Probabilistic Graphical Lasso

NIANG Mohamed

06 April 2020

Contents

1 Exercise 1: The elements of Statistical Learning

1.1 Choice of section to be read

For this part, we have chosen to read the section 17.3.2 entitled **Estimation of the Graph Structure**.

Given some realizations of X, we would like to estimate the parameters of an undirected graph that approximates their joint distribution. Suppose first that the graph is complete (fully connected). We assume that we have N multivariate normal realizations x_i, $i = 1, ..., N$ with population mean μ and covariance Σ.

Let

$$\mathbf{S} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

be the empirical covariance matrix, with $\bar{x}$ the sample mean vector.

1.2 Summary of chosen section: Estimation of the Graph Structure

The **L1 (lasso) regularization** is a way to find out from the data which **edges** should be omitted from the graph. Meinshausen et al (2006) take a lasso regression approach using each variable as a response and the others as predictors to estimate which components of θ_{ij} are non-zero. The θ_{ij} component is then estimated as non-zero if the estimated coefficient of variable i on j is non-zero, or inversely. This procedure makes it possible to consistently estimate all non-zero components of Θ.

It is possible to adopt a more systematic approach with the lasso penalty by maximizing the log-likelihood penalized

$$\log \det \Theta - \text{trace}(\mathbf{S}\Theta) - \lambda \|\Theta\|_1,$$

where $\|\Theta\|_1$ is the L1 norm - the sum of the absolute values of the elements of $\sum^{-1}$, and where constants are ignored. The negative of this penalized likelihood is a convex function of Θ.

The latter system is exactly equivalent to the estimating equations of a lasso regression. Indeed, let us consider the usual regression configuration with the result variables y and the predictive matrix Z. Here, the lasso minimizes

$$\frac{1}{2}(\mathbf{y}-\mathbf{Z}\beta)^T(\mathbf{y}-\mathbf{Z}\beta)+\lambda\cdot\|\beta\|_1$$

The gradient of this expression is

$$\mathbf{Z}^T\mathbf{Z}\beta-\mathbf{Z}^T\mathbf{y}+\lambda\cdot\text{Sign}(\beta)=0$$

Algorithm Graphical Lasso

1. Initialize $\mathbf{W}=\mathbf{S}+\lambda\mathbf{I}$. The diagonal of $\mathbf{W}$ remains unchanged in what follows.
2. Repeat for $j=1,2,\ldots p,1,2,\ldots p,\ldots$ until convergence:
(a) Partition the matrix $\mathbf{W}$ into part $1:$ all but the j th row and column, and part $2:$ the j th row and column.
(b) Solve the estimating equations $\mathbf{W}_{11}\beta-s_{12}+\lambda\cdot\operatorname{sign}(\beta)=0$ using the cyclical coordinate-descent algorithm for the modified lasso.
(c) Update $w_{12}=\mathbf{W}_{11}\hat{\beta}$
3. In the final cycle (for each j) solve for $\hat{\theta}_{12}=-\hat{\beta}\cdot\hat{\theta}_{22},$ with $1/\hat{\theta}_{22}=w_{22}-w_{12}^T\hat{\beta}$

Thus, up to a factor of $\frac{1}{N}$, $\mathbf{Z}^T\mathbf{y}$ is the analogue of s_{12} and we replace $\mathbf{Z}^T\mathbf{Z}$ by $\mathbf{W}_{11}$.

Assuming that $\mathbf{V}=\mathbf{W}_{11}$, the update has the form

$$\hat{\beta}_j\leftarrow S\left(s_{12j}-\sum_{k\neq j}V_{kj}\hat{\beta}_k,\lambda\right)/V_{jj}$$

for $j=1,2,\ldots,p-1,1,2,\ldots,p-1,\ldots$, where S is the soft-threshold operator:

$$S(x,t)=\operatorname{sign}(x)(|x|-t)+$$

The procedure runs through the predictors until convergence.

