



## Inference in Graphical Gaussian Models with Edge and Vertex Symmetries with the `gRc` package for R

Søren Højsgaard  
Aarhus University

Steffen L. Lauritzen  
University of Oxford

---

### Abstract

In this paper we present the R package `gRc` for statistical inference in graphical Gaussian models in which symmetry restrictions have been imposed on the concentration or partial correlation matrix. The models are represented by coloured graphs where parameters associated with edges or vertices of same colour are restricted to being identical. We describe algorithms for maximum likelihood estimation and discuss model selection issues. The paper illustrates the practical use of the `gRc` package.

*Keywords:* concentration matrix, conditional independence, covariance selection, graphical model, graph, graph colouring, iterative partial maximization, iterative proportional scaling, multivariate normal distribution, partial correlation .

---

## 1. Introduction

1 This paper describes an R package, (R Development Core Team 2007), for statistical inference  
2 in a class of graphical Gaussian models with edge and vertex symmetries as introduced by  
3 Højsgaard and Lauritzen (2007), see also Højsgaard and Lauritzen (2005). The models gen-  
4 eralise graphical Gaussian models (hereafter abbreviated GGMs) (Whittaker 1990; Lauritzen  
5 1996), also known as covariance selection models (Dempster 1972).

6 There are two types of models available in `gRc`. In one type, denoted RCON models, selected  
7 elements of the concentration matrix (the inverse covariance matrix) are restricted to being  
8 identical. These models are all linear in the inverse covariance matrix and are therefore  
9 instances of models discussed by Anderson (1970). In the other class, denoted RCOR models,  
10 it is the partial *correlations* rather than the concentrations which are restricted to being equal.  
11 We use RCOX models as a generic term for both types. The `gRc` package is part of the gR  
12 initiative (Lauritzen 2002) aiming to make graphical models available in R.

## 2. Preliminaries and notation

### 2.1. Graph colouring

Consider an undirected graph  $\mathcal{G} = (V, E)$ . Colouring the vertices of  $\mathcal{G}$  with  $R \leq |V|$  different colours induces a partitioning of  $V$  into disjoint sets  $V_1, \dots, V_R$  called *vertex colour classes* where all vertices in  $V_r$  have the same colour. Here  $|V|$  denotes the number of elements in  $V$ . A similar colouring of the edges  $E$  with  $S \leq |E|$  different colours yields a partitioning of  $E$  into disjoint sets  $E_1, \dots, E_S$  called *edge colour classes* where all edges in  $E_s$  have the same colour. We say that  $\mathcal{V} = \{V_1, \dots, V_R\}$  is a *vertex colouring* and  $\mathcal{E} = \{E_1, \dots, E_S\}$  is an *edge colouring*.

A colour class with only one element is said to be *atomic*. A colour class which is not atomic is *composite*. A set  $a \subset V$  is called *neutral* if its induced subgraph has only atomic colour classes.

When drawing vertices/edges we make the convention that black and white are used for atomic colour classes. Thus two edges displayed in black will be in different (atomic) colour classes.

Figure 1 illustrates a graph colouring. The edge between vertices 1 and 2 is written 1:2 etc. The coloured graph in (a) is given by  $(\mathcal{V}, \mathcal{E})$  where

$$\mathcal{V} = [1, 4][2, 3], \quad \mathcal{E} = (1:2, 1:3)(2:4, 3:4)$$

whereas the graph in (b) is given by  $\mathcal{V} = [1, 4][2][3]$  and  $\mathcal{E} = (1:2, 1:3)(2:4)(3:4)$ .

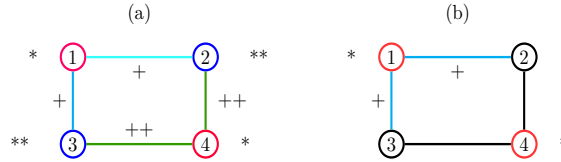


Figure 1: Coloured graphs. (a): The edges 1:2 and 1:3 are in the same (light blue) edge colour class as also indicated by the “+”-sign. Likewise, 2:4 and 3:4 are in the same (green) edge colour class, also indicated by “++”. The vertices 1 and 4 are in the red vertex colour class (also indicated by “\*\*”) while vertices 2 and 3 are in the blue vertex colour class (indicated by “\*”). (b): Illustration of atomic colour classes. The vertices 2 and 3 are drawn in black and are atomic, so 2 and 3 are in different vertex colour classes. Likewise for edges 2:4 and 3:4.

### 2.2. Graphical Gaussian models

Graphical Gaussian models are concerned with the distribution of a multivariate random vector  $Y = (Y_\alpha)_{\alpha \in V}$  following a  $N_d(\mu, \Sigma)$  distribution where  $d = |V|$ . For simplicity we assume throughout that  $\mu = 0$ . In the following we use Greek letters to refer to single variables and Latin letters to refer to sets of variables. We let  $K = \Sigma^{-1}$  denote the inverse covariance, also known as the *concentration* with elements  $(k_{\alpha\beta})_{\alpha, \beta \in V}$ . The partial correlation between  $Y_\alpha$  and  $Y_\beta$  given all other variables is then

$$\rho_{\alpha\beta|V \setminus \{\alpha, \beta\}} = -k_{\alpha\beta} / \sqrt{k_{\alpha\alpha} k_{\beta\beta}}. \quad (1)$$

35 Thus  $k_{\alpha\beta} = 0$  if and only if  $Y_\alpha$  and  $Y_\beta$  are conditionally independent given all other variables.  
 36 A *graphical Gaussian model* (hereafter abbreviated GGM) is represented by an undirected  
 37 graph  $\mathcal{G} = (V, E)$  where  $V$  is a set of vertices representing the variables and  $E$  is a set of  
 38 undirected edges. The graph represents the model with  $K$  being a positive definite matrix  
 39 having  $k_{\alpha\beta} = 0$  whenever there is no edge between  $\alpha$  and  $\beta$  in  $\mathcal{G}$ .

### 40 2.3. RCON models – Restricted CONcentration models

An *RCON* model with vertex colour classes  $\mathcal{V}$  and edge colour classes  $\mathcal{E}$  is obtained by restricting the elements of  $K = \Sigma^{-1}$  further as follows: 1) All partial variances (i.e. all diagonal elements of  $K$ ) corresponding to vertices in the same vertex colour class must be identical. 2) All off-diagonal entries of  $K$  corresponding to edges in the same edge colour class must be identical. Thus, the diagonal of  $K$  can be specified by an  $R$  dimensional vector  $\eta$  while the off-diagonal elements are given by an  $S$  dimensional vector  $\delta$  so we can write  $K = K(\eta, \delta)$ . Figure 1 (a) thereby represents the concentration matrix

$$K = \begin{bmatrix} \eta_1 & \delta_1 & \delta_1 & 0 \\ \delta_1 & \eta_2 & 0 & \delta_2 \\ \delta_1 & 0 & \eta_2 & \delta_2 \\ 0 & \delta_2 & \delta_2 & \eta_1 \end{bmatrix}.$$

### 41 2.4. RCOR models – Restricted partial CORrelation models

42 An *RCOR* model with vertex classes  $\mathcal{V}$  and edge classes  $\mathcal{E}$  is obtained by restricting the  
 43 elements of  $K = \Sigma^{-1}$  as follows: 1) All partial variances corresponding to vertices in the same  
 44 vertex colour class must be identical. 2) All partial correlations corresponding to edges in the  
 45 same edge colour class must be identical.

As an RCOR model, Figure 1 (b) represents a concentration matrix  $K$  written as

$$K(\eta, \delta) = A(\eta)C(\delta)A(\eta),$$

where

$$A = \begin{bmatrix} \eta_1 & 0 & 0 & 0 \\ 0 & \eta_2 & 0 & 0 \\ 0 & 0 & \eta_3 & 0 \\ 0 & 0 & 0 & \eta_1 \end{bmatrix} \text{ and } C = \begin{bmatrix} 1 & \delta_1 & \delta_1 & 0 \\ \delta_1 & 1 & 0 & \delta_2 \\ \delta_1 & 0 & 1 & \delta_3 \\ 0 & \delta_2 & \delta_3 & 1 \end{bmatrix}.$$

46 Hence from (1),  $A$  contains the inverse partial standard errors on the diagonal while  $C$  contains  
 47 minus the partial correlations on the off-diagonal. The vertex colour classes of an RCOR  
 48 model is then restricting elements of  $A$  whereas the edge colour classes are restricting elements  
 49 of  $C$ .

## 3. Specifying and displaying models

### 50 3.1. Working data set: Mathematics marks

51 The `gRc` package will be illustrated on the basis of the following data set (taken from Mardia,  
 52 Kent, and Bibby (1979), see also Edwards (2000)). Data contains the examination marks for  
 53 88 students in 5 different mathematics subjects: Mechanics (`me`), Vectors (`ve`), Algebra (`al`),  
 54 Analysis (`an`) and Statistics (`st`). Data is contained the data set `math`. A stepwise backward  
 55 model selection yields the “butterfly” model shown in Figure 2, (a), see also Whittaker (1990),  
 56 p. 4.

