Chapter 7
High Dimensional Modelling

7.1 Introduction

This chapter describes and compares some methods available in R for selecting and working with high-dimensional graphical models. By 'high-dimensional' we are thinking of models with hundreds to tens of thousands of variables. Modelling such data has become of central importance in molecular biology and other fields, but is challenging. Many graph-theoretic operations scale poorly: for example, finding the cliques of a general undirected graph is known to be NP-hard. Model selection algorithms that work well in low dimensional applications may be quite infeasible for high dimensional ones. There can be statistical as well as algorithmic limitations: for example, for high-dimensional Gaussian data with modest numbers of observations, maximum likelihood estimates will not exist for complex models. Generally it is necessary to assume that relatively simple models are adequate to model high-dimensional data.

In Sect. 7.2 two example datasets are described. In Sect. 7.3 some model selection algorithms available in R are compared in respect to their scalability. Sections 7.4, 7.5 and 7.6 describe the use of some of the more scalable methods in more detail. Finally, in Sect. 7.7 we describe a Bayesian approach, showing how to identify the MAP (maximum a posteriori) forest for high-dimensional discrete data.

7.2 Two Datasets

We illustrate the methods in this chapter using two datasets. The first is supplied along with **gRbase** and is taken from a study comparing gene expression profiles in tumours taken from two groups of breast cancer patient, namely those with and those without a mutation in the p53 tumour suppression gene. See Miller et al. (2005) for a further description of the study.

```
> data(breastcancer)
> dim(breastcancer)
```

S. Højsgaard et al., *Graphical Models with R*, Use R!,
DOI 10.1007/978-1-4614-2299-0_7, © Springer Science+Business Media, LLC 2012

```
[1]  250 1001
> table(sapply(breastcancer, class))

 factor numeric
      1    1000
> table(breastcancer$code)

   case control
     58     192
```

There are $N = 250$ observations and 1001 variables, comprising 1000 continuous variables (the log-transformed gene expression values) and one binary factor, code. There are 58 cases (with a p53 mutation) and 192 controls (without the mutation).

The second dataset comes from a large multinational project to study human genetic variation, the HapMap project (http://www.hapmap.org/). The dataset concerns a sample of 90 Utah residents with northern and western European ancestry, the so-called CEU population, and contains information on genetic variants and gene expression values for this sample. The genetic variants are SNPs (single nucleotide polymorphisms), that is to say, individual bases at particular loci on the genome that show variation in the population. In statistical terms SNPs are categorical variables with three levels—two homozygotes and a heterozygote. Around 10 million SNPs have been identified in humans. Datasets containing both SNP and gene expression data enable study of the the genetic basis for differences in gene expression. The data from this sample are supplied along with the package **GGtools** in the BioConductor repository. The code

```
> data(hmceuB36.2021)
> k <- 200
> ggdata <- data.frame(hmceuB36.2021$male,
+   as(smList(MAFfilter(hmceuB36.2021, lower=.1))
+   [["21"]][,1:k], "character"), t(exprs(hmceuB36.2021))[,1:k])
> ggdata[,1:(k+1)] <- lapply(ggdata[,1:(k+1)], factor)
```

loads an object `hmceuB36.2021` containing SNP data from chromosomes 20 and 21, gene expression data and other phenotypic information recorded for individuals in the sample. In all it contains data on 199921 SNPs on chromosome 20, 50165 on chromosome 21, and expression values for 47293 genes. The above code fragment creates a dataframe `ggdata` by extracting the individuals sex, the first 200 SNPs from chromosome 21 and the first 200 log-transformed gene expression values from `hmceuB36.2021`. Prior to extraction the SNPs are filtered so that SNPs with a minimum allele frequency of less than 10% are discarded. Some values of the SNPs are missing, but here the missing values are coded as a distinct character value, so the SNPs are factors with up to four levels. The last line converts the discrete variables into factors.

7.3 Computational Efficiency

A thorough study of the computational efficiency of the algorithms described in this book would be a huge and complex task. In this section, we report on some timings