2 Exercise 2: Gaussian Graphical Model

2.1 Simulation of data using a Gaussian graphical model

```
library(simone)
```

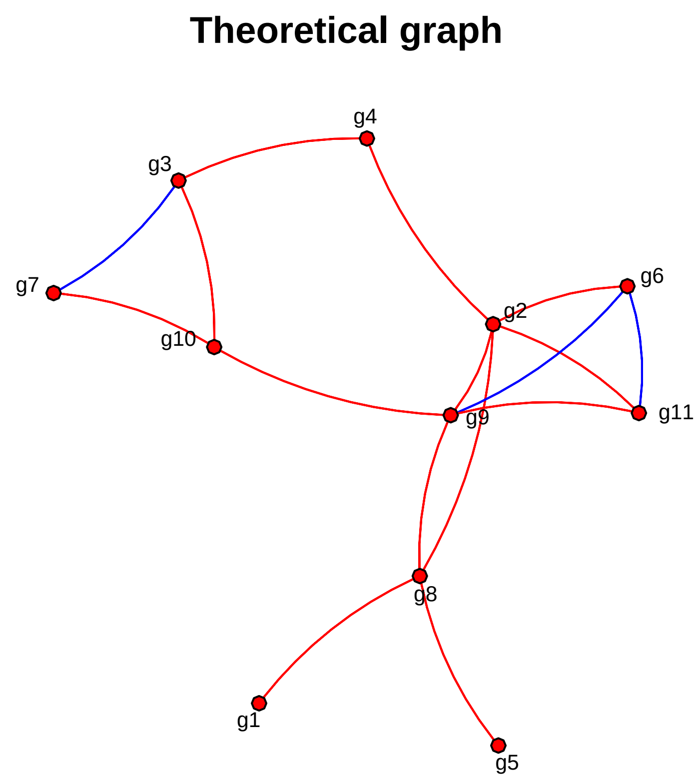
```
## Warning: package 'simone' was built under R version 3.6.3
## Loading required package: blockmodels
## Warning: package 'blockmodels' was built under R version 3.6.3
## Loading required package: Rcpp
## Loading required package: parallel
```

```
## Loading required package: digest
##
## ----------------------------------------------------------------------
##
##       'simone' package version 1.0-3
##       SIMoNe page (http://julien.cremeriefamily.info/simone.html)
##
## ----------------------------------------------------------------------
## Note that versions >= 1.0-0 are not compatible with versions < 1.0.0.
## ----------------------------------------------------------------------
```

```
set.seed(2)
```

```
edges <- 11
graphe <- rNetwork(edges, pi=0.2, name="Theoretical graph")
```

```
plot(graphe)
```




```
data <- rTranscriptData(n=1500, graphe, sigma=0)
```

```
# Display the Sample
head(data$X)
```

```
##            g1         g2          g3         g4           g5         g6
```

```
## 1 -1.6166909 -1.5839797  2.02983317  0.2202606 -1.199402085 -0.7698566
## 2  2.1888756  1.2887958  2.37650366  0.5684360 -0.005703941 -1.0537361
## 3  1.3156013  1.4330767  0.26238022 -0.3697946  1.580161776  0.2277201
## 4  0.6827336 -1.8793593 -2.12553963 -0.3667555  0.271718177  0.7647731
## 5  2.2551746 -1.4116024 -0.77945921 -0.1480044 -0.706231447 -0.0639244
## 6 -1.0659654  0.9937526 -0.09718451  0.9283668 -0.936451539  1.5101216
##           g7         g8         g9        g10        g11
## 1  0.6252290 -0.9573505 -0.5114361 -0.1660802 -1.0974591
## 2 -0.7654529  1.5681021  2.4010002  1.4274294  1.4725906
## 3  1.7625641  0.8570342 -0.5297434  1.0038496 -0.5421239
## 4 -1.6575949  1.0285248  0.5938429  0.1433947  0.3328120
## 5  0.9458830 -0.4791508  0.3052772  0.6904118 -1.0165759
## 6 -1.4022189  0.2558606  0.2457645 -0.1109221  0.5430077
```

2.2 Inference of a conditional independence graph

```r
simone(data$X,control = setOptions(penalties=0.1)) -> infered_net
```