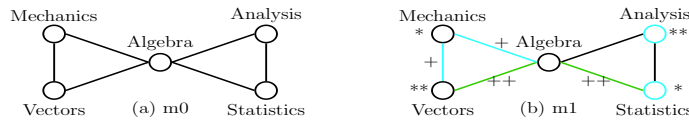


Figure 2: (a) The graphical Gaussian “butterfly” model selected for the mathmarks data. All vertices and edges are neutral, i.e. the parameters are unrestricted. In the following this model is called `m0`. (b) A representation of an RCON / RCOR model with restrictions on both vertices and edges. This model is denoted `m1` in the following.

### 57 3.2. Specifying the butterfly model – a GGM

58 Initially we specify the butterfly model for the mathmark data as a GGM which, by definition,  
 59 is also an RCON and RCOR model. The engine for specifying and fitting the models is the  
 60 `rcox` function which takes a `type` argument specifying the model type. The default model  
 61 type is `type='rcon'`.

62 In the following we shall show different ways of specifying models. For a GGM, the edge and  
 63 vertex colour classes can be specified indirectly by a generating class, e.g. the cliques of the  
 64 independence graph. For example, the butterfly model `m0` can be specified as:

```
m0 <- rcox(~me:ve:al + al:an:st, data = math)
```

```
RCON model: logL= -1278.991 dimension= 11 method= scoring time= 0.05
vcc: ~me, ~ve, ~al, ~an, ~st
ecc: ~me:ve, ~al:me, ~al:ve, ~al:an, ~al:st, ~an:st
```

65 Alternatively one can specify the vertex and edge colour classes (which are all atomic because  
 66 the model is a GGM) directly as:

```
m0 <- rcox(vcc = list(~me, ~ve, ~al, ~an, ~st), ecc = list(~me:ve,
  ~me:al, ~ve:al, ~al:an, ~al:st, ~an:st), data = math)
```

### 67 3.3. Mixed representations of RCON / RCOR models

68 In connection with model specification it is convenient to be able to work with a mixed  
 69 representation of a model as a triple  $(\mathcal{C}, \mathcal{V}, \mathcal{E})$  where  $\mathcal{C}$  is the generating class for a GGM. The  
 70 convention in connection with such a triple specification is as follows: 1)  $\mathcal{C}$  specifies vertices

71 and edges in the model. These are a priori unrestricted. 2)  $\mathcal{E}$  also specifies edges. Some of  
 72 these may already have been specified in  $\mathcal{C}$  but in that case restrictions in  $\mathcal{E}$  will be imposed.  
 73 3)  $\mathcal{V}$  also specifies vertices. Some of these may already have been specified in  $\mathcal{C}$  but in that  
 74 case restrictions in  $\mathcal{V}$  will be imposed.

### 75 3.4. Adding symmetry restrictions to the butterfly model

76 To illustrate RCON models which are not standard GGMs we impose the following restrictions  
 77 on `m0` to obtain `m1` (which is illustrated in Figure 2, (b)):

- 78 1. Vertices `me` and `st` are in the same vertex colour class and so are `ve` and `an`.
- 79 2. Edges `me:ve` and `me:al` are in the same edge colour class and so are `ve:al` and `al:st`.

80 The RCON model is fitted by

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

81 Note that here we have specified some of the edges through a generating class for a graphical  
 82 model, i.e. `~al:an:st`.

83 Setting `type='rcor'` in `rcox` will similarly fit the corresponding RCOR model:

```
m1c <- rcox(~al:an:st, vcc = list(~me + st, ~ve + an), ecc = list(~me:ve +
  me:al, ~ve:al + al:st), data = math, type = "rcor")
```

```
RCOR model: logL= -118.8656 dimension= 7 method= scoring time= 0.06
vcc: ~al, ~me + st, ~ve + an
ecc: ~al:an, ~an:st, ~me:ve + me:al, ~ve:al + al:st
```

### 84 *Retrieving vertex and edge colour classes*

85 The colour classes can be retrieved using the `getecc` and `getvcc` functions, e.g.

```
getecc(m1)
```

```
ecc1 ~al:an
ecc2 ~an:st
ecc3 ~me:ve + me:al
ecc4 ~ve:al + al:st
```

### 86 *Model summaries etc.*

87 Different type of model summaries are available. The default summary type is `type="coef"`  
 88 which gives a table with parameter estimates, standard errors etc:

```
summary(m1)
```

```
RCON model: logL= -1279.710 dimension= 7 method= scoring time= 0.03
vcc: ~al, ~me + st, ~ve + an
ecc: ~al:an, ~an:st, ~me:ve + me:al, ~ve:al + al:st

      cctype  cc      estimate      stderr      X2      p
vcc1  vcc vcc1  0.028096016  0.0036801167  58.286273  2.264855e-14
vcc2  vcc vcc2  0.005869607  0.0005849235  100.697789  0.000000e+00
vcc3  vcc vcc3  0.010044090  0.0009482858  112.187040  0.000000e+00
ecc1  ecc ecc1 -0.008025724  0.0015468068   26.921316  2.119089e-07
ecc2  ecc ecc2 -0.001763193  0.0007441495    5.614091  1.781662e-02
ecc3  ecc ecc3 -0.002957588  0.0004448611   44.200423  2.964173e-11
ecc4  ecc ecc4 -0.004738956  0.0008238733   33.086017  8.817053e-09

vcc1 ~al
vcc2 ~me + st
vcc3 ~ve + an
ecc1 ~al:an
ecc2 ~an:st
ecc3 ~me:ve + me:al
ecc4 ~ve:al + al:st
```

89 The reason for displaying the colour classes below the body of the table is that a colour class  
90 can consist of many edges/vertices thereby making the table very large.

91 The tests in the table are Wald tests for the corresponding parameters being zero. These  
92 tests only make sense for edge colour classes, but the standard errors for vertex colour classes  
93 are still informative for how precisely the parameters are estimated.

94 An alternative summary type is "KC" which when applied to the RCON model above gives

```
summary(m1, type = "KC")
```

```
RCON model: logL= -1279.710 dimension= 7 method= scoring time= 0.03
vcc: ~al, ~me + st, ~ve + an
ecc: ~al:an, ~an:st, ~me:ve + me:al, ~ve:al + al:st
      me      ve      al      an      st
me 0.07661336 -0.002957588 -0.002957588  0.000000000  0.000000000
ve 0.38519248  0.100220207 -0.004738956  0.000000000  0.000000000
al 0.23030890  0.282101238  0.167618661 -0.008025724 -0.004738956
an 0.00000000  0.000000000  0.477756429  0.100220207 -0.001763193
st 0.00000000  0.000000000  0.369024986  0.229636063  0.076613361
```

95 The fitted concentrations for edge colour classes appear above the diagonal (some of these are  
96 restricted to being identical under the model). The diagonal contains the fitted concentrations  
97 for for vertex colour classes, i.e. the partial variances (some of these are restricted to being  
98 identical under the model). Below the diagonal are the corresponding partial correlations.

99 When applied to the RCOR model above the summary type "KC" gives

```
summary(m1c, type = "KC")
```

```

RCOR model: logL= -118.8656 dimension= 7 method= scoring time= 0.06
vcc: ~al, ~me + st, ~ve + an
ecc: ~al:an, ~an:st, ~me:ve + me:al, ~ve:al + al:st
      me      ve      al      an      st
me 1.3185867 -0.5213360 -0.6737147 0.0000000 0.0000000
ve 0.2849471 1.3875378 -0.8754901 0.0000000 0.0000000
al 0.2849471 0.3518871 1.7930942 -1.0706683 -0.8319843
an 0.0000000 0.0000000 0.4303354 1.3875378 -0.4406480
st 0.0000000 0.0000000 0.3518871 0.2408454 1.3185867

```

100 As before, the fitted concentrations for edge colour classes appear above the diagonal. The di-  
 101 agonal contains the fitted concentrations for for vertex colour classes, i.e. the partial variances.  
 102 Below the diagonal are the corresponding partial correlations (of which some are restricted  
 103 to being identical under the model).

104 Other types of summaries are "K" and "ACA".

105 Standard methods like `coef` (for obtaining the parameter estimates) and `vcov` (for obtaining  
 106 the asymptotic variance for the estimators) are available.

107 The graph in Figure 3 is obtained by

```
plot(m1)
```

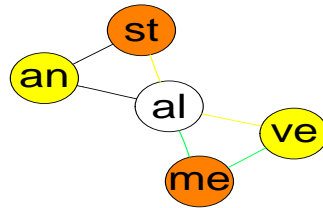


Figure 3: The display of the model `m1` produced by the `plot()` function.