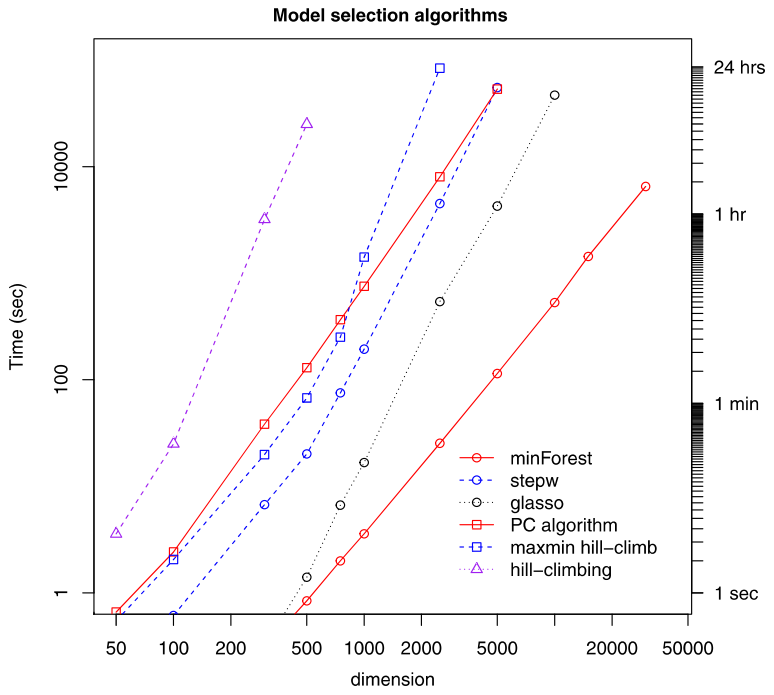


Fig. 7.1 Timing comparisons

of the algorithms when applied to a specific dataset, in a specific computing environment. It is hoped that this will give at least a rough impression of their relative efficiency on similar datasets in other computing environments.

The algorithms were applied to data from the HapMap project described in Sect. 7.2. For various values of the dimension p, the first p gene expression values were used. So there were 90 cases and p Gaussian variables, where p ranges from 50 to 50000. Three algorithms to select undirected graphical models, and four to select (equivalence classes of) graphical models based on DAGs were compared. The computations were run under Redhat Fedora 10 Linux on a Intel i7 four-core 2.93 GHz machine with 48 GB RAM. The timings are shown in Fig. 7.1.

The undirected model selection methods were:

(i) The extended Chow-Liu algorithm, implemented in the `minForest()` function in the **gRapHD** package, that finds the minimum BIC forest. This is further described in Sect. 7.4.

(ii) A greedy decomposable search algorithm, implemented in the `stepw()` function in the **gRApHD** package, that seeks (but is not guaranteed to find) the minimum BIC decomposable model. See Sect. 7.5 below. Here the minimum BIC forest is used as initial model.

(iii) The `glasso()` function in the **glasso** package described in Sect. 4.4.2. Here the tuning parameter $\rho = 0.2$ is used.

The DAG selection methods were:

(iv) The PC-algorithm implemented in the `pc()` function in the **pcalg** package, as described in Sect. 4.6.1. Here $\alpha = 0.05$ is used.
(v) The hill-climbing algorithm implemented in the `hc()` function in the **bnlearn** package, as described in Sect. 4.6.2.1.
(vi) The max-min hill-climbing algorithm implemented in the `mmhc()` function in the **bnlearn** package, a hybrid constraint- and score-based algorithm, described above in Sect. 4.6.2.2. Here $\alpha = 0.05$ is used.

Note that (i) and (ii) return decomposable models, which represent equivalence classes of DAGs (see Sect. 4.5.1), so these can also be regarded as DAG selection methods.

We note that the algorithms for undirected models are more efficient than those for directed models. The most efficient of the latter is the pc-algorithm: however, when $p = 5000$, this takes approximately 24 hours whereas the extended Chow–Liu algorithm takes about 1 minute for these data.

7.4 The Extended Chow–Liu Algorithm

In a paper predating much of the theoretical development of graphical models, Chow and Liu (1968) described an algorithm to find the maximum likelihood tree model for multivariate discrete data. In modern terminology, tree models are discrete graphical models whose graphs are trees. Trees and forests are special cases of undirected graphs. A forest is an acyclic undirected graph, that is, an undirected graph with no cycles. A tree is a connected acyclic undirected graph. So a forest may have several connected components, these being trees. Chow and Liu showed that finding the maximum likelihood tree can be formulated as finding a maximum weight spanning tree—a task for which highly efficient algorithms exist. Their approach requires, first, that all edge weights are calculated, and then a maximum weight spanning tree algorithm is applied to find a maximum weight spanning tree. (This may be non-unique if there are ties in the edge weights.)

Usually the algorithm due to Kruskal (1956) is used to find the maximum weight spanning tree. This starts with the null graph and successively selects edges $e_1, \ldots, e_r$. If edges $e_1, \ldots, e_k$ have been selected, the algorithm selects an edge e such that

(a) $e \notin \{e_1, \ldots e_k\}$ and $\{e_1, \ldots e_k, e\}$ is a forest, and
(b) e has maximum weight among all edges satisfying (a).

Chow and Liu's approach may be extended in various ways (Edwards et al. 2010):

- It can be applied to Gaussian data using appropriate weights.
- By modifying the weights appropriately it can be adapted to find the minimal AIC or BIC forest. If this has several connected components we can analyze these separately—a dimension reduction that can be very useful with high-dimensional problems.

