Lasso graphique

Fernanda TcHOUACHEU

24/10/2018

Exercice 1

1. Lisez le chapitre 17 du livre “The elements of Statistical Learning” : https://web.stanford.edu/~hastie/Papers/ESLII.pdf

2. Faire un résumé sous forme de 5 à 10 transparents (fichier rmd) en anglais de la section 17.3.1 ou de la section 17.3.2 au choix.

17.3.1 Estimation of the Parameters when the Graph Structure is Known

First we will set the basics

my plan is as follows:

1.what is an undirected graphical model ?

2.Global Markov Properties on Undirected Graphs

3.Examples of undirected graph models for continuous observations

4.Estimation for the full graph

5.Estimation for a sparse graph

1. what is an undirected graphical model ?

- An undirected graphical model represents a distribution $P(X_1, \dots, X_d)$ defined by an undirected graph H, and a set of positive potential functions ψc associated with the cliques of H, s.t.

$$p(x_1, \dots, x_d) = \frac{1}{Z} \prod_{c \in C} \psi\, c(x_c)$$

where C is the set of maximal cliques, and the positive functions ψC (·) are called clique potentials. These are not in general density functions, but rather are affinities that capture the dependence in X_C by scoring certain instances x_C higher than others.

where Z is known as the partition function. It is a normalization constant.

$$Z = \sum_{X_1,\dots,X_d} \prod_{c \in C} \psi\, c(X_c)$$

- Also known as Markov Random Fields, Markov network.
- The potential function can be understood as a contingency function of its arguments assigning “pre-probabilistic” score of their joint configuration.

Hammersley-Clifford Theorem

Let P be a positive distribution over V, and H a Markov network graph over V. If H is an Independence map for P, then P is a Gibbs distribution over H.

2.Global Markov Properties on Undirected Graphs

- Let H be an undirected graph
- C separates A and B if every path from a node in A into a node in B passes through a node in B :

$sep_H(A; B|C)$.

- A probability distribution satisfies the global Markov property if for any disjoint A, B,C, such that C separates A and B, A is independent of B given C :

$I(H) = A \perp B|C: sep_H(A; B|C)$

It turns out that the pairwise and global Markov properties of a graph are equivalent (for graphs with positive distributions).

3.Examples of undirected graph models for continuous observations

- Here we consider Markov networks where all the variables are continuous
- The Gaussian distribution is almost always used for such graphical models, because of its convenient analytical properties.

$$X \sim \mathcal{N}(\mu, \Sigma)$$

- The Gaussian distribution represents at most second-order relationships, it automatically encodes a pairwise Markov graph.
- The Gaussian distribution has the property that all conditional distributions are also Gaussian
- One of the nice properties of the multivariate Gaussian distribution is that all its conditional distributions are Gaussian too.

Let Y and z be such that $Z = (X_1, \cdots, X_{p-1})$ and $Y = X_p$.Then we have the conditional distribution of Y give Z

$$Y|Z = z \sim \mathcal{N}(\mu_y + (z - \mu_Z)^{\mathcal{T}} \Sigma_{ZZ}^{-1} \sigma_{Zy}, \sigma_{yy} - \sigma_{Zy}^{\mathcal{T}} \Sigma_{ZZ}^{-1} \sigma_{Zy}$$

Where we have partitioned Σ as

$$\Sigma = \begin{pmatrix} \Sigma_{ZZ} & \sigma_{ZY} \\ \sigma_{ZY}^T & \sigma_{YY} \end{pmatrix}$$

Remark:

$$\mu_Y + (z - \mu_Z)^T \Sigma_{ZZ}^{-1} \sigma_{ZY}$$

is exactly of the same form as the linear regression of Y over Z with

$$\beta = \Sigma_{ZZ}^{-1} \sigma_{ZY}$$

- with the same partition for $\Theta = \Sigma^{-1}$ and σ,since $\Theta\Sigma = I$

standard formulas for partitioned inverses give

$$\theta_{ZY} = -\theta_{YY} \Sigma_{ZZ}^{-1} \sigma_{ZY}$$

- with $\frac{1}{\theta_{YY}} = \sigma_{YY} - \sigma_{ZY}^T \Sigma_{ZZ}^{-1} \sigma_{ZY} > 0$.So that,

$\beta = \Sigma_{ZZ}^{-1} \sigma_{ZY} = \dfrac{-\theta_{ZY}}{\theta_{YY}}$

- The conditional dependencies between Y and Z are described by the mean.The Zero coefficients of θ_{ZY} leads to zero coefficients in β (and reversely)
- Finally,Θ captures all the second order structure that is needed to describe the conditional distribution.

now that we have fixed the basics we are going to talk about estimating the parameters when the structure of the graph is known in the continuous case.

