

# Package ‘gRc’

March 20, 2012

**Version** 0.4-0

**Title** Inference in Graphical Gaussian Models with Edge and Vertex Symmetries

**Author** Søren Højsgaard <sorenh@mail.dk>, Steffen L. Lauritzen <steffen@stats.ox.ac.uk>

**Maintainer** Søren Højsgaard <sorenh@agrsci.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)

**License** GPL

**Encoding** latin1

**Depends** gRbase,graph

**Suggests** Rgraphviz

**Repository** CRAN

**Date/Publication** 2012-03-20 21:30:22

## R topics documented:

add1drop1 . . . . .	2
comparecc . . . . .	3
fit . . . . .	4
getSlot . . . . .	6
gRc . . . . .	7
join1split1 . . . . .	7
rcox . . . . .	8
stepwise . . . . .	10
tr . . . . .	12
update.rcox . . . . .	13

<b>Index</b>	<b>15</b>
--------------	-----------

---

`add1drop1`*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**

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

**Value**

A list with entries:

<code>tab</code>	A data frame with the test results
<code>cc</code>	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.

**Author(s)**

Søren Højsgaard, [sorenh@agrsci.dk](mailto:sorenh@agrsci.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

```

---

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@agrsci.dk](mailto:sorenh@agrsci.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

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

---

fit

*Fit RCOX models*

---

**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,...)
```

**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.

**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@agrsci.dk](mailto:sorenh@agrsci.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")

## MISSING
```

---

`getSlot`*Accessing RCOX model objects*

---

**Description**

Accessing RCOX model objects

**Usage**

```
## Accessor functions
getSlot(object, slot)
fitInfo(object, slot)
intRep(object, slot)
dataRep(object, slot)
getecc(object)
getvcc(object)
getedges(object, complement=FALSE)
```

**Arguments**

<code>object</code>	An RCOX model object.
<code>slot</code>	A name of a slot.
<code>complement</code>	If FALSE, the edges of the model is returned. If TRUE, the edges not in the model is returned

**Author(s)**

Søren Højsgaard, [sorenh@agrsci.dk](mailto:sorenh@agrsci.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)
getecc(m1)

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

gRc

*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**

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

**Authors**

Søren Højsgaard, [sorenh@agrsci.dk](mailto:sorenh@agrsci.dk)

**See Also**

[rcox](#)

join1split1

*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.

**Usage**

```
join1(object, scope=NULL, type = "ecc", details = 1, stat = "wald")
split1(object, scope=NULL, type = "ecc", details = 1)
```

**Arguments**

<code>object</code>	An RCOX model, an object of class RCOX
<code>scope</code>	A specification of colour classes which should be considered for joining/splitting. If NULL, then all colour classes are considered.
<code>type</code>	Either "ecc" for edge colour classes or "vcc" for vertex colour classes.
<code>stat</code>	Either "wald" for a Wald statistic or "dev" for deviance statistic.
<code>details</code>	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@agrsci.dk](mailto:sorenh@agrsci.dk)

**References**

~put references to the literature/web site here ~

**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)
```

---

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, control = list(),
     details=1, trace=0)
```

**Arguments**

gm	Generating class for a graphical 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'.

**Note**

`demo("gRc-JSS")` gives a more comprehensive demo.

**Author(s)**

Søren Højsgaard, [sorenh@agrsci.dk](mailto:sorenh@agrsci.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)

```

---

stepwise

*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.

**Usage**

```

stepadd1 (object, criterion = "aic", steps = 1000, k = 2, alpha = 0.05,
headlong=FALSE, random=TRUE, details=1, trace=0,...)
stepdrop1 (object, criterion = "aic", steps = 1000, k = 2, alpha = 0.05,
stat = "wald", headlong=FALSE, random=TRUE, details=1, trace=0,...)
stepjoin1 (object, scope, type = "ecc", criterion = "aic", steps = 1000, k = 2, alpha = 0.05, stat = "wald", de
stepsplit1(object, type = "ecc", criterion = "aic", steps = 1000, k = 2, alpha = 0.05, stat = "wald", de

```

**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@agrsci.dk](mailto:sorenh@agrsci.dk)

**See Also**

[split1](#) [join1](#) [add1.rcox](#) [drop1.rcox](#) [comparecc](#)

---

tr *Calculate trace of various matrix products*

---

### Description

Calculate trace of various matrix products.