## 4. Maximum likelihood estimation

108 This section describes estimation in RCON and RCOR models. See Højsgaard and Lau-  
 109 ritzen (2007) for discussion of problems concerning existence and uniqueness of estimators  
 110 and convergence properties of the algorithms.

### 4.1. Likelihood analysis of RCON models

112 We consider a sample  $y^1, \dots, y^n$  of  $n$  observations of  $Y \sim N_d(0, \Sigma)$  where  $d = |V|$  and  
 113  $\Sigma = K^{-1}$  and let  $W$  denote the matrix of sums of squares and products  $W = \sum_{\nu=1}^n Y^\nu (Y^\nu)^\top$ .  
 114 The log-likelihood function based on the sample is

$$\log L = \frac{f}{2} \log \det(K) - \frac{1}{2} \text{tr}(KW), \quad (2)$$

115 where in this case  $f = n$  is the degrees of freedom in the Wishart distribution of  $W$ . Taking  
 116 into account a possible unknown mean  $\mu$  and calculating  $W$  based on residuals would yield  
 117 degrees of freedom  $f = n - 1$ .

118 First we consider an RCON model  $(\mathcal{V}, \mathcal{E})$ . For each vertex colour class  $u \in \mathcal{V}$  let  $T^u$  be the  
 119  $d \times d$  diagonal matrix with entries  $T_{\alpha\alpha}^u = 1$  if  $\alpha \in u$  and 0 otherwise. For each edge colour  
 120 class  $u \in \mathcal{E}$  let  $T^u$  be the  $d \times d$  symmetrical matrix with entries  $T_{\alpha\beta}^u = 1$  if  $\{\alpha, \beta\} \in u$  and  
 121 0 otherwise. For convenience we shall identify a vertex  $\alpha$  with a set  $\{\alpha, \alpha\}$  such that vertex  
 122 classes and edge classes can be treated simultaneously in the following. Hence we can refer  
 123 to a generator  $u$  for an RCON model  $(\mathcal{V}, \mathcal{E})$  without specifying whether  $u$  is a vertex colour  
 124 class or an edge colour class. Consequently, we can rewrite  $(\eta, \delta)$  as  $\theta$  which is an  $R + S$   
 125 dimensional vector.

126 The concentration matrix  $K = K(\theta)$  can be written  $K = \sum_u \theta_u T^u$ . Letting  $t^u = \text{tr}(T^u W)$  we  
 127 have  $\text{tr}(KW) = \sum_u \theta_u \text{tr}(T^u W) = \sum_u \theta_u t^u$ . RCON models are thus linear exponential fami-  
 128 lies where  $(-t^1/2, \dots, -t^{S+T}/2)$  are canonical sufficient statistics and  $\psi(\theta) = -\frac{f}{2} \log \det(K)$  is  
 129 the logarithm of the normalising constant. The maximum likelihood estimate is unique and is  
 130 obtained by equating the canonical sufficient statistics to their expectation (Barndorff-Nielsen  
 131 1978).

132 Taking first and second derivatives of the logarithm of the normalising constant using that

$$\frac{\partial \det(M)}{\partial x} = \det(M) \text{tr}(M^{-1} \frac{\partial M}{\partial x}) \quad \text{and} \quad \frac{\partial M^{-1}}{\partial x} = -M^{-1} \frac{\partial M}{\partial x} M^{-1}$$

133 gives

$$\text{E}(-t^u/2) = \frac{\partial \psi}{\partial \theta_u} = -\frac{f}{2} \text{tr}(T^u \Sigma), \quad \text{Var}(-t^u/2) = \frac{\partial^2 \psi}{\partial \theta_u^2} = \frac{f}{2} \text{tr}(T^u \Sigma T^u \Sigma),$$

134 so the system of likelihood equations is

$$\text{tr}(T^u W) = f \text{tr}(T^u \Sigma), \quad u \in \mathcal{V} \cup \mathcal{E}. \quad (3)$$

## 135 4.2. Algorithms for estimation in RCON models

136 This section describes algorithms for estimation in RCON models. Let  $\hat{\Sigma} = \hat{K}^{-1}$  denote the  
 137 current estimate of  $\Sigma$  at any time during the iteration.

### 138 *Scoring algorithm*

139 The likelihood equations for RCON models can be solved using Fisher scoring if good starting  
 140 values can be found and if the Fisher information matrix is moderate in size. The algorithm,  
 141 however, is not globally convergent in general.

142 It is convenient to parametrise the model with  $\lambda_u = \log \eta_u$ . With this parametrisation,  
 143 differentiation of (2) yields the score function

$$S_u(\lambda, \delta) = \frac{f}{2} \begin{cases} (\text{tr}(T^u \Sigma) - \text{tr}(T^u W)/f) e^{\lambda_u} & \text{for } u \in \mathcal{V} \\ (\text{tr}(T^u \Sigma) - \text{tr}(T^u W)/f) & \text{for } u \in \mathcal{E} \end{cases} \quad (4)$$

144 Differentiating further and changing sign gives the Fisher information matrix

$$I(\lambda, \delta)_{uv} = \frac{f}{2} \begin{cases} \text{tr}(T^u \Sigma T^v \Sigma) e^{\lambda_u + \lambda_v} & \text{for } u, v \in \mathcal{V} \\ \text{tr}(T^u \Sigma T^v \Sigma) e^{\lambda_u} & \text{for } u \in \mathcal{V}, v \in \mathcal{E} \\ \text{tr}(T^u \Sigma T^v \Sigma) & \text{for } u, v \in \mathcal{E}. \end{cases} \quad (5)$$



145 The Fisher scoring step becomes

$$(\lambda, \delta) \leftarrow (\lambda, \delta) + I(\lambda, \delta)^{-1}S(\lambda, \delta) \quad (6)$$

146 which we found to sometimes be unstable in practice for RCON models.

147 [Jensen, Johansen, and Lauritzen \(1991\)](#) describe an estimation algorithm for linear expo-  
148 nential families, (see also [Lauritzen \(1996\)](#), p. 269), which applies Newton iteration to the  
149 reciprocal of the  $f$ th root of the likelihood function. This algorithm becomes

$$(\lambda, \delta) \leftarrow (\lambda, \delta) + [I(\lambda, \delta) + S(\lambda, \delta)S(\lambda, \delta)^\top / f]^{-1}S(\lambda, \delta). \quad (7)$$

150 This algorithm is globally convergent in the one-parameter case ([Jensen et al. 1991](#)). In the  
151 multi-parameter case the global convergence properties are unknown but empirical evidence  
152 suggests that it is quite stable and may be globally convergent.

153 Observe that omitting  $S(\lambda, \delta)S(\lambda, \delta)^\top / f$  from (7) will give Fisher scoring for  $(\lambda, \delta)$ . If we  
154 maximise the reciprocal likelihood itself instead of its  $f$ th root, the term  $S(\lambda, \delta)S(\lambda, \delta)^\top / f$  is  
155 replaced with  $S(\lambda, \delta)S(\lambda, \delta)^\top$ .

156 We further define the *discrepancy*  $\Delta(\lambda, \delta) = 2S(\lambda, \delta)/f$  where  $S$  is the score vector. Expressed  
157 in terms of  $\Delta$ , (7) becomes

$$(\lambda, \delta) \leftarrow (\lambda, \delta) + [2I(\lambda, \delta)/f + \Delta(\lambda, \delta)\Delta(\lambda, \delta)^\top / 2]^{-1}\Delta(\lambda, \delta). \quad (8)$$

158 Using the default `method='scoring'` in `gRc` for RCON models invokes (7) which can be seen  
159 as a stabilised version of Fisher scoring (6).

### 160 *Iterative partial maximisation*

161 [Jensen et al. \(1991\)](#) show that (7) is globally convergent in an exponential family if applied  
162 to one parameter at the time while keeping all other parameters fixed at their current values.

163 This iterative partial maximisation scheme works as follows for RCON models. Repeatedly  
164 loop through the elements of  $u \in \mathcal{V} \cup \mathcal{E}$  until convergence doing the following: The discrepancy  
165 is  $\Delta_u = \text{tr}(T^u \hat{\Sigma}) - \text{tr}(T^u W)/f = 2S_u/f$  and (8) for a single parameter becomes in this case

$$\theta_u^{n+1} \leftarrow \theta_u^n + Q_u, \text{ where } Q_u = \frac{\Delta_u}{\text{tr}(T^u \hat{\Sigma} T^u \hat{\Sigma}) + \Delta_u^2/2}. \quad (9)$$

166 The substitution (9) is repeated until convergence for the set  $u$  before moving on to the next  
167 set in  $\mathcal{V} \cup \mathcal{E}$ . Thus the algorithm consists of two nested loops: 1) An outer loop running over  
168 the elements  $u \in \mathcal{V} \cup \mathcal{E}$  and 2) an inner loop maximising  $L$  with respect to  $\theta_u$  while keeping  
169 all other parameters fixed.

170 Contrary to scoring, iterative partial maximisation does not directly produce the asymptotic  
171 variance-covariance matrix of the parameter estimates. However, after convergence the inverse  
172 Fisher information may be calculated.