4.Estimation for the full graph

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 $x_i, i = 1, \dots, N$.

- Assumption : the graph is full, μ and Σ^{-1} have no zero coefficients.
- The empirical covariance matrix is given by

$$S_n = \frac{1}{N} \sum_{i=1}^{N} (x_i - \overline{x})(x_i - \overline{x})^T$$

where $\overline{x}$ is the empirical mean.

- The log-likelihood is given by

$$L(\Theta) = logdet(\Theta) - trace(S\Theta)$$

up to an additive constant

- And

$$\frac{\partial}{\partial \theta} logdet(\Theta) = \Theta^{-1}; \frac{\partial}{\partial \theta} trace(S\Theta) = S$$

- So that the estimator of the maximum likelihood is $\Sigma = \Theta^{-1} = S$

5.Estimation for a sparse graph

The idea is based on linear regression, as inspired by (17.6) and (17.9)(ebook). In particular, suppose that we want to estimate the edge parameters θ_{ij} for the vertices that are joined to a given vertex i, restricting those that are not joined to be zero. Then it would seem that the linear regression of the node i values on the other relevant vertices might provide a reasonable estimate. But this ignores the dependence structure among the predictors in this regression. It turns out that if instead we use our current (model-based) estimate of the cross-product matrix of the predictors when we perform our regressions, this gives the correct solutions and solves the constrained maximum-likelihood problem exactly.

- Generally, the graphs are sparse
- If the graph structure is known and denoted E, the maximization of the likelihood leads to a quadratic optimization problem under equality constraints.

$$logdet(\Theta) - trace(S\Theta) - \sum_{(j,k)\notin E} \gamma_{jk}\, \theta_{jk}$$

- The gradient equation for maximizing can be written as

$$\Theta^{-1} - S - \Gamma = 0$$

using the fact that the derivative of $logdet\Theta$ equals Θ^{-1} , Γ is a matrix of Lagrange parameters with non-zero values for all pairs with edges absent.

We will show how we can use regression to solve for Θ and its inverse $W = \Theta^{-1}$ one row and column at a time.

Ideas of the modified regression algorithm

- We partition the matrices Θ and Σ into two parts:
  - Part 1 : the first p − 1 rows and columns
  - Part 2 : the last row and columns

$$\begin{pmatrix} W_{11} & W_{12} \\ W_{12}^T & W_{22} \end{pmatrix} \begin{pmatrix} \Theta_{11} & \Theta_{12} \\ \Theta_{12}^T & \Theta_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0^T & 1 \end{pmatrix}$$

- This implies

$$W_{12} = -W_{11}\theta_{12}/\theta_{22} = W_{11}\beta$$

- Then, with empirical estimates,

$$W_{11}\beta - s_{12} - \gamma_{12} = 0$$

- There are $p - q$ non-zero elements in γ_{12} (edges constrained to be zero), so that the previous equation is equivalent to

$$\beta^* = (W_{11}^*)^{-1}s_{12}^*$$

- Furthermore,$\frac{1}{\theta_{22}} = W_{22} - W_{12}^T\beta$ and $W_{22} = s_{22}$ since the diagonal of Γ is zero.

- This leads to a simple iterative algorithm.

A modified Regression Algorithm for Estimation of an Undirected Gaussian Graphical Model with Known Structure.

1. Initialize $W = \hat{\Sigma} = S$
2. Repeat for j=1,2,…,p,, until convergence.

(a) Partition the matrix W into part 1: all but the jth row and column, and part 2: the jth row and column.

