc=vcc, ecc=ecc, data=math, method='ipm')

m1
m2
m3

summary(m1)
summary(m2)
summary(m3)

coef(m1)
coef(m2)
coef(m3)

vcov(m1)
vcov(m2)
vcov(m3)
```

`score-information` *Get score and information for RCOX models.*

Description

Get score and information for RCOX models.

Usage

```
getScore(m, K, scale = "original")

## S3 method for class 'rcon'
getScore(m, K, scale = "original")

## S3 method for class 'rcor'
getScore(m, K, scale = "original")
```

Arguments

`m`	Model obect.
`K`	Concentration matrix
`scale`	On which scales are the parameters computed.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

---

step *Stepwise model selection in RCOX models*

Description

These allow for stepwise model selection in RCOX models by. Model expansion (i.e. forward selection) is obtained by adding edge colour classes and by splitting edge/vertex colour classes. Model reduction (i.e. backward selection) is obtained by dropping edge colour classes and by joining edge/vertex colour classes.

Arguments

`object`	An RCOX model, an object of class RCOX
`scope`	A set (list) of items (edge colour classes or vertex colour classes) to be considered. If missing, then all items are considered.
`criterion`	Either "aic" (the default), "bic" or "test" (for significance test)
`type`	Either "ecc" for edge colour classes or "vcc" for vertex colour classes.
`k`	The multiple of the number of degrees of freedom used for the penalty when criterion is "aic". Ignored when criterion is "bic" or "test". Only k = 2 gives the genuine AIC.
`steps`	The maximum number of steps to be considered. The default is 1000 (essentially as many as required). It is typically used to stop the process early
`stat`	Either "wald" for a Wald statistic or "dev" for a deviance statistic.
`alpha`	Critical value if 'criterion' is "test". If criterion is "aic" or "bic", the critical value is 0.
`headlong`	If TRUE then at each step the first encountered edge that may be removed/added according to the current criterion is done so.
`random`	If TRUE, then the edges are examined in random order
`details`	Control the amount of output created.
`trace`	For debugging purposes
`...`	Additional arguments, currently not used.

Value

Either NULL or a new RCOX model.

Note

Note that the keyword 'stat' is not available for stepadd1 and stepsplit1 because these functions expand the current model and hence the Wald statistic is not available.

Author(s)

Søren Højsgaard, `<sorenh@math.aau.dk>`

See Also

`split1` `join1` `add1.rcox` `drop1.rcox` `comparecc`

`toLisp` *For printing in lisp style*

Description

For printing in lisp style

Usage

```
toLisp(v)
```

Arguments

`v`	Object to be printed

`update-doc` *Update an RCOX model*

Description

update will update and (by default) re-fit an RCOX model. It does this by extracting the call stored in the object, updating the call and (by default) evaluating that call. Sometimes it is useful to call update with only one argument, for example if the data frame has been corrected

Arguments

`object`	An RCOX model, an object of class RCOX
`vcc`	Specification of the vertex colour classes in the model
`ecc`	Specification of the edge colour classes in the model
`splitvcc`	Existing vertex colour class to be split
`splitecc`	Existing edge colour class to be split
`joinvcc`	Existing vertex colour classes to be joined
`joinecc`	Existing vertex colour classes to be joined
`addecc`	New edge colour classes to be added
`dropecc`	Existing vertex color classes to be dropped (deleted)
`Kstart`	A start value for K

`fit`	Should the updated model be fitted.
`control`	A list of control parameters.
`trace`	For debugging purposes
`...`	Additional arguments, currently not used.

Value

A new model object of class 'rcox'.

Warning

Only one of the arguments pertaining to edge colour classes (i.e. ecc, splitecc, joinecc, dropecc, addecc) should be applied at the time. Likewise for the arguments pertaining to the vertex colour classes.

The result will otherwise be highly unpredictable and is likely to cause an error.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

See Also

rcox

Examples

```
data(math)
gm  = ~al:an:st
vcc = list(~me+st, ~ve+an, ~al)
ecc = list(~me:ve+me:al, ~ve:al+al:st)

m1 <- rcox(gm=gm, vcc=vcc, ecc=ecc, data=math, method='matching', trace=0)

update(m1, joinvcc=list(~me+st, ~ve+an))
update(m1, joinecc=list(~al:an, ~an:st))

update(m1, splitvcc=~ve+an)
update(m1, splitecc=~me:ve+me:al)


update(m1, dropecc=list(~me:st+st:an,~al:an,~st:al))
update(m1, addecc=list(~an:me+st:ve))
```

`xxx2yyy` *Conversions of the type xxx2yyy*

Description

Conversions of the type xxx2yyy.

Author(s)

Søren Højsgaard, `<sorenh@math.aau.dk>`

Index