- It can be applied to mixed discrete and Gaussian data by modifying the weights appropriately and limiting the search space in (a) to strongly decomposable forests, that is, forests containing no forbidden paths. Recall that a forbidden path is a path between non-adjacent discrete nodes passing through continuous nodes. This restriction implies that in each tree of the forest, the discrete nodes induce a connected subgraph.
- In the conditional Chow–Liu algorithm (Kirshner et al. 2004) the search space is extended to graphs that include a given set of edges, E_0 say. Formally, the search space becomes

$$\{\mathcal{G} = (V, E) : E_0 \subseteq E \wedge \text{any cycle in } \mathcal{G} \text{ has all edges in } E_0\}$$

  To do this, the algorithm starts off from $\mathcal{G}_0 = (V, E_0)$ and (a) is modified to restrict candidate edges to those that do not create *new* cycles.

The extended algorithm is implemented in the `minForest()` function in the **gRapHD** package. It requires the data to be supplied as a dataframe with discrete and/or continuous variables. The discrete variables must be represented as factors. For example, we can apply it to the `breastcancer` dataframe as follows:

```
> bF <- minForest(breastcancer)
> bF

gRapHD object
Number of edges       = 1000
Number of vertices    = 1001
Model                 = mixed and homogeneous
Statistic (minForest) = BIC
Statistic (stepw)     =
Statistic (user def.) =
Edges (minForest)     = 1...1000
Edges (stepw)         = 0...0
Edges (user def.)     = 1...1000
```

Per default, the `minForest` function returns the minimal BIC forest, in the form of a `gRapHD` object. Note that `bF` has 1001 nodes and 1000 edges. Since a forest with n nodes and k connected components has $n - k$ edges, we see that `bF` is a tree: all nodes are interconnected.

These `gRapHD` objects are essentially undirected graphs represented in node and edge list form, in which nodes are identified by their column numbers in the input dataframe. They also contain information on variable types (discrete or continuous) and names (which are used to label the nodes in plots). They may be displayed using the `plot` function

```
> plot(bF)
```

but here, plotting a high-dimensional graph like `bF` would not be a good idea: no structure would be visible. Instead, since we are primarily interested in the effect of the mutation on gene expression, let us look at the neighbourhood of the discrete variable, `code`. This is the last column of `breastcancer`, column number 1001. The following two lines of code extract the nodes of `bF` whose path length from `code` is less than or equal to 4, and then display the subgraph of `bF` induced by these nodes.

```
> nby <- neighbourhood(bF, orig=1001, rad=4)$v[,1]
> plot(bF, vert=nby, numIter=1000)
```

The `plot()` function, when applied to `gRapHD` objects, shows discrete variables as dots and continuous variables as circles. It uses the iterative layout algorithm of Fruchterman and Reingold (1991): here we specify 1000 iterations to get a clear layout. We see that the effect of the mutation on gene expression appears to be mediated by its effect on the expression of gene `A.202870_s_at`. To find out more about this gene we can google this string, from which we learn that under the alias CDC20 the gene "... appears to act as a regulatory protein interacting with several other proteins at multiple points in the cell cycle. It is required for two microtubule-dependent processes, nuclear movement prior to anaphase and chromosome separation." In other words, it is involved in cell division. Below, using strongly decomposable models, we re-examine the hypothesis that the effect of p53 mutation on gene expression is mediated by its effect on the expression of this gene.

The following code illustrates the extended Chow-Liu approach applied to the `ggdata` dataset. The minimal BIC forest is obtained using the `minForest()` function:

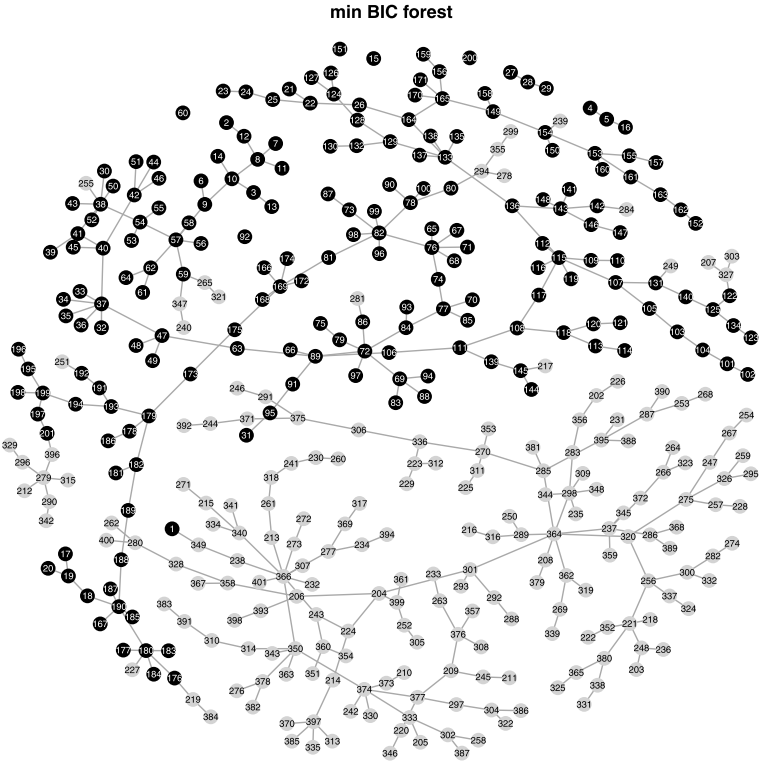
```
> ggF <- minForest(ggdata)
> ggF
```

```
gRapHD object
Number of edges       = 392
Number of vertices    = 401
Model                 = mixed and homogeneous
Statistic (minForest) = BIC
Statistic (stepw)     =
Statistic (user def.) =
Edges (minForest)     = 1...392
Edges (stepw)         = 0...0
Edges (user def.)     = 1...392
```

```
> table(Degree(ggF))

  0   1   2   3   4   5   6   7   8  10
  5 198 103  49  23   7  10   4   1   1
```

```
> plot(ggF, numIter=500, vert.labels=1:ggF@p, main="min BIC forest")
```