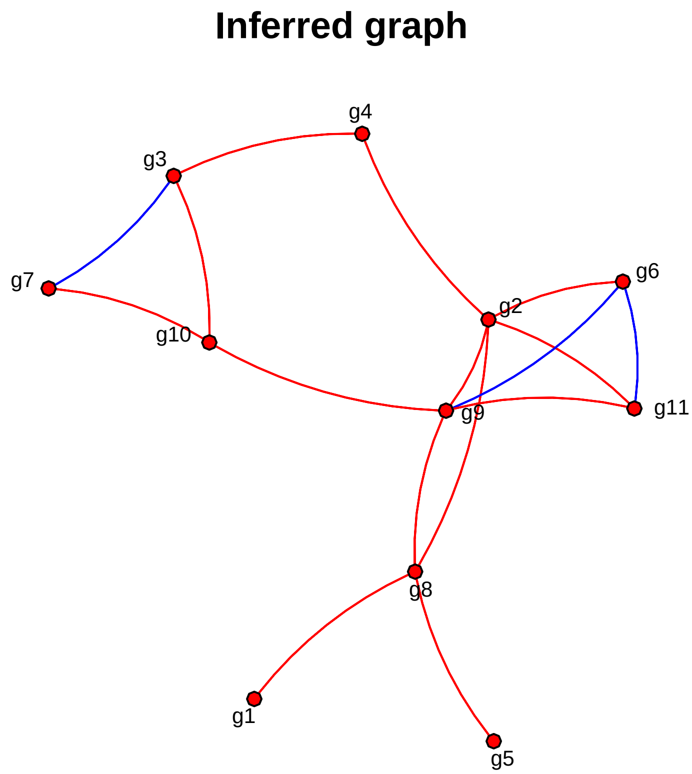
```
##
## Network Inference: neighborhood.selection with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##        0.1         16      -8048
```

```r
infered_net_graph <- getNetwork(infered_net)
```

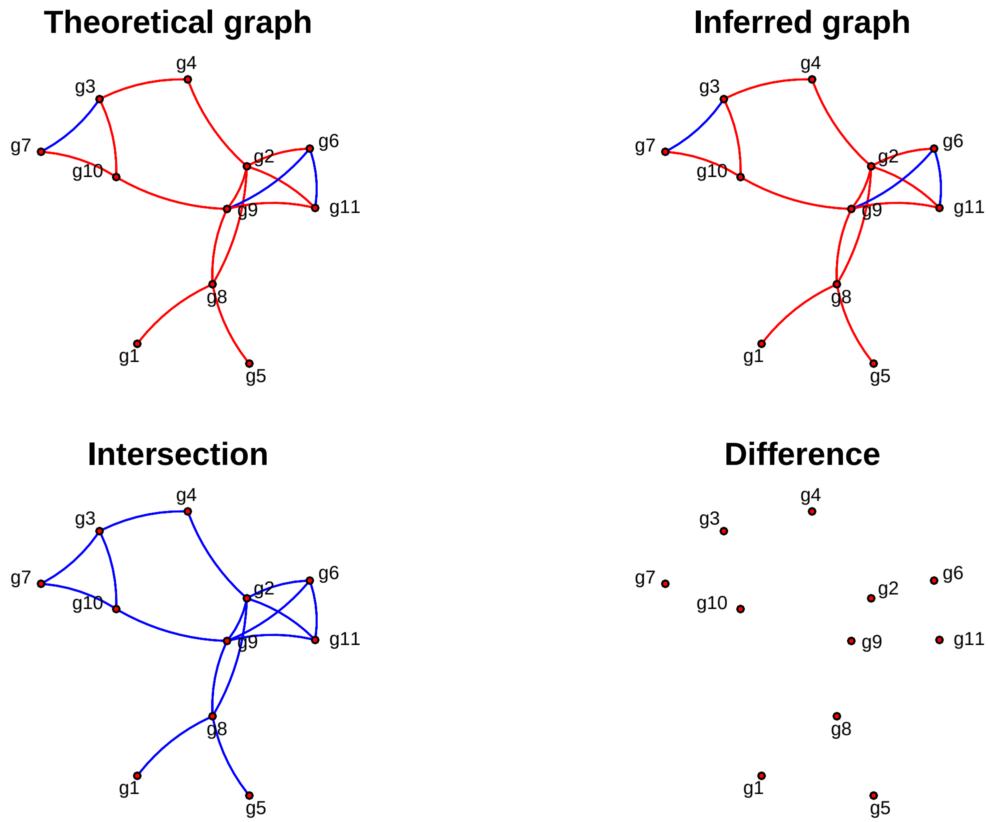
```
##
## Found a network with 16 edges.
```

```r
infered_net_graph$name <- "Inferred graph"
```