### Usage

```
trA(A)
trAW(A,W)
trAWB(A,W,B)
trAWBW(A,W,B)
trAWBV(A,W,B,V)
```

### Arguments

A,B	Square matrices represented as matrices or lists (see examples below).
W,V	Square matrices

### Value

A number

### Author(s)

Søren Højsgaard, [sorenh@agrsci.dk](mailto:sorenh@agrsci.dk)

### Examples

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

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

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

## Just for checking the calculations
##
symMat <- function(A,d){
```

```

ans <- matrix(0,nr=d,nc=d)
for (i in 1:length(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)
}

trAW(A1, W)
#sum(diag(symMat(A1,d=d) %*% W))

trAW(A2, W)
#sum(diag(symMat(A2,d=d) %*% W))

trAWB(A1, W, A2)
#sum(diag(symMat(A1,d=d) %*% W %*% symMat(A2,d=d)))

trAWBV(A1, W, A2, V)
#sum(diag(symMat(A1,d=d) %*% W %*% symMat(A2,d=d) %*% V))

```

---

update.rcox

*Update an RCOX model*


---

## Description

update will update and (by default) re-fit an RCOX model.

## Usage

```

## S3 method for class 'rcox'
update(object, vcc = NULL, ecc = NULL, splitecc = NULL,
splitvcc = NULL, joinvcc = NULL, joinecc = NULL, addecc = NULL, dropecc
= NULL, Kstart = NULL, fit = TRUE, control=NULL, trace = object$trace, ...)

```

## 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@agrsci.dk](mailto:sorenh@agrsci.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))
```

# Index

- \*Topic **graphs**
  - [gRc](#), [7](#)
- \*Topic **htest**
  - [add1drop1](#), [2](#)
  - [comparecc](#), [3](#)
  - [join1split1](#), [7](#)
  - [stepwise](#), [10](#)
- \*Topic **models**
  - [fit](#), [4](#)
  - [gRc](#), [7](#)
  - [rcox](#), [8](#)
  - [update.rcox](#), [13](#)
- \*Topic **multivariate**
  - [gRc](#), [7](#)
- \*Topic **utilities**
  - [getSlot](#), [6](#)
  - [tr](#), [12](#)

[add1.rcox](#), [4](#), [11](#)  
[add1.rcox \(add1drop1\)](#), [2](#)  
[add1drop1](#), [2](#)

[comparecc](#), [2](#), [3](#), [8](#), [11](#)

[dataRep \(getSlot\)](#), [6](#)  
[drop1.rcox](#), [4](#), [11](#)  
[drop1.rcox \(add1drop1\)](#), [2](#)

[fit](#), [4](#)  
[fitInfo \(getSlot\)](#), [6](#)

[getecc \(getSlot\)](#), [6](#)  
[getedges \(getSlot\)](#), [6](#)  
[getSlot](#), [6](#)  
[getvcc \(getSlot\)](#), [6](#)  
[gRc](#), [7](#)

[intRep \(getSlot\)](#), [6](#)

[join1](#), [4](#), [11](#)  
[join1 \(join1split1\)](#), [7](#)

[join1split1](#), [7](#)

[rcox](#), [5–7](#), [8](#), [8](#), [14](#)

[split1](#), [4](#), [11](#)  
[split1 \(join1split1\)](#), [7](#)  
[stepadd1](#), [2](#), [4](#)  
[stepadd1 \(stepwise\)](#), [10](#)  
[stepdrop1](#), [2](#), [4](#)  
[stepdrop1 \(stepwise\)](#), [10](#)  
[stepjoin1](#), [4](#)  
[stepjoin1 \(stepwise\)](#), [10](#)  
[stepsplit1](#), [4](#)  
[stepsplit1 \(stepwise\)](#), [10](#)  
[stepwise](#), [10](#)

[tr](#), [12](#)  
[trA \(tr\)](#), [12](#)  
[trAW \(tr\)](#), [12](#)  
[trAWB \(tr\)](#), [12](#)  
[trAWBV \(tr\)](#), [12](#)  
[trAWBW \(tr\)](#), [12](#)

[update](#), [8](#)  
[update.rcox](#), [5](#), [13](#)