We see that ggF is a forest with $401 - 392 = 9$ connected components. The `Degree()` function returns a vector containing the degree (number of adjacent nodes) of each node: we see that there are 5 isolated nodes. We can identify the components by converting the `gRapHD` object to a `graphNEL` object using the `as()` function, and then applying the `connComp()` function, which returns a list of components.

```
> cc <- connComp(as(ggF, "graphNEL"))
> sapply(cc, length)

[1] 172 218   3   1   3   1   1   1   1
```

The 9 connected components consist of two large components (with 172 and 218 nodes), two components with 3 nodes, and 5 isolated nodes. If we look at the two largest components

```
> intersect(cc[[1]], names(ggdata)[1:201])

[1] "hmceuB36.2021.male"

> length(intersect(cc[[2]], names(ggdata)[1:201]))

[1] 189
```

we see that the first contains only one discrete variable (sex), but the second contains 189 SNPs and 29 gene expression variables.

7.5 Decomposable Stepwise Search

In a celebrated paper, Chickering (1996) showed that identifying the Bayesian network that maximizes a score function is in general a NP-hard problem, and it is reasonable to suppose that this is also true of undirected graphical models (Markov networks). However, there are ways to improve computational efficiency. A useful approach is to restrict the search space to models with explicit estimates, the decomposable models. The following key result is exploited: if $\mathcal{M}_0 \subset \mathcal{M}_1$ are decomposable models differing by one edge $e = \{u, v\}$ only, then e is contained in one clique C of $\mathcal{M}_1$ only, and the likelihood ratio test for $\mathcal{M}_0$ versus $\mathcal{M}_1$ can be performed as a test of $u \perp\!\!\!\perp v | C \setminus \{u, v\}$. These computations only involve the variables in C. It follows that for likelihood-based scores such as AIC or BIC, score differences can be calculated locally—which is far more efficient then fitting both $\mathcal{M}_0$ and $\mathcal{M}_1$—and then stored, indexed by u, v and C, so that they can be reused again if needed in the course of the search. This can lead to considerable efficiency gains.

The `stepw()` function in the **gRapHD** package implements forward search through decomposable models to minimize the AIC or BIC. At each step, the edge giving the greatest reduction in AIC or BIC is added. A convenient choice of start model is the minimal AIC/BIC forest, but an arbitrary decomposable start model may be used. We illustrate use of this function by resuming the analysis of the breast cancer dataset. The minimal BIC forest for the neighbourhood of the `code` variable is obtained as follows.

```
> bc.marg <- breastcancer[,nby]
> mbF <- minForest(bc.marg)
> plot(mbF, numIter=1000)
```

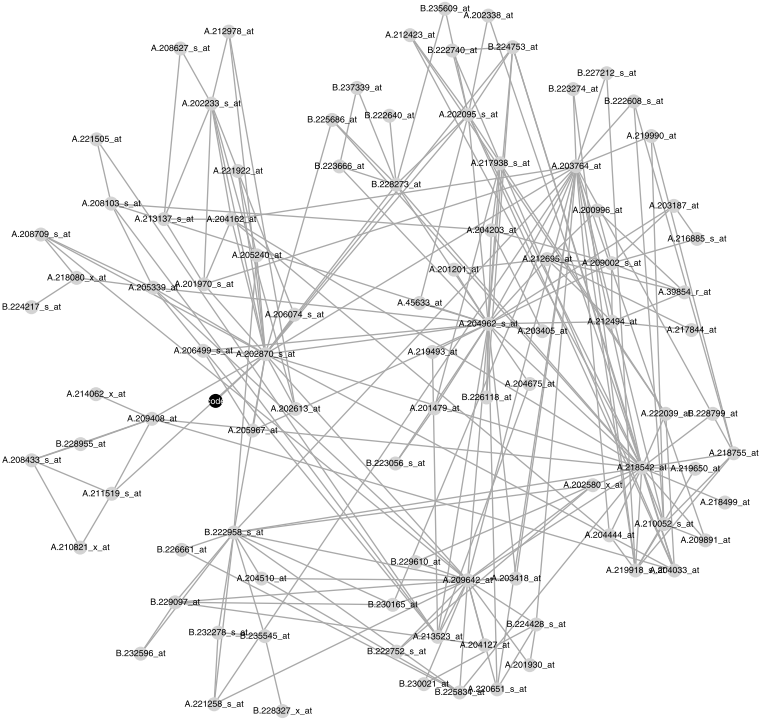
The gene adjacent to `code` is `A.202870_s_at` (CDC20) as we saw before. This suggests that the effect of p53 mutation on gene expression is mediated by its effect on CDC20. However, this might be a consequence of adopting this restrictive—and sparse—model class. It is interesting to expand the search space to decomposable models. The minimal BIC decomposable model is obtained using the `stepw()` function:

```
> mbG <- stepw(model=mbF, data= bc.marg)
> mbG

gRapHD object
Number of edges       = 225
Number of vertices    = 94
Model                 = mixed and homogeneous
Statistic (minForest) = BIC
Statistic (stepw)     = BIC
Statistic (user def.) =
Edges (minForest)     = 1...93
Edges (stepw)         = 94...225
Edges (user def.)     = 1...93
```