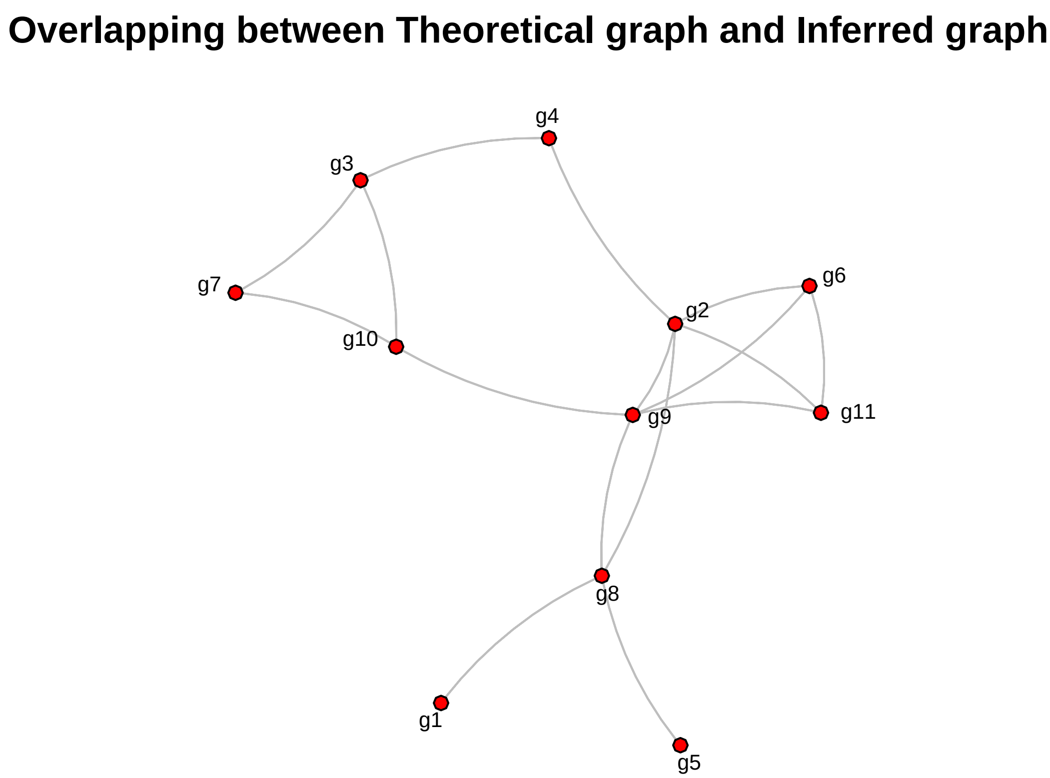
```r
plot(infered_net_graph)
```



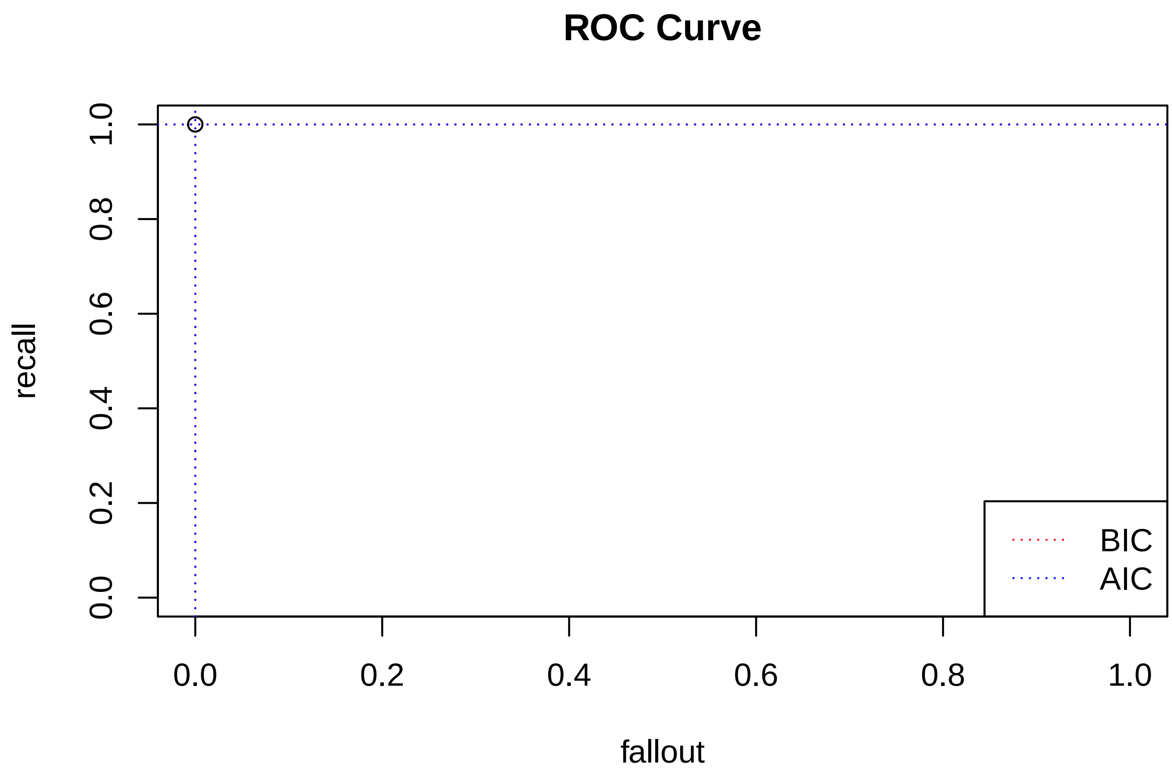

```
## Let us compare the two networks
plot(graphe,infered_net_graph)
```




```
plot(graphe,infered_net_graph, type="overlap")
```




```
plot(infered_net, output=c("ROC"), ref.graph=graphe$A)
```




```
##
## Press return for next plot...
```

Comments:

In terms of common edges, the inferred graph and the theoretical graph have no difference.