173 To ensure global convergence, the substitution in (9) should only be effectuated if the new  
174 matrix  $K$  is positive definite. Otherwise the update should be made follows: Find the largest  
175 value of  $\lambda$  (where  $\lambda < 1$ ) for which an update  $\theta_u^n + \lambda Q_u$  would yield a positive semidefinite value  
176 of  $K$ , i.e. where the update would bring the parameter to the boundary of the parameter space.

177 The update should then be  $\theta_u^{n+1} \leftarrow \theta_u^n + \lambda Q_u/2$ . Thereby the update moves half the distance  
 178 towards the boundary of the parameter space. This refinement has not been implemented  
 179 and seems unnecessary. (We have not shown that the Newton steps are guaranteed to keep  
 180  $K$  positive definite but empirical evidence suggests that it is so.)

181 For some colour classes, the partial maximisation of the likelihood can be made explicitly  
 182 using a single step of the iterative proportional scaling (IPS) algorithm for graphical Gaussian  
 183 models, see e.g. Lauritzen (1996), p. 134. Thus for such colour classes the iterative scheme in  
 184 (9) needs not to be applied.

185 Consider a neutral set  $\{\alpha, \beta\}$ . The parameters  $k_{\alpha\alpha}, k_{\beta\beta}$  and  $k_{\alpha\beta}$  are not restricted other than  
 186 through  $K$  being positive definite. In this case these parameters can be updated with a single  
 187 IPS step, i.e. without using the Newton method. Let  $a = \{\alpha, \beta\}$ , let  $b$  denote the complement  
 188 to  $a$ , let  $K_{aa}$  be the  $2 \times 2$  submatrix of  $K$  comprising  $k_{\alpha\alpha}, k_{\beta\beta}$  and  $k_{\alpha\beta}$ , and let  $K_{ab}$  and  $K_{bb}$   
 189 be defined similarly. The likelihood equations are that  $(K_{aa} - K_{ab}(K_{bb})^{-1}K_{ba})^{-1} = W_{aa}/f$   
 190 which are solved by setting

$$K_{aa} \leftarrow (W_{aa}/f)^{-1} + K_{ab}(K_{bb})^{-1}K_{ba}. \quad (10)$$

191 This IPS step maximises the likelihood over the particular section of the parameter space  
 192 given by  $k_{\alpha\alpha}, k_{\beta\beta}$  and  $k_{\alpha\beta}$  and thus no iteration is needed. This IPS step can be applied for  
 193 any neutral set  $a \subset V$ . A special case is for a single parameter  $k_{\alpha\alpha}$  (i.e. for an atomic vertex  
 194 colour class).

195 An IPS step can also be used for estimating a single parameter  $k_{\alpha\beta}$  where  $\alpha \neq \beta$ , i.e. for  
 196 an atomic edge colour class: Following the notation above, let  $B = K_{ab}(K_{bb})^{-1}K_{ba}$ . The  
 197 likelihood equations state that  $\{(K_{aa} - B)^{-1}\}_{\alpha\beta} = W_{\alpha\beta}/f$  where  $\{A\}_{\alpha\beta}$  is the off-diagonal of  
 198 a symmetric  $2 \times 2$  matrix  $A$  with entries indexed by  $\alpha$  and  $\beta$ . This yields the following 2nd  
 199 degree equation in  $k_{\alpha\beta}$ :

$$\frac{-(k_{\alpha\beta} - B_{\alpha\beta})}{(k_{\alpha\alpha} - B_{\alpha\alpha})(k_{\beta\beta} - B_{\beta\beta}) - (k_{\alpha\beta} - B_{\alpha\beta})^2} = W_{\alpha\beta}/f. \quad (11)$$

200 By inspection of the equation it can be seen that only one of the solutions leads to a positive  
 201 definite  $K$ , and the solution is

$$k_{\alpha\beta} \leftarrow B_{\alpha\beta} + \frac{1 - \sqrt{1 + 4W_{\alpha\beta}(k_{\alpha\alpha} - B_{\alpha\alpha})(k_{\beta\beta} - B_{\beta\beta})}}{2W_{\alpha\beta}/f}. \quad (12)$$

202 The IPM algorithm is illustrated in an example below. In this connection it is convenient to  
 203 work with a *mixed representation* of an RCON model which is a 3-tuple,  $(\mathcal{V}, \mathcal{E}, \mathcal{N})$  where  $\mathcal{N}$   
 204 is a set of neutral sets.

**Example 1** The graph in Figure 4 has vertex and edge colour classes

$$\mathcal{V} = [1][2, 3][4][5] \quad \mathcal{E} = (1:2, 2:3)(3:4)(4:5),$$

and specifies a RCON model with restrictions  $k_{22} = k_{33}$  and  $k_{12} = k_{23}$ . The vertex colour  
 classes [4] and [5] are both atomic and so is the edge colour class (4, 5) so  $\{4, 5\}$  is a neutral

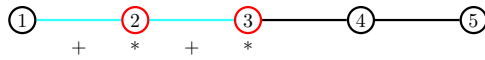


Figure 4: Representation of the RCON model with the restrictions that  $k_{22} = k_{33}$  and  $k_{12} = k_{23}$ .

set. Algorithmically, we add  $\{4, 5\}$  to  $\mathcal{N}$ , remove  $(4:5)$  from  $\mathcal{E}$ , and remove  $[4]$  and  $[5]$  from  $\mathcal{V}$ . This yields the *mixed representation*

$$\mathcal{V} = [1][2, 3], \quad \mathcal{E} = (1:2, 2:3)(3:4), \quad \mathcal{N} = \{4, 5\}.$$

205 One full cycle of the outer loop of the IPM algorithm goes as follows:

206 **1.**  $[1]$  is an atomic vertex colour class so  $k_{11}$  can be updated with a single IPS step (10) on a  
207  $1 \times 1$  matrix.

208 **2.**  $[2, 3]$  is a composite vertex colour class and  $(1:2, 2:3)$  is a composite edge colour class.  
209 Hence  $k_{22} = k_{33}$  and  $k_{12} = k_{23}$  must be updated separately with the Newton sequence  
210 (9). So this step is computationally demanding.

211 **3.**  $(3:4)$  is an atomic edge colour class so  $k_{34}$  can be updated with a single IPS step (12) on  
212 the off-diagonal of a  $2 \times 2$  matrix.

213 **4.**  $\{4, 5\}$  is a neutral set so all three parameters  $k_{44}, k_{55}$  and  $k_{45}$  can be updated in a single  
214 IPS step (10) on a  $2 \times 2$  matrix.

215 To avoid complex book keeping we have not exploited that  $\{4, 5\}$  is a neutral set and can be  
216 fitted as such in the current version of `gRc`. Instead **4.** above is replaced with

217 **4a.** Update in turn  $k_{44}, k_{55}$  and  $k_{45}$  with a single IPS step (10).

218 The `method='ipm'` for RCON models is used for the scheme where IPS is applied whenever  
219 possible (i.e. for atomic colour classes) and where (9) is applied for all composite colour classes.

## 220 *Computational savings*

221 The following considerations lead to additional substantial computational savings:

222 1. The matrices  $T^u$  are symmetrical matrices consisting of zeros and ones. It is not nec-  
223 essary (and becomes prohibitive in terms of space requirements for high dimensional  
224 models) to actually create these matrices. When calculating traces like e.g.  $\text{tr}(T^u \hat{\Sigma} T^u \hat{\Sigma})$   
225 and related quantities it is sufficient to identify relevant entries of  $\hat{\Sigma}$  etc. to be added  
226 up.

227 2. After updating entries of  $K$ , it is not necessary to find  $\Sigma = K^{-1}$ . The relevant part of  
228  $\Sigma_{aa}$  is  $(K_{aa} - K_{ab}(K_{bb})^{-1}K_{ba})^{-1}$ . Note here 1) that  $K_{ab}(K_{bb})^{-1}K_{ba}$  is fixed throughout  
229 the whole Newton sequence and 2) that the dimension of  $\Sigma_{aa}$  is often much smaller than  
230 the dimension of  $\Sigma$ .

231 *Starting values*

232 A starting value for  $K$  for both the scoring and iterative partial maximisation methods is  
 233 found on the basis of the method of score matching (Hyvärinen 2005). The score matching  
 234 estimate  $\tilde{K}$  is obtained by solving a linear system of  $R + S$  equations with  $R + S$  unknowns  
 235 where  $R + S$  is the number of parameters in  $K$ . Hence  $\tilde{K}$  can be calculated very easily. While  
 236 the score matching estimate is asymptotically unbiased, is not guaranteed to be positive  
 237 definite. Moreover, we have not shown that diagonal elements of  $\tilde{K}$  are indeed positive.