To plot `mbG` using the same layout as in the previous plot, we store the node coordinates from the previous plot and reuse them when plotting `mbG`, as follows:

```
> posn <- plot(mbF, numIter=1000, disp=F)
> plot(mbG, numIter=0, coord=posn)
```



Although the minimal BIC decomposable model is considerably less sparse, the interpretation is unaltered: it still appears that the effect of p53 mutation on gene expression is mediated by its effect on the expression of CDC20.

An interesting aspect of this example is the presence of so-called *hub* genes—nodes of high degree—that may play a key role in the regulatory network. If we compare the degree distributions of the two graphs

```
> table(Degree(mbF))

 1  2  3  4  5  6 12 17
68 12  5  1  2  2  3  1

> table(Degree(mbG))

 1  2  3  4  5  6  7  8  9 12 15 21 22 25 29
 7 28 21 10  8  1  8  3  1  1  1  1  1  2  1

> Degree(mbF)[Degree(mbF)>4]

 2  6 13 18 20 24 27 31
12 17  6  6  5 12 12  5

> Degree(mbG)[Degree(mbF)>4]

 2  6 13 18 20 24 27 31
25 29 15 12 21 22 25  9
```

we see that the hub genes in the—presumably more realistic—graph `mbG` are reliably identified using the forest `mbF`.

7.6 Selection by Approximation

Here we illustrate use of the graphical lasso algorithm of Friedman et al. (2008) described in Sect. 4.4.2. We apply it to the breastcancer dataset (omitting the discrete class variable since the algorithm is only applicable to Gaussian data).

```
> S <- cor(breastcancer[,nby[-1]])
> res.lasso <- glasso(S, rho=0.8)
> AM <- res.lasso$wi != 0
> diag(AM) <- F
> rownames(AM) <- colnames(AM) <- names(breastcancer)[nby[-1]]
> g.lasso <- as(AM, "graphNEL")
> g.lasso
```

```
A graphNEL graph with undirected edges
Number of Nodes = 93
Number of Edges = 198
```

```
> g.HD <- as(g.lasso, "gRapHD")
> plot(g.HD, numIt=1000)
```

The example selects a model to the variables in the neighbourhood of the `code` variable in the breast cancer dataset (omitting the `code` variable itself since it is discrete). We apply the `glasso()` function to the expirical correlation matrix of the variables, in effect standardizing the variables to unit variance. Note that since the glasso procedure is not scale invariant, this is normally a sensible step. As penalty