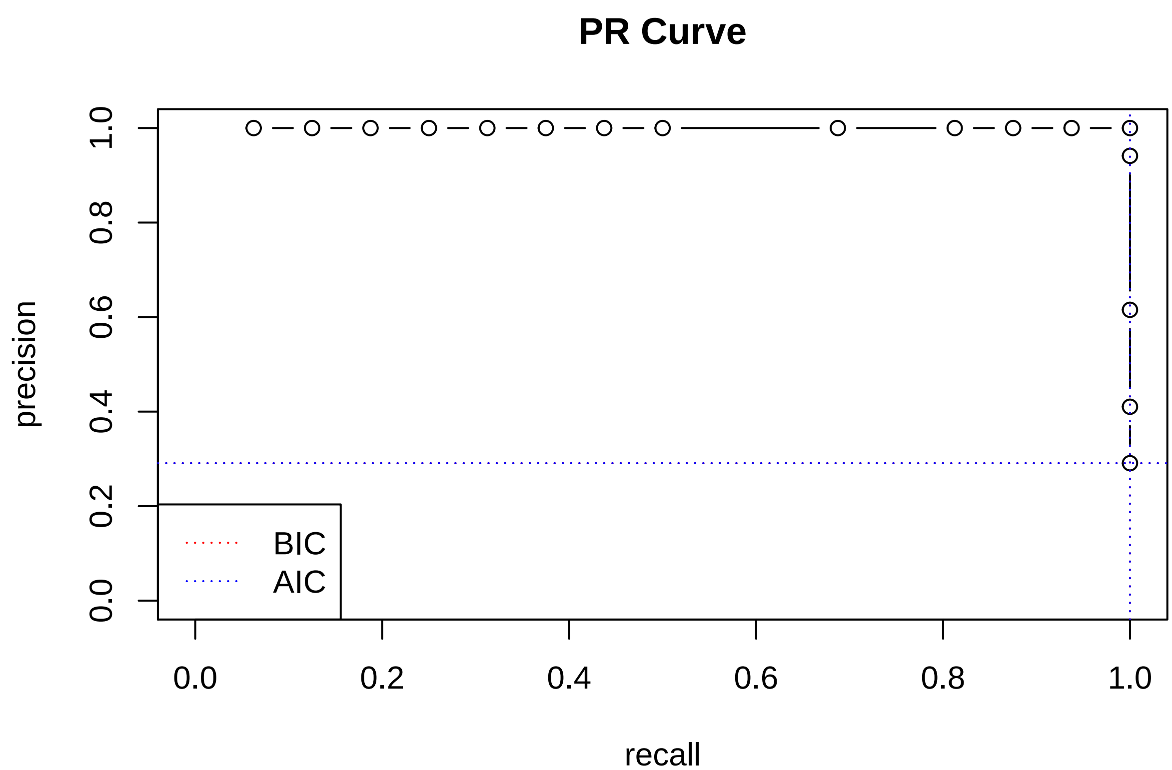
2.3 Precision-recall curve for 100 different penalties

```
simone(data$X,control=setOptions(n.penalties = 100)) -> infered_net_100
```

```
##
## Network Inference: neighborhood.selection with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##          1          0     -23698
##     0.3737          1     -18528
##     0.3434          2     -18030
##     0.3333          3     -17824
##     0.3232          4     -17545
##      0.303          5     -16931
##     0.2727          6     -15899
##     0.2525          7     -15173
##     0.2424          8     -14731
##     0.2323         11     -14259
##     0.2121         13     -13248
##      0.202         14     -12737
```

```
##      0.1818         15      -11735
##      0.1111         16       -8551
##     0.04041         17       -5551
##     0.02021         26       -4721
##     0.01011         39       -4294
##       1e-05         55       -3752
```

```
plot(infered_net_100, output=c("PR"), ref.graph=graphe$A)
```




```
##
## Press return for next plot...
```

2.4 Inference and display of conditional independence networks

```
PATH <- getwd()
setwd(dir = PATH)
# Load data
dataset <- read.table(file.choose(), header = FALSE, sep= "", encoding="UTF-8")
# Display the Sample
head(dataset)
```

```
##           V1         V2        V3        V4         V5         V6
## 1  -97.67193 -132.18100 -46.03364 -132.8207  31.765040 -20.021190
## 2  -88.17193 -128.88100 -42.55364 -134.3207 -18.904960  -8.031193
## 3  -64.67193 -101.28100 -40.25364 -140.9207 -14.034960 -11.731190
## 4  -51.07193  -62.58096 -31.75364 -137.6207 -25.744960 -20.801190
```

```
## 5  -90.37193 -125.58100 -49.66364 -141.3907  -2.234962  -5.531193
## 6 -105.27190 -141.63100 -37.25364 -129.0207 -16.134960 -14.731190
##          V7         V8        V9       V10        V11
## 1 -64.16721 -211.75860 -13.34166  -90.1145 -33.267500
## 2 -48.66721 -273.75860 -26.97166 -118.5145 -11.767500
## 3 -48.66721 -222.75860 -18.94166 -103.1145 -53.767500
## 4 -69.36721  -97.75859 -16.64166 -106.4145 -50.167500
## 5 -35.06721 -320.75860 -25.68166 -109.3145   8.032497
## 6 -55.46721  -15.75859 -16.64166  -85.9145 -15.467500
```

```
simone(dataset,control=setOptions(penalties=0.2)) -> infered_net_0.2
```

```
##
## Network Inference: neighborhood.selection with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##        0.2          7     -30732
```

```
simone(dataset,control=setOptions(penalties=0.1)) -> infered_net_0.1
```

```
##
## Network Inference: neighborhood.selection with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##        0.1          9     -17115
```