238 If  $\tilde{K}$  is positive definite then  $\tilde{K}$  is taken to be the initial value of  $K$ . If  $\tilde{K}$  is not positive definite  
 239 we make it so as follows: If the diagonal elements of  $\tilde{K}$  are not all positive we apply score  
 240 matching to a model with the same vertex colour classes as the model under consideration  
 241 but with no edge colour classes. This yields an estimate  $\tilde{\tilde{K}}$ , which is also the MLE under this  
 242 hypothesis and is formed by taking simple averages over corresponding diagonal elements of  
 243  $W$ . Hence  $\tilde{\tilde{K}}$  has positive diagonal elements. We then replace the diagonal elements of  $\tilde{K}$  by  
 244 the corresponding elements of  $\tilde{\tilde{K}}$ .

245 If the modified  $\tilde{K}$  is still not positive definite, let  $\text{diag}(\tilde{K})$  be the diagonal matrix with diagonal  
 246 entries being the diagonals of  $\tilde{K}$ . Starting from  $\alpha = 0.95$  and working downwards in steps of  
 247 0.05 we search the largest  $0 < \alpha < 1$  such that  $K_\alpha = \text{diag}(\tilde{K}) + \alpha(\tilde{K} - \text{diag}(\tilde{K}))$  is positive  
 248 definite. To obtain numerical stability we then set  $\alpha \leftarrow 0.95\alpha$  and calculate  $K_\alpha$  again and  
 249 take this as the initial value of  $K$ .

250 Setting `method='matching'` means that we first find an initial estimate of  $K$  as described  
 251 above and then perform one iteration of the the scoring algorithm (7). This yields a fast  
 252 estimate of  $K$  which is efficient to the first order.

253 *Comparison of the estimation methods*

254 The scoring method is in general somewhat faster than iterative partial maximisation, but  
 255 iterative partial maximisation will tend to be more economical in terms of space requirements.

256 **4.3. Likelihood analysis of RCOR models**

257 For an RCOR model  $(\mathcal{V}, \mathcal{E})$  we write  $K(\eta, \delta) = A(\eta)C(\delta)A(\eta)$ . Then  $A$  is diagonal and  
 258 consists of the inverse partial standard deviations while  $C$  has ones on the diagonal and will  
 259 contain minus the partial correlations on the off diagonals. The log likelihood is

$$\log L = \frac{f}{2} \log \det(C) + f \log \det(A) - \frac{1}{2} \text{tr}(ACAW). \quad (13)$$

260 **4.4. Algorithms for estimation in RCOR models**

261 This section describes two iterative algorithms for estimation in RCOR models.

262 *Scoring algorithm*

263 As for RCON models, Fishers method of scoring can be applied for solving the likelihood  
 264 equations. It is convenient to parametrise the model with  $\lambda_u = \log \eta_u$  in which case the score  
 265 becomes

$$S_u(\lambda, \delta) = \begin{cases} f \text{tr}(T^u) - \text{tr}(T^u A C A W) & \text{for } u \in \mathcal{V} \\ f \text{tr}(T^u C^{-1})/2 - \text{tr}(T^u A W A)/2 & \text{for } u \in \mathcal{E}. \end{cases} \quad (14)$$

266 Differentiating further and changing sign yields the observed information matrix

$$J(\lambda, \delta)_{uv} = \begin{cases} 2 \operatorname{tr}(T^u A W A T^v C) & \text{for } u, v \in \mathcal{V}, u = v \\ \operatorname{tr}(T^u A W A T^v C) & \text{for } u, v \in \mathcal{V}, u \neq v \\ \operatorname{tr}(T^u A W A T^v) & \text{for } u \in \mathcal{V}, v \in \mathcal{E} \\ \frac{f}{2} \operatorname{tr}(T^u C^{-1} T^v C^{-1}) & \text{for } u, v \in \mathcal{E}. \end{cases} \quad (15)$$

267 Taking expectations gives the Fisher information matrix,

$$I(\lambda, \delta)_{uv} = \begin{cases} 2f \operatorname{tr}(T^u C^{-1} T^v C) & \text{for } u, v \in \mathcal{V}, u = v \\ f \operatorname{tr}(T^u C^{-1} T^v C) & \text{for } u, v \in \mathcal{V}, u \neq v \\ f \operatorname{tr}(T^u C^{-1} T^v) & \text{for } u \in \mathcal{V}, v \in \mathcal{E} \\ \frac{f}{2} \operatorname{tr}(T^u C^{-1} T^v C^{-1}) & \text{for } u, v \in \mathcal{E}. \end{cases} \quad (16)$$

268 Using `method='scoring'` for RCOR models invokes the iteration (7) with score and infor-  
 269 mation given by (14) and (16). For RCOR models we have found that the iteration (7)  
 270 can lead to a decrease of the log likelihood. When this occurs, the step size  $[I(\lambda, \delta) +$   
 271  $S(\lambda, \delta)S(\lambda, \delta)^\top / f]^{-1} S(\lambda, \delta)$  is repeatedly halved until the log likelihood has increased.

#### 272 *Iterative partial maximisation*

273 Contrary to RCON models, the restrictions on the concentration matrix are in general not  
 274 linear in  $\eta$  and  $\delta$  for RCOR models. However, for known  $\eta$ , the restrictions are linear in  $\delta$  and  
 275 for known  $\delta$ , the restrictions are quadratic in  $\eta$ . This suggests to estimate the parameters by  
 276 alternating between  $\eta$  and  $\delta$  as follows:

- 277 1. Suppose that  $C$  is known, i.e. that  $\delta$  is known. Then we maximize  $\log L$  over  $\eta$ . Max-  
 278 imising  $\log L$  over a given  $\eta_u$  keeping the other  $\eta_s$  fixed yields a 2nd order equation  
 279 which has a unique positive root. Note that  $\eta_u$  depends on the remaining  $\eta_s$ . Therefore,  
 280 we must iterate to solve for  $\eta$ . For the specific form of these equations we refer to  
 281 [Højsgaard and Lauritzen \(2007\)](#).
2. Suppose that  $A$  is known, i.e. that  $\eta$  is known and let  $Q = AWA$ . Then  $\operatorname{tr}(ACAW) =$   
 $\operatorname{tr}(CQ)$  and  $\log L$  can be maximized over  $\delta$  by maximising

$$\log L(\delta) = \frac{f}{2} \log |C| - \frac{1}{2} \operatorname{tr}(CQ)$$

282 This maximisation can be made by applying the IPM algorithm for RCON models to the  
 283 off-diagonal elements of  $C$  only, letting  $Q$  play the role as  $W$  in the likelihood equations  
 284 for RCON models. That is, the diagonal elements of  $C$  remain constantly equal to one.  
 285 Atomic edge colour classes are updated with an IPS step and composite edge colour  
 286 classes are updated with the modified Newton algorithm.

287 The `method='ipm'` in `gRc` for RCOR models is used for the scheme where IPS is applied  
 288 whenever possible (i.e. for neutral sets and for atomic edge colour classes) and where (9) is  
 289 applied for all composite colour classes.

290 *Starting values*

291 Starting values for RCOR are obtained as follows: First rescale data to have unit variance  
 292 and find an initial estimate  $\check{K}$  by score matching as if the model were an RCON model, see  
 293 Section 4.2 for details.

294 Next we force  $\check{K}$  to satisfy the RCOR model as follows: Let  $\text{diag}(\check{K})$  be the diagonal matrix  
 295 with diagonal entries being the diagonals of  $\check{K}$ . Since  $K = ACA$  we set  $\check{A} = \sqrt{\text{diag}(\check{K})}$  and  
 296  $\check{C} = \check{A}^{-1}\check{K}\check{A}^{-1}$ , i.e. we rescale  $\check{K}$  to have ones on the diagonal to obtain  $\check{C}$ . Elements of  $\check{A}$   
 297 and  $\check{C}$  which under the model are restricted to being identical are replaced by their average.  
 298 Now  $\check{A}$  is necessarily positive definite. If also  $\check{C}$  is positive definite then we take the starting  
 299 value of  $K$  to be  $\check{A}\check{C}\check{A}$ .

300 If  $\check{C}$  is not positive definite then adopt the same strategy as for RCON models by rescaling  
 301 the off-diagonals of  $\check{C}$  towards zero until a positive definite matrix  $C_\alpha$  is obtained. We then  
 302 take the starting value of  $K$  to be  $\check{A}C_\alpha\check{A}$ .

303 As for RCON models one can set `method='matching'` which means that we first find an initial  
 304 estimate of  $K$  as described above and then perform one iteration of the scoring algorithm (7).  
 305 Note that the estimated covariances of the parameter estimates may be misleading.

306 *Comparison of the estimation methods*

307 For RCOR models the scoring method tends to be slightly faster than iterative partial max-  
 308 imisation.

## 5. Model editing

309 Before discussing further statistical aspects of `gRc` we shall in this section describe methods  
 310 for modifying RCOX models. The modification is made by the `update` function. The new  
 311 model is fitted if the original model is fitted unless `fit=FALSE` is specified to the `update`  
 312 function. To explicitly fit a model, use the `fit()` function.