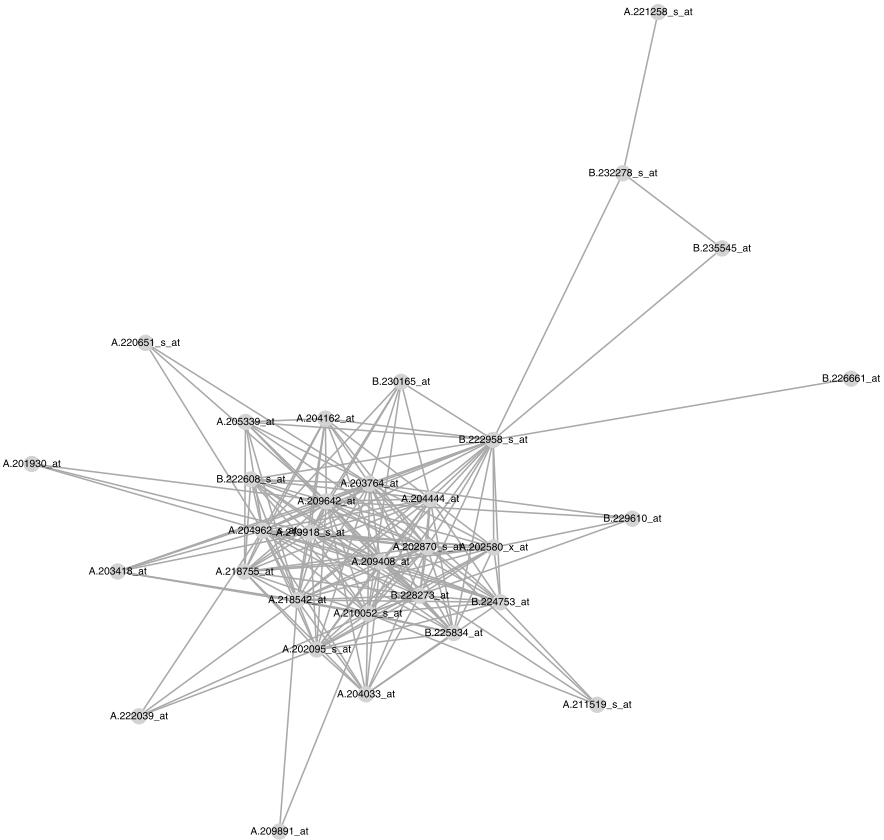
parameter we use $\rho = 0.8$: this choice was made so as to obtain a graph of comparable density to those obtained previously. The `glasso()` function returns a list containing the estimated inverse covariance matrix `wi`. Note that the method combines model selection with parameter estimation. In the code fragment shown we derive the adjacency matrix from the inverse covariance and use this to construct a `graphNEL` graph of the model. The diagonal elements of the adjacency matrix are set to false to omit self-edges from the graph.

The graph selected by the algorithm contains a module of interconnected variables and a large number of isolated ones. To see the former more closely, we can use the following code:

```
> cc <- connComp(g.lasso)
> sapply(cc, length)
```

```
 [1] 32  1  1  2  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2
[23]  1  2  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
[45]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