(b) Solve $W_{11}^*\beta^* - s_{12}^* = 0$ for the unconstrained edge parameters β^* , using the reduced system of equations.Obtain $\hat{\beta}^*$ by padding $\hat{\beta}^*$ with zeros in the appropriate positions.

(c) Update $W_{12} = W_{11}\hat{\beta}$

3.In the final cycle (for each j) solve $\hat{\theta}_{12} = -\hat{\beta}\,\hat{\theta}_{12}$, with $1/\hat{\theta}_{22} = \hat{\sigma}_{22} - W_{12}^T\hat{\beta}$

Exemple

- S empirical covariance matrix
- By applay the algorithm; ,in the modified regression for variable 1 in step (b), variable 3 is left out.The procedure quickly converged to the solutions:

Note the zeroes in $\hat{\Sigma}^{-1}$, corresponding to the missing edges (1,3) and (2,4).Note also that the corresponding elements in $\hat{\Sigma}$ are the only elements different from S. The estimation of $\hat{\Sigma}$ is an example of what is sometimes called the positive definite “completion” of S.

$$S = \begin{pmatrix} 10 & 1 & 5 & 1 \\ 1 & 10 & 2 & 6 \\ 5 & 2 & 10 & 3 \\ 4 & 6 & 3 & 10 \end{pmatrix}$$

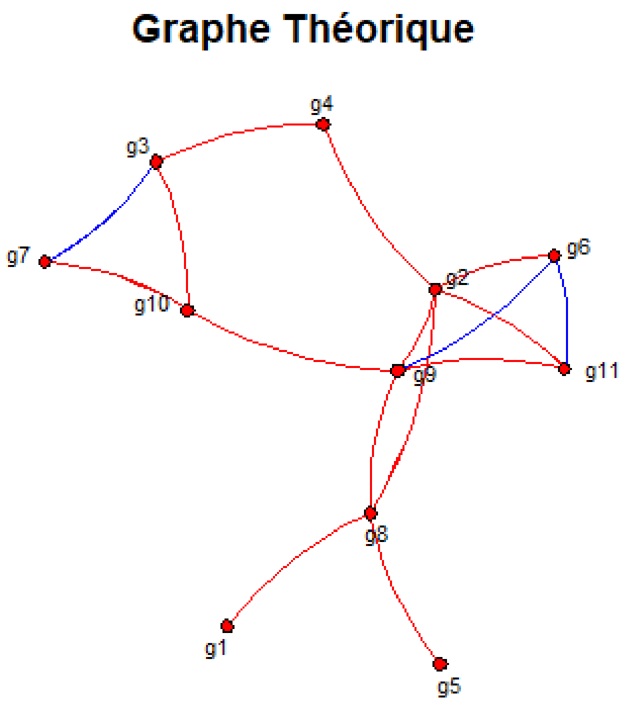
$$\hat{\Sigma} = \begin{pmatrix} 10.00 & 1.00 & 1.31 & 4 \\ 1.00 & 10.00 & 2.00 & 0.87 \\ 1.31 & 2.00 & 10.00 & 3.00 \\ 4.00 & 0.87 & 3.00 & 10.00 \end{pmatrix}$$

$$\hat{\Sigma}^{-1} = \begin{pmatrix} 0.12 & -0.01 & 0.00 & -0.05 \\ -0.01 & 0.11 & -0.02 & 0.00 \\ 0.00 & -0.02 & 0.11 & -0.03 \\ -0.05 & 0.00 & -0.03 & 0.13 \end{pmatrix}$$

Exercice 2

1. Simuler des données suivant un modèle graphique gaussien à l'aide de la fonction `rTranscriptData` du package R `simone`

```
rm(list=ls())

#install.packages('simone')
library(simone)

set.seed(2)
p <- 11
g <- rNetwork(p,pi=0.2, name="Graphe Théorique")
plot(g)
```



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

2. Utiliser un lasso graphique avec une pénalité de 0.1 (disponible dans les packages `simone` ou `glasso`) pour inférer un graphe d'indépendances conditionnelles. Comparer

votre graphe inféré et le graphe théorique en terme d'arêtes communes (Faux positifs, vrais positifs, faux négatifs, vrais négatifs). Attention il faut simplement évaluer les arêtes communes et la diagonale ne compte pas.

```
#simone(data$X,control=setOptions(penalties=0.1,edges.steady="graphical.lasso
"))
control <- setOptions(penalties= 0.1, edges.steady="graphical.lasso")
theta.simone <- simone(data$X, control=control)$networks[[1]] # SIMoNe
```

```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##        0.1         16      -8718
```