313 **5.1. Joining and splitting colour classes**

314 Colour classes can be joined and split using the `update` function. For example, joining the  
 315 edge colour classes `an:st` and `me:ve + me:al` can be achieved by:

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

```
RCON model: logL= -1281.271 dimension= 6 method= scoring time= 0.03
vcc:  ~al, ~me + st, ~ve + an
ecc:  ~al:an, ~ve:al + al:st, ~an:st + me:ve + me:al
```

316 These colour classes are number 2 and 3 in the list of edge colour classes, cfr. Section 3.4.  
 317 They can hence also be joined by

```
update(m1, joinecc = getecc(m1)[c("ecc2", "ecc3")])
```

318 This approach is more convenient if e.g. programming a model selection strategy. Likewise a  
 319 vertex colour class can be split by

```
update(m1, splitvcc = ~ve + an)
```

320 or by

```
update(m1, splitvcc = getvcc(m1)["vcc3"])
```

## 321 5.2. Adding and dropping edge colour classes

322 Adding an edge colour class corresponds to adding a set of edges to the graph and forcing  
 323 these to be in the same colour class. Dropping an edge colour class corresponds to deleting  
 324 a set of edges from the graph. (Corresponding operations on vertex colour classes make no  
 325 sense.) For example:

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

## 6. Methods for comparison of colour classes

326 This section describes methods for investigating model reductions. That is 1) investigate if  
 327 the parameters for two colour classes  $u$  and  $v$  in a model are significantly different (so that  $u$   
 328 and  $v$  can be joined) and 2) investigate if the parameter for an edge colour class  $u$  in a model  
 329 is significantly different from zero (so that  $u$  can be dropped). We also described methods  
 330 investigating model expansions. That is 1) investigate if a composite colour class can be split  
 331 into atomic colour classes and 2) investigate if an atomic edge colour class can be added to  
 332 the model.

333 The output of these methods is a list with two components: 1) a data frame with the results  
 334 of the tests and 2) a list of the colour classes.

### 335 6.1. Model comparison

**Model reductions** Consider a model  $\mathcal{M}_0$  and two colour classes  $u$  and  $v$  (of the same type) in  $\mathcal{M}_0$ . The feasibility of joining  $u$  and  $v$  can be judged by the likelihood ratio statistic. This requires fitting a new model  $\mathcal{M}_1 \subset \mathcal{M}_0$  in which  $u$  and  $v$  are joined. Then  $\mathcal{M}_1$  can be tested against  $\mathcal{M}_0$  with a deviance (likelihood ratio) test statistic

$$D = -2 \log LR_{10} = -2 \log \frac{L_1}{L_0}.$$

To avoid fitting the model  $\mathcal{M}_1$ , one can instead use the Wald statistic

$$W = \frac{(\hat{\theta}_u - \hat{\theta}_v)^2}{\text{Var}(\hat{\theta}_u - \hat{\theta}_v)}.$$

336 The asymptotic variance of the difference is obtained from the inverse Fisher information  
 337 matrix. Both statistics have under the hypothesis approximately a  $\chi_d^2$  distribution where  
 338  $d = df_1 - df_0$ . The same alternatives are available for testing if the parameter  $\theta_u$  for an edge  
 339 colour class is zero. This generalises immediately to cases where several colour classes are  
 340 compared simultaneously.

The comparisons can also be made using AIC (Akaike 1974) and BIC (Schwarz 1978). With reference to the setup above,  $\mathcal{M}_1$  is preferred to  $\mathcal{M}_0$  (according to BIC) if

$$\Delta BIC = -D + (df_0 - df_1) \log n > 0,$$

341 otherwise  $\mathcal{M}_0$  is preferred. If using AIC,  $\log n$  is replaced by 2 above. Because the Wald and  
 342 the deviance statistics are asymptotically equivalent we can calculate an approximation to  
 343 AIC/BIC by replacing  $D$  with  $W$ .

344 Functions for making model reductions accept a `stat` keyword. Default is `stat='wald'`  
 345 because it is the fastest to calculate.

346 **Model expansions** Consider a model  $\mathcal{M}_0$  and a composite colour class  $u$  in  $\mathcal{M}_0$ . The  
 347 feasibility of splitting  $u$  can be judged by the likelihood ratio statistic. This requires fitting a  
 348 new model  $\mathcal{M}_1 \supset \mathcal{M}_0$  in which  $u$  is split. The deviance becomes  $D = -2 \log(L_0/L_1)$ . When  
 349 models are expanded, the Wald statistic is not available as the larger model has not been  
 350 fitted to data. Apart from that everything else is as above.

351 Consequently, functions for model expansion are all based on the deviance statistic and do  
 352 therefore not accept a `stat` keyword.

## 353 6.2. Model reductions

354 Pairwise comparisons of edge/vertex colour classes can be made using the `comparecc` function.  
 355 To compare two specific edge colour classes using the deviance statistic do:

```
ctab <- comparecc(m1, cc1 = list(~me:ve + me:al, ~ve:al +
  al:st), cc2 = list(~an:st, ~al:an), type = "ecc", stat = "dev")
```

```
Comparing colour classes of type: ecc using statistic: dev
  cc1 cc2      X2 df      p      aic      bic
1 ecc1 ecc1  3.122960  1 0.0771964605 -1.122960  1.3543773
2 ecc1 ecc2 11.989965  1 0.0005348778 -9.989965 -7.5126287
3 ecc2 ecc1  5.430822  1 0.0197843675 -3.430822 -0.9534849
4 ecc2 ecc2  4.798558  1 0.0284835737 -2.798558 -0.3212208

cc1:
ecc1 ~me:ve + me:al
ecc2 ~ve:al + al:st
cc2:
ecc1 ~an:st
ecc2 ~al:an

Available components: tab cc1 cc2
```

356 According to this table, the colour classes `ecc1` from `cc1` and `ecc1` from `cc2` are not signifi-  
 357 cantly different according to a significance test and BIC.



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

362 **Joining colour classes** Joining colour classes leads to a model reduction. The `join1`  
 363 function is essentially a wrapper for `comparecc`. Based on the Wald statistic (the default)  
 364 the pairwise comparisons of the colour classes of a specific type are made. The set of colour  
 365 classes under consideration can be restricted using the `scope` argument (default is that all  
 366 colour classes are considered) e.g.

```
join1(m1, scope = list(~an:st, ~me:ve + me:al, ~ve:al + al:st),
      type = "ecc")
```

```
Comparing colour classes of type: ecc using statistic: wald
```

|   | cc1  | cc2  | X2       | df | p          | aic       | bic        |
|---|------|------|----------|----|------------|-----------|------------|
| 1 | ecc1 | ecc2 | 3.011035 | 1  | 0.08269946 | -1.011035 | 1.4663017  |
| 2 | ecc1 | ecc3 | 5.180254 | 1  | 0.02284499 | -3.180254 | -0.7029171 |
| 3 | ecc2 | ecc3 | 3.318470 | 1  | 0.06850555 | -1.318470 | 1.1588667  |

```
cc1:
ecc1 ~an:st
ecc2 ~me:ve + me:al
ecc3 ~ve:al + al:st
cc2:
ecc1 ~an:st
ecc2 ~me:ve + me:al
ecc3 ~ve:al + al:st
```

```
Available components: tab cc1 cc2
```

367 **Dropping edge colour class** It is possible to test if the parameter for an edge colour  
 368 class is zero. (Such a test makes no sense for vertex colour classes). The set of edge colour  
 369 classes under consideration can be restricted using the `scope` argument (default is that all  
 370 edge colour classes are considered) e.g.

```
drop1(m1, scope = list(~al:an, ~an:st, ~me:ve + me:al))
```

```
Statistic: wald
```

|   | cc   | X2        | df | p            | aic        | bic        |
|---|------|-----------|----|--------------|------------|------------|
| 1 | ecc1 | 26.921316 | 1  | 2.119089e-07 | -24.921316 | -22.443979 |
| 2 | ecc2 | 5.614091  | 1  | 1.781662e-02 | -3.614091  | -1.136754  |
| 3 | ecc3 | 44.200423 | 1  | 2.964173e-11 | -42.200423 | -39.723086 |

```
cc:
ecc1 ~al:an
ecc2 ~an:st
ecc3 ~me:ve + me:al
```

```
Available components: tab cc
```

### 371 6.3. Model expansions

372 **Splitting colour classes** Splitting a (composite) colour class leads to a model expansion.  
 373 To investigate if edge colour class in `m1` can be split do:

```
split1(m1, scope = list(~ve:al + al:st, ~me:ve + me:al),
       type = "ecc")
```

```

  cc      X2 df      p      aic      bic
1 ecc1 0.2306087  1 0.6310729  1.769391  4.246728
2 ecc2 0.1719902  1 0.6783491  1.828010  4.305347

cc:
ecc1 ~ve:al + al:st
ecc2 ~me:ve + me:al

Available components: tab cc
```

374 Thus there is no evidence that splitting either of the composite edge colour classes would  
 375 significantly enhance the fit of the model. Note that splitting a composite colour class into  
 376 atomic colour classes can lead to fairly large increase in the model complexity, i.e. in the  
 377 number of parameters in the model.