```
> g.sub <- subGraph(cc[[1]], g.lasso)
> g.subHD <- as(g.sub, "gRapHD")
> plot(g.subHD, numIt=1000)
```



7.7 Finding MAP Forests

In this section we describe a Bayesian equivalent to the minimal AIC/BIC forest approach described in Sect. 7.4. This builds on the framework of Dawid and Lauritzen (1993). We write a collection of d discrete random variables as $X = (X_v)_{v \in V}$, and we write a generic observation as $i = (i_1, \ldots, i_d)$, and the observed data as $\mathcal{X} = (i^v, v = 1, \ldots, N)$. We are interested in a collection of graphs Γ with vertex set V. For each graph $\mathcal{G} \in \Gamma$, let $\Theta_\mathcal{G}$ be the associated parameter space, and $\mathcal{L}_\mathcal{G}$ be a prior distribution (or law) on $\Theta_\mathcal{G}$. Then the marginal likelihood of $\mathcal{G}$ is

$$f(\mathcal{X}|\mathcal{G}) = \int_{\Theta_\mathcal{G}} f(\mathcal{X}|\mathcal{G}, \theta) \mathcal{L}_\mathcal{G}(d\theta).$$

If $\pi(\mathcal{G})$ is the prior probability of $\mathcal{G}$, then the posterior probability is given as

$$\pi^*(\mathcal{G}) = \pi(\mathcal{G}|\mathcal{X}) \propto f(\mathcal{X}|\mathcal{G})\pi(\mathcal{G}). \tag{7.1}$$

The maximum a posteriori (MAP) estimate is the graph in Γ that maximizes $\pi^*(\mathcal{G})$.

We now sketch how a prior distribution $\mathcal{L}_\mathcal{G}$ on $\Theta_\mathcal{G}$ may be chosen. Consider first an unconstrained multinomial distribution on an array with parameters $p = (p(i))_{i \in \mathcal{I}}$, and let $\lambda = (\lambda(i))_{i \in \mathcal{I}}$ be an array of positive numbers. The Dirichlet distribution $D(\lambda)$ has density

$$\pi(p|\lambda) \propto \prod p(i)^{\lambda(i)-1}.$$

If the prior distribution of p is $D(\lambda)$ and counts $n = (n(i))_{i \in \mathcal{I}}$ are observed, then the posterior distribution of p is $D(\lambda + n)$: in other words, the Dirichlet distribution is the conjugate prior of the multinomial. The numbers λ are called the equivalent sample size, or smoothing parameter.

Dawid and Lauritzen (1993) generalize this to construct the conjugate prior for a decomposable graphical model $\mathcal{G}$, which they term the *hyper-Dirichlet* distribution. Essentially this involves specifying a Dirichlet prior for each clique of $\mathcal{G}$. Let $\mathcal{C} = (C_1, \ldots, C_k)$ be these cliques. Thus a hyper-Dirichlet prior is specified though the collection of arrays $(\lambda_C)_{C \in \mathcal{C}}$. These must satisfy a consistency criterion, namely that for all cliques $C, D \in \mathcal{C}$, $\lambda_C(i_{C \cap D}) = \lambda_D(i_{C \cap D})$ for all cells $i_{C \cap D}$. Without loss of generality we can specify the λ_C's by specifying a $\lambda = (\lambda(i))_{i \in \mathcal{I}}$ for the whole array and setting λ_C to the marginal totals $\lambda_C(i_C) = \sum_{j \in \mathcal{I}: j_C = i_C} \lambda(j)$. This construction automatically fulfills the consistency criteria. It also allows the array λ to function as a 'master-prior' to specify the smoothing parameters for the hyper-Dirichlet prior for the parameters for *any* decomposable model.

Dawid and Lauritzen (1993) also show that for a hyper-Dirichlet prior, the marginal likelihood factorizes in a fashion similar to the likelihood:

$$f(\mathcal{X}|\mathcal{G}) = \prod_{i=1\ldots k} \frac{f(\mathcal{X}_{C_i}|\mathcal{G})}{f(\mathcal{X}_{S_i}|\mathcal{G})} \tag{7.2}$$

where $\mathcal{S} = (S_1, \ldots, S_k)$ are the separators corresponding to $\mathcal{C}$. Moreover the factors $f(\mathcal{X}_{C_i}|\mathcal{G})$ are constant for all $\mathcal{G}$ in which C_i is (or is contained in) a clique, so in that sense the conditioning on $\mathcal{G}$ is unnecessary.