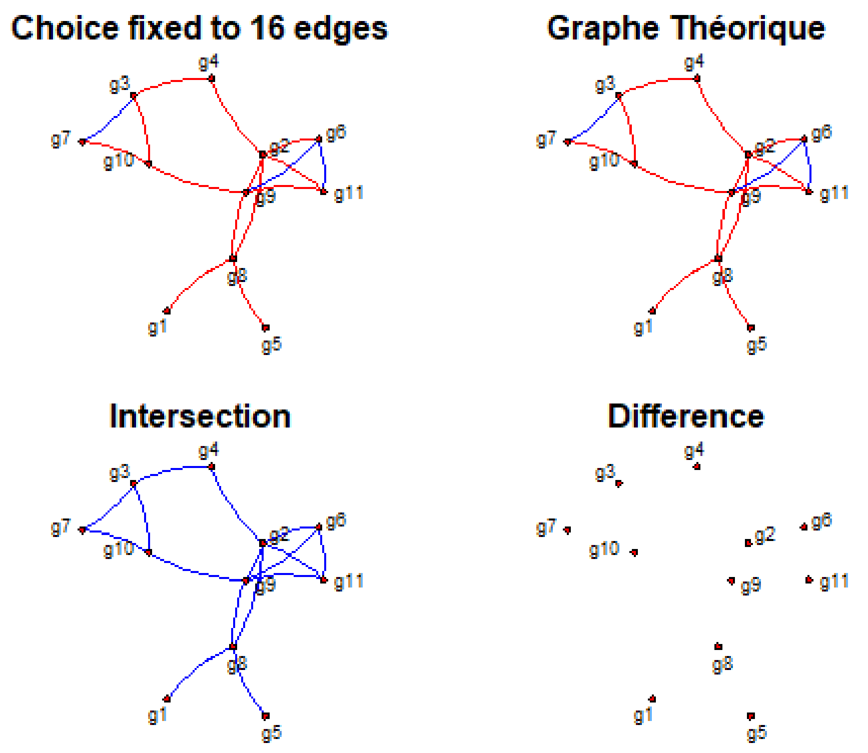
```
net.simone <- structure(list(A        = sign(theta.simone),
                             Theta    = theta.simone,
                             directed = FALSE,
                             clusters = factor(rep("N",p)),
                             name     = "inference graph",
class="simone.network"))
```

comparaison des deux graphes

```
g_inference<- getNetwork(simone(data$X, control=control))
```

```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##        0.1         16      -8718
##
## Found a network with 16 edges.
```

```
plot(g_inference,g)
```



Ici on peut observer que pour une pénalité de 0.1 on a aucune difference de liaisons entre le graphe théorique et le graphe inféré.

```
table(abs(net.simone$A))

##
##  0  1
## 78 43

table(g$A)

##
##  0  1
## 89 32

table(abs(net.simone$A),g$A)

##
##      0  1
##   0 78  0
##   1 11 32

#nombre d'incohérence de signe
sum(sign(theta.simone) != sign(g$Theta))

## [1] 0
```

```
library(matrixcalc)
A=abs(net.simone$A)
is.symmetric.matrix(A);is.symmetric.matrix(g$A)

## [1] TRUE

## [1] TRUE
```

Pour construire la matrice de confusion,nous avons pris la partie superieure des matrices adjacents des differents graphes.