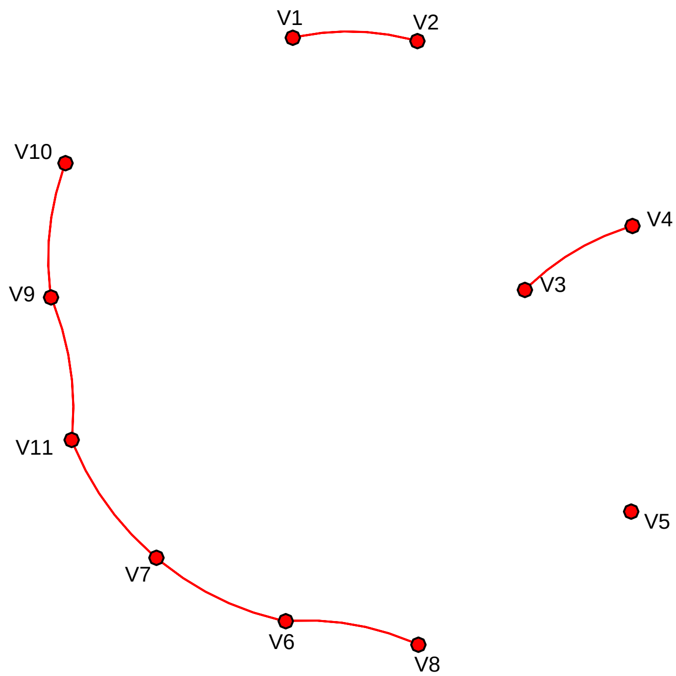
```
infered_net_0.2 <- getNetwork(infered_net_0.2)
```

```
##
## Found a network with 7 edges.
```

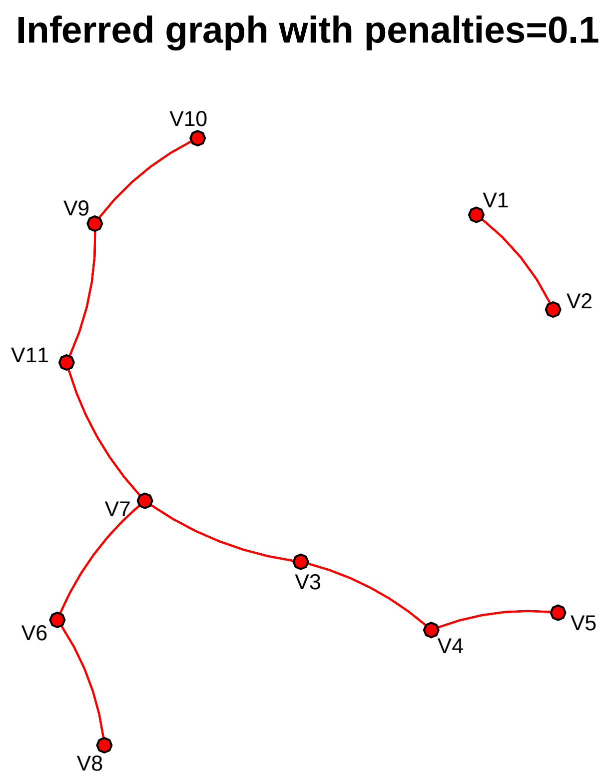
```
infered_net_0.2$name <- "Inferred graph with penalties=0.2"
```

```
plot(infered_net_0.2)
```