Let now Γ be the set of forests with vertex set V. These models are decomposable, and so using (7.2) we obtain for $\mathcal{G} \in \Gamma$

$$f(\mathcal{X}|\mathcal{G}) = \frac{\prod_{e\in E(\mathcal{G})} f(\mathcal{X}_e)}{\prod_{v\in V} f(\mathcal{X}_v)^{d_{\mathcal{G}}(v)-1}} \tag{7.3}$$

where $d_{\mathcal{G}}(v)$ is the degree of v in $\mathcal{G}$. Let $\mathrm{BF}(e)$ be the Bayes factor for independence along edge $e = (u, v)$, so that

$$\mathrm{BF}(e) = \frac{f(\mathcal{X}_e)}{f(\mathcal{X}_u) f(\mathcal{X}_v)}.$$

Then (7.3) can be written as

$$f(\mathcal{X}|\mathcal{G}) = \prod_{v\in V} f(\mathcal{X}_v) \prod_{e\in E(\mathcal{G})} \mathrm{BF}(e) \tag{7.4}$$

It follows from (7.1) and (7.4) that assuming a uniform prior on Γ we can find the MAP estimate by using a maximum weight spanning tree algorithm, using logarithms to the $\mathrm{BF}(e)$ as edge weights.

From (41) in Dawid and Lauritzen (1993) we can derive an expression for $\mathrm{BF}(e)$ in terms of ratios of gamma functions:

$$\mathrm{BF}(e) = \frac{\Gamma(\lambda_{\cdot\cdot} + n_{\cdot\cdot})/\Gamma(\lambda_{\cdot\cdot}) \prod_{ij} \Gamma(\lambda_{ij} + n_{ij})/\Gamma(\lambda_{ij})}{\prod_i \Gamma(\lambda_{i\cdot} + n_{i\cdot})/\Gamma(\lambda_{i\cdot}) \prod_j \Gamma(\lambda_{\cdot j} + n_{\cdot j})/\Gamma(\lambda_{\cdot j})}$$

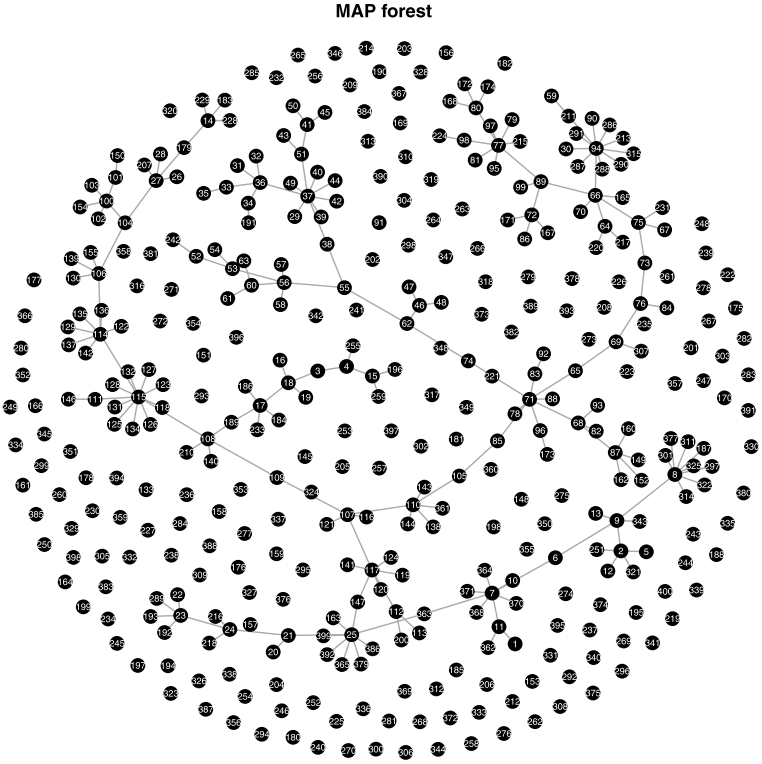
where i and j range over the number of levels of X_u and X_v, $\{n_{ij}\}$ is the corresponding table of counts, $\{\lambda_{ij}\}$ the corresponding array of smoothing parameters, and the $\cdot$ notation indicates marginal totals.

The following example illustrates application of this approach to find the MAP forest for a dataset with 400 discrete variables (SNPs). A convenient choice is $\lambda(i) = \alpha/|\mathcal{I}|\ \forall i$, where α is a scalar. This implies that $\lambda_{u,v} = \alpha/(|X_u||X_v|)$ where $|X_u|$ and $|X_v|$ are the number of levels of X_u and X_v. In the following fragment we extract the dataset from the **GGtools** package, define a function to calculate the logarithms of the Bayes factors, and call the `minForest()` function specifying that `logBF` be used to calculate the edge weights. For comparison purposes we also find the minimum BIC forest:

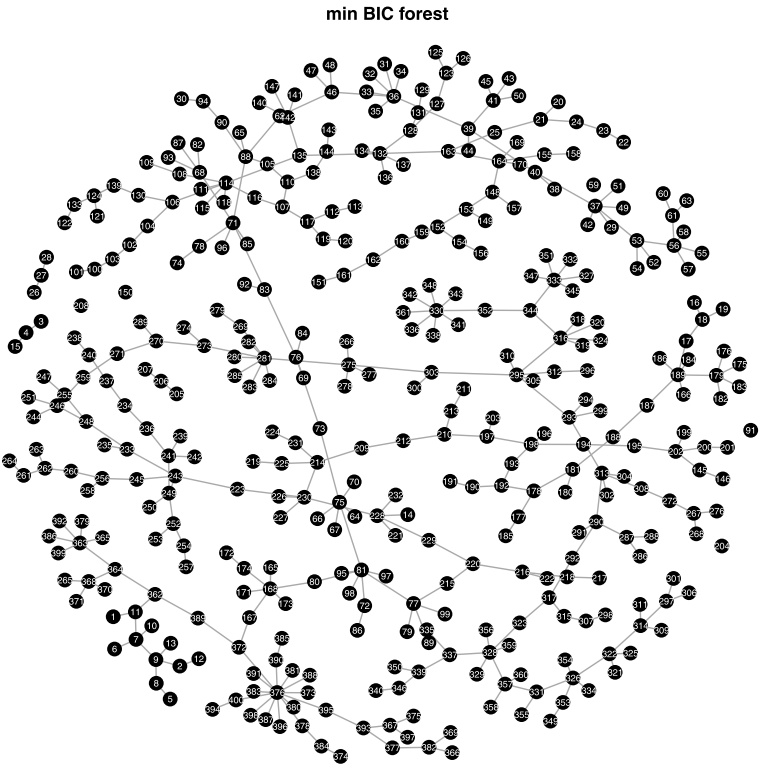
```
> data(hmceuB36.2021)
> p <- 400
> SNPdata <- data.frame(as(smList(MAFfilter(hmceuB36.2021, lower=.1))
+      [["21"]][,1:p],"character"))
> SNPdata[,1:p] <- lapply(SNPdata[,1:p], factor)
> logBF <- function(newEdge, numCat, dataset, alpha=1) {
+   i <- newEdge[1]; j <- newEdge[2]
+   n <- table(dataset[,i], dataset[,j])
+   I <- dim(n)[1];  J <- dim(n)[2]; IJ<-I*J
+   nm <- addmargins(n)
+   ni <- nm[1:I,J+1]; nj <- nm[I+1,1:J]; N <- nm[I+1,J+1]
+   fij <- sum(lgamma(n+alpha/IJ)-lgamma(alpha/IJ))
+   fi <- sum(lgamma(ni+alpha/I)-lgamma(alpha/I))
+   fj <- sum(lgamma(nj+alpha/J)-lgamma(alpha/J))
+   f <- lgamma(N+alpha) - lgamma(alpha)
+   logBF <- fij - fi - fj + f
+   return(logBF)
+ }
> snp.MAP <- minForest(SNPdata, stat=logBF, alpha=1)
> snp.F <- minForest(SNPdata)
```

Then we display the two graphs:

```
> plot(snp.MAP, numIt=500, vert.labels=1:snp.MAP@p, main="MAP forest")
```



```
> plot(snp.F, numIt=500, vert.labels=1:snp.F@p, main="min BIC forest")
```



We see that the MAP estimate has many isolated vertices, indicating a stronger tendency to negative logBF values than negative BIC values for weakly associated variables.