378 **Adding edge colour class** In the same spirit one can make a test for addition of (atomic)  
 379 edge colour classes to the model:

```
add1(m1)
```

```

  cc      X2 df      p      aic      bic
1 ecc1 0.2475697  1 0.6187915 -1.752430 -4.229767
2 ecc2 0.1480575  1 0.7003987 -1.851943 -4.329279
3 ecc3 0.9819775  1 0.3217111 -1.018023 -3.495359
4 ecc4 0.2666198  1 0.6056083 -1.733380 -4.210717

cc:
ecc1 ~an:me
ecc2 ~me:st
ecc3 ~an:ve
ecc4 ~st:ve

Available components: tab cc
```

380 Hence there is no evidence that adding additional (atomic) edge colour classes to the model  
 381 would significantly enhance the fit of the model.

## 7. Stepwise procedures

382 This section contains a description of functions for stepwise model selection. The output of  
 383 these methods is either a new model object or NULL if no change was made. These functions  
 384 are based on repeated applications of the functions described in Section 6, and therefore their  
 385 arguments are similar. Default is that the selection criterion is AIC.

### 386 7.1. The need for selection strategies

387 The number of different models which can be formed by colouring edges/vertices in a given  
 388 graph is enormous. To illustrate the complexity, consider graphs with three vertices (for which  
 389 there are 8 different graphs). A tedious enumeration shows that there are in total (over all 8  
 390 graphs) 15 possible edge colour classes. There are 5 possible vertex colour classes which gives  
 391  $5 \times 15 = 75$  different models. Therefore, good model selection strategies become important.  
 392 This section discusses methods which would be part of model selection strategies; however  
 393 much additional work is required in this area.

### 394 7.2. Nested versus non-nested models

395 It should be noted that addition of an edge to a coloured graph can have different meanings:  
 396 The model  $M_0$  in Figure 5 is given by  $[1][2][3](1:2, 2:3)$ . Consider addition of the edge  $2:3$  in  
 397  $M_0$ . If a new colour class is formed as in  $M_1$  then  $M_1$  will contain  $M_0$  and these two models  
 398 can be tested e.g. with a deviance test. An alternative is to add the edge to the already  
 399 existing colour class as in  $M_2$ . In that case,  $M_0$  and  $M_2$  are not nested. Likewise dropping  
 400 e.g.  $1:2$  from  $M_0$  does not lead to a model reduction. A model comparison can however be  
 401 made in terms of AIC or BIC.

402 The methods described in the following all act within nested models and hence AIC and BIC  
 403 as well as significance testing can be used as selection criteria.

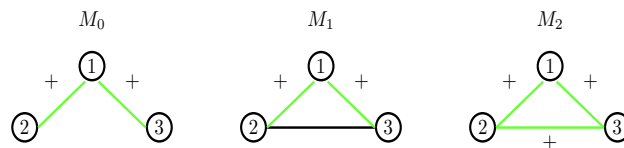


Figure 5: Illustration of addition of the edge  $2:3$  to a coloured graph. If the edge is added to  $M_0$  by forming a new edge colour class as in  $M_1$ , then  $M_0$  and  $M_1$  are nested and can be tested. Alternatively, the edge can be added to an existing edge colour class as in  $M_2$ . The models  $M_0$  and  $M_2$  are then not nested.

### 404 7.3. Aspects of stepwise procedures

405 **Stepwise joining of the two most homogeneous colour classes** In RCOX models,  
 406 model reductions can be achieved by joining colour classes. Suppose there are  $p$  colour classes  
 407 of a given type, e.g. edge colour classes. The number of ways these can be combined for  
 408 joining is enormous. Therefore we consider only to join pairs of colour classes and there are  
 409  $p(p-1)/2$  such pairs to consider. We say we join the two most homogeneous colour classes.  
 410 The `stepjoin1` function facilitates doing this in a stepwise fashion.

411 **Stepwise dropping of the least significant edge colour class** Model reductions can  
 412 also be achieved by dropping edge colour classes, which is the counterpart to dropping in-  
 413 significant edges in GGMs. The `stepdrop1` function facilitates doing this.

414 **Stepwise splitting of the most heterogeneous composite colour class** In RCOX  
 415 models, model expansions can be achieved by splitting composite colour classes. Consider a  
 416 composite colour class with  $p$  elements. The number of colour classes it can be split into is in  
 417 general very large, so therefore we consider only the operation of splitting into atomic colour  
 418 classes. This is done for each composite colour class of a given in turn. The colour class who,  
 419 if split, gives the the largest improvement in model fit (i.e. the smallest  $p$ -values (smaller than  
 420 the critical level) in terms of significance testing) is said to the most heterogeneous. This leads  
 421 to splitting of the most heterogeneous colour class. If a colour class with  $p$  elements is split  
 422 into atoms there dimension of the model will increase by  $p - 1$ . The `stepsplit1` function  
 423 facilitates doing this.

424 **Stepwise addition of the most significant atomic edge colour class** Model expan-  
 425 sions can also be achieved by adding edge colour classes to a model in a stepwise fashion.  
 426 This is the counterpart to stepwise addition of edges in GGMs. If  $p$  edges missing from the  
 427 graph the number of possible edge colour classes which can be formed becomes enormous –  
 428 even for small  $p$ . Therefore we only consider adding the most significant atomic edge colour  
 429 class. The `stepadd1` function facilitates doing this.

#### 430 7.4. Stepwise model reductions

431 **Stepwise join of colour classes** Starting from the butterfly model `m0` we first join vertex  
 432 colour classes and then join edge colour classes afterwards:

```
m01 <- stepjoin1(m0, type = "vcc")
```

```
Stepwise joining of colour classes of type: vcc using statistic: wald ;criterion: aic alpha: 0
Joining: ~ve; ~an
  X2: 0.059888 df: 1 p: 0.806673 aic: 1.940112 bic: 4.417449
Joining: ~me; ~st
  X2: 0.954332 df: 1 p: 0.328619 aic: 1.045668 bic: 3.523005

RCOx model: logL= -1279.506 dimension= 9 method= scoring time= 0.04
vcc: ~al, ~ve + an, ~me + st
ecc: ~me:ve, ~me:al, ~ve:al, ~al:an, ~al:st, ~an:st
```

```
m02 <- stepjoin1(m01, type = "ecc")
```

```
Stepwise joining of colour classes of type: ecc using statistic: wald ;criterion: aic alpha: 0
Joining: ~me:ve; ~me:al
  X2: 0.175196 df: 1 p: 0.675534 aic: 1.824804 bic: 4.302140
Joining: ~ve:al; ~al:st
  X2: 0.229890 df: 1 p: 0.631605 aic: 1.770110 bic: 4.247447

RCOx model: logL= -1279.710 dimension= 7 method= scoring time= 0.03
vcc: ~al, ~ve + an, ~me + st
ecc: ~al:an, ~an:st, ~me:ve + me:al, ~ve:al + al:st
```

433 The resulting model (which is identical to `m2` in Section 3) is shown in Figure 6.

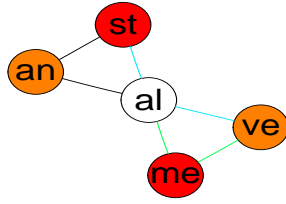


Figure 6: The model obtained after 1) first successively joining vertex colour classes and 2) then successively joining edge colour classes.