Inferred graph with penalties=0.2

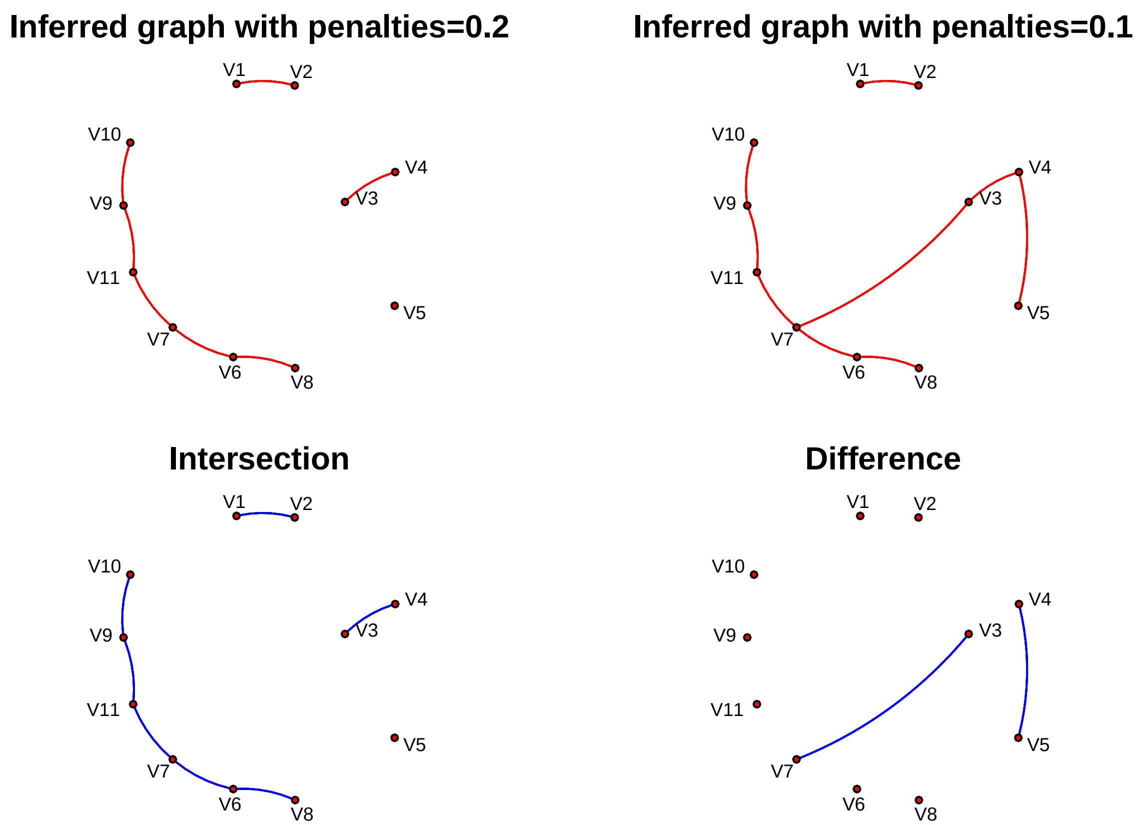



```
infered_net_0.1 <- getNetwork(infered_net_0.1)

##
## Found a network with 9 edges.

infered_net_0.1$name <- "Inferred graph with penalties=0.1"
plot(infered_net_0.1)
```



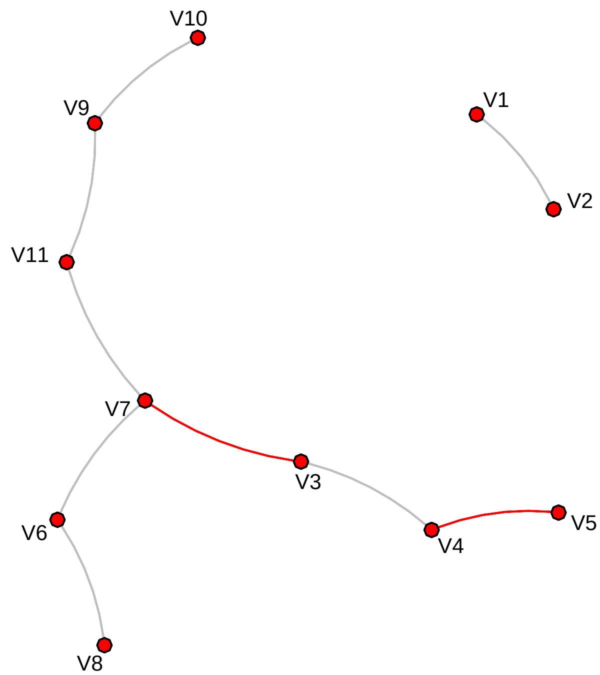
2.5 Discussion of results

```
## Let us compare the two networks
plot(infered_net_0.2,infered_net_0.1)
```




```
plot(infered_net_0.2,infered_net_0.1, type="overlap")
```

g between Inferred graph with penalties=0.2 and Inferred graph with pe



The inferred network with a penalty of 0.2 has seven edges while the inferred network with a penalty of 0.1 has nine edges.