434 **Stepwise drop of edge colour classes** Dropping edge colour classes leads to a model  
 435 reduction. The `stepdrop1` function tests for deletion of each edge colour class in the model  
 436 and deletes the least significant of these. Using 0.01 as significance level we can do:

```
stepdrop1(m1, criterion = "test", alpha = 0.01)
```

437 In this case, the function returns NULL because it is not feasible to drop any of the edge colour  
 438 classes.

## 439 7.5. Stepwise model expansions

440 **Stepwise split of composite colour classes** The split operation for colour classes can  
 441 be applied in a stepwise fashion. The model in Figure 7, left is given by:

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

442 Splitting first vertex colour classes gives the middle graph in Figure 7. Continuing and  
 443 splitting edge colour classes gives the rightmost graph:

```
m3 <- stepsplit1(m2, type = "vcc")
```

```
Stepwise splitting of colour classes, type: vcc
criterion: aic alpha: 0
Splitting: ~al + me + st
  X2: 85.408451 df: 2 p: 0.000000 aic: -81.408451 bic: -76.453777

RCOON model: logL= -1284.651 dimension= 6 method= scoring time= 0.03
vcc: ~ve + an, ~al, ~me, ~st
ecc: ~me:ve + me:al + ve:al, ~al:an + al:st + an:st
```

```
m4 <- stepsplit1(m3, type = "ecc")
```

```

Stepwise splitting of colour classes, type: ecc
criterion: aic alpha: 0
Splitting: ~al:an + al:st + an:st
X2: 8.028886 df: 2 p: 0.018053 aic: -4.028886 bic: 0.925788

RCON model: logL= -1280.637 dimension= 8 method= scoring time= 0.05
vcc: ~ve + an, ~al, ~me, ~st
ecc: ~me:ve + me:al + ve:al, ~al:an, ~al:st, ~an:st

```

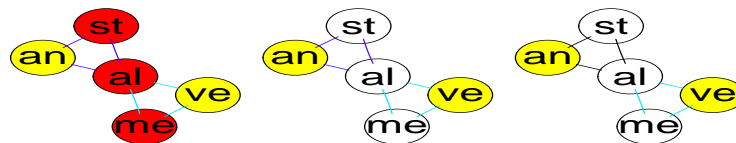


Figure 7: Left: Starting model. Middle: Model after splitting vertex colour classes. Right: Model after also splitting edge colour classes

444 **Stepwise addition of (atomic) edge colour classes** Adding an (atomic) edge colour  
445 class leads to a model expansion. The `stepadd1` function will take any edge not in the model  
446 and try to add. The edge with the smallest  $p$ -value (not larger than  $\alpha$ ) will be added. The  
447 test made here is always a deviance test:

```
stepadd1(m1, criterion = "test")
```

448 In this case, the function returns NULL because it is not feasible to add any edge colour classes.

## 8. Discussion and perspectives

449 We have described an R package `gRc` for statistical inference in RCON and RCOR models.  
450 These models have been described in some detail, including a description of various algorithms  
451 for maximum likelihood estimation. For further details on the models and their properties  
452 we refer to [Højsgaard and Lauritzen \(2007\)](#).

453 The facilities of this package cover model editing functions, functions for comparing colour  
454 classes and stepwise model selection functions. These facilities are described. We have also  
455 presented some examples of how to use the package.

456 Improvements of `gRc` can be made in several directions of which we outline some here:

457 i) The current implementation of `gRc` is made entirely in R. The computing time could be  
458 reduced by implementing larger parts of the algorithms, in particular the iterative partial  
459 maximisation, in a compiled language. Further, it would be desirable to develop faster faster  
460 estimation algorithms such that `gRc` can be applied to problems of higher dimension.

461 ii) Additional model selection criteria and strategies in RCON and RCOR models must be  
462 investigated further and implemented.

463 Finally, `gRc` should be integrated more closely with other packages in created in the `gR`  
 464 initiative; most importantly the `gRbase` package, [Dethlefsen and Højsgaard \(2005\)](#) and the  
 465 `ggm` package, [Marchetti and Drton \(2006\)](#).

## 9. Acknowledgements

466 This work has been supported by a grant financed by the Directorate for Food, Fisheries and  
 467 Agro Business, Denmark, Lattec I/S, the Danish Cattle Association and the Danish Institute  
 468 of Agricultural Sciences and by the Danish Natural Science Research Council.

## A. Additional aspects of the package

469 This section describes a few additional aspects of the package.

### 470 A.1. Controlling the estimation methods

471 The `rcox` function is controlled by the `control` argument which is a list with named entries.  
 472 The details of this list is given in the documentation of the `rcox` function. Here we mention  
 473 a few important issues.

474 The iterations in the scoring and iterative partial maximisation methods are controlled as  
 475 follows: With iterative partial maximisation there are an outer and an inner loop, cfr. Sec-  
 476 tion 4.2.2 and the maximum number of iterations are controlled by setting e.g. `maxouter=10`  
 477 and `maxinner=5` in the control list. The number of iterations for the scoring method is  
 478 controlled by `maxouter`.

479 A colour class is essentially either a list of edges or a list of vertices. For high dimensional  
 480 models these lists can be very long and displaying them on the screen can be confusing.  
 481 Setting `short=TRUE` in the control list implies that the colour classes are not printed. Note  
 482 that the colour classes can however be retrieved using the `getvcc` and `getecc` functions.

483 The methods described in Sections 6 and 7 can all be given a `details` keyword. Default  
 484 is `details=1` which produces a reasonable amount of output. Increasing `details` produces  
 485 more output while setting `details=0` suppresses all output.

### 486 A.2. Deferring model fitting

487 Default for `rcox` as well as for all functions described later for editing models is that the model  
 488 is fitted. To avoid fitting, one can supply `rcox` with the argument `fit=FALSE`. To explicitly  
 489 fit a model use the function `fit`.

### 490 A.3. Specification of colour classes in different forms

491 Colour classes can be specified as a list of formulae, as well as a list of lists. For example,

```

rcox(vcc = list(~me + ve + al, ~st), data = math)
rcox(vcc = list(list("me", "ve", "al"), list("st")), data = math)
```

492 both represent the same models with restrictions on the vertices. Likewise,

```
rcox(ecc = list(~me:ve + me:al, ~ve:al), data = math)
rcox(ecc = list(list(c("me", "ve"), c("me", "al")), list(c("ve",
"al"))), data = math)
```

493 specify the same models with restrictions on the edges. The representation as a list of lists is  
 494 convenient in connection with programming an automatic model search strategy.

495 Following these conventions the scope for the functions in Section 6 and 7 can be represented  
 496 in two different ways. For example:

```
add1(m1, scope = list(c("an", "me"), c("me", "st")))
add1(m1, scope = list(~an:me, ~me:st))
```

## References

- 497 Akaike H (1974). “A New Look at the Statistical Model Identification.” *IEEE Transactions*  
 498 *on Automatic Control*, **19**(6), 716–723.
- 499 Anderson TW (1970). “Estimation of covariance matrices which are linear combinations or  
 500 whose inverses are linear combinations of given matrices.” In RC Bose, IM Chakravarti,  
 501 PC Mahalanobis, CR Rao, KJC Smith (eds.), “Essays in Probability and Statistics,” pp.  
 502 1–24. University of North Carolina Press, Chapel Hill, N.C.
- 503 Barndorff-Nielsen OE (1978). *Information and Exponential Families in Statistical Theory*.  
 504 Wiley, New York.
- 505 Dempster AP (1972). “Covariance Selection.” *Biometrics*, **28**, 157–175.
- 506 Dethlefsen C, Højsgaard S (2005). “A Common Platform for Graphical Models in R: The  
 507 gRbase Package.” *Journal of Statistical Software*, **14**, 1–12.
- 508 Edwards D (2000). *Introduction to Graphical Modelling*. Springer Verlag, New York, 2nd  
 509 edition edition.
- 510 Højsgaard S, Lauritzen SL (2005). “Patterned graphical Gaussian models with concentration  
 511 parameters restricted to being equal.” In “Proceedings of AISTATS 2005,” Barbados.
- 512 Højsgaard S, Lauritzen SL (2007). “Graphical Gaussian Models with Edge and Vertex Sym-  
 513 metries.” *Manuscript*.
- 514 Hyvärinen A (2005). “Estimation of Non-Normalized Statistical Models by Score Matching.”  
 515 *Journal of Machine Learning Research*, **6**, 695–709.
- 516 Jensen ST, Johansen S, Lauritzen SL (1991). “Globally Convergent Algorithms for Maximizing  
 517 a Likelihood Function.” *Biometrika*, **78**(4), 867–877.
- 518 Lauritzen SL (1996). *Graphical Models*. Oxford University Press.



- 519 Lauritzen SL (2002). “gRaphical Models in R: A new initiative within the R project.” *Rnews*,  
520 **2**, 39.
- 521 Marchetti GM, Drton M (2006). *ggm: Graphical Gaussian Models*. R package version 1.0.2.
- 522 Mardia KV, Kent JT, Bibby JM (1979). *Multivariate Analysis*. Academic Press.
- 523 R Development Core Team (2007). *R: A Language and Environment for Statistical Comput-*  
524 *ing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3, URL  
525 <http://www.R-project.org>.
- 526 Schwarz G (1978). “Estimating the Dimension of a Model.” *Annals of Statistics*, **6**, 461–464.
- 527 Whittaker J (1990). *Graphical Models in Applied Multivariate Statistics*. Wiley.

528 **Affiliation:**

529 Søren Højsgaard  
530 Institute of Genetics and Biotechnology  
531 Faculty of Agricultural Sciences  
532 Aarhus University  
533 8830 Tjele, Denmark  
534 E-mail: [sorenh@agrsci.dk](mailto:sorenh@agrsci.dk)  
535 URL: <http://gbi.agrsci.dk/~sorenh/>  
536

537 Steffen L. Lauritzen  
538 Department of Statistics  
539 University of Oxford  
540 1 South Parks Road  
541 Oxford OX1 3TG  
542 United Kingdom  
543 E-mail: [steffen@stats.ox.ac.uk](mailto:steffen@stats.ox.ac.uk)  
544 URL: <http://www.stats.ox.ac.uk/~steffen/>