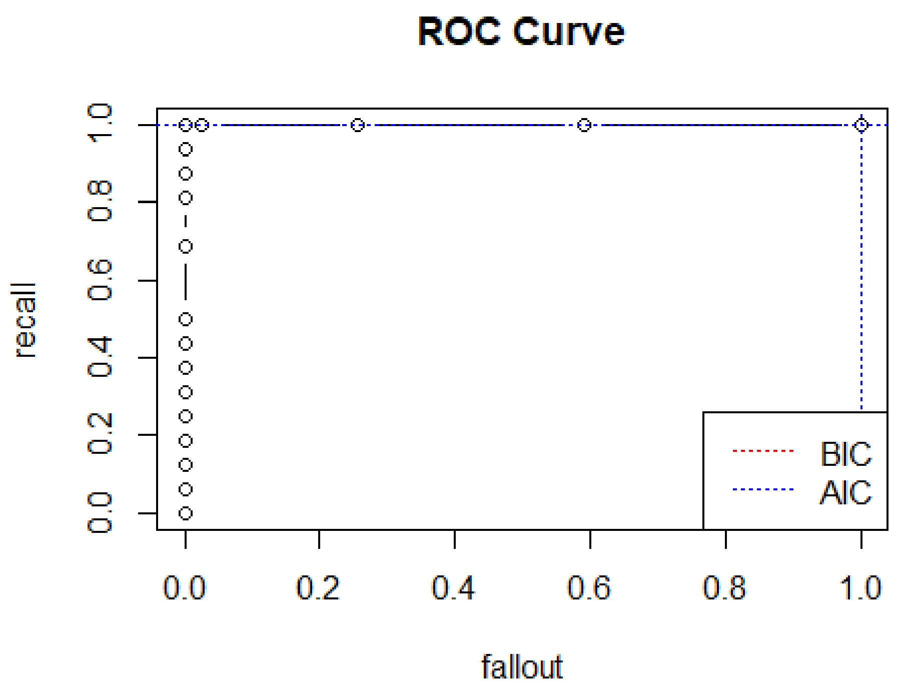
legend:

- 1<-arête(positif)
- O<- pas d'arête (négatif)

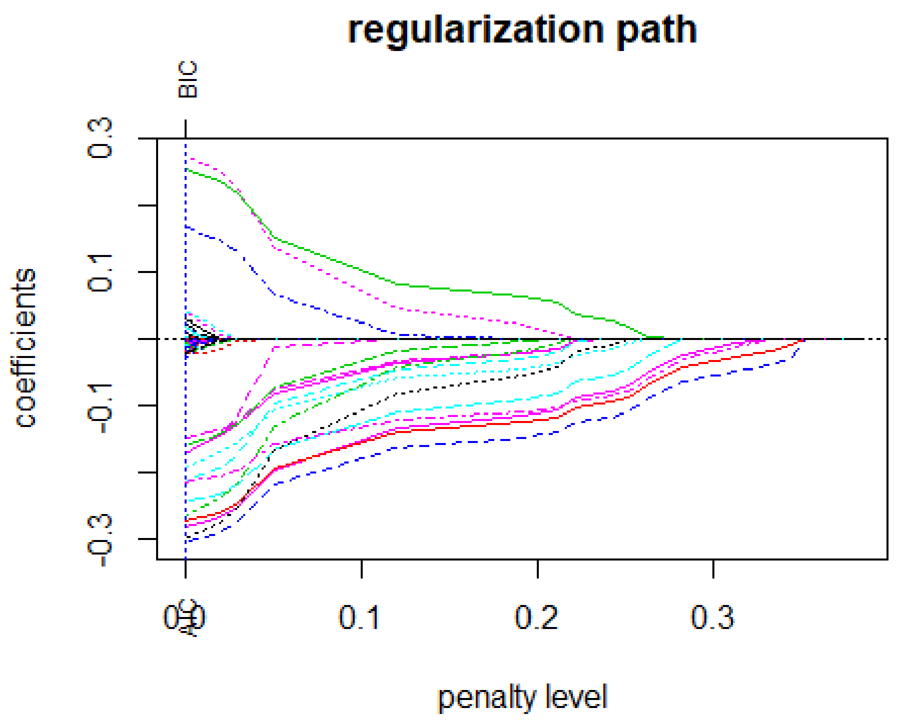
```
# matrice de confusion

m=table(graphe_infere=A[upper.tri(A,
diag=F)],graphe_theorique=g$A[upper.tri(g$A, diag=F)])
m

##               graphe_theorique
## graphe_infere  0  1
##              0 39  0
##              1  0 16
```

3. Tracer une courbe précision-rappel pour 100 pénalités différentes.

```
plot(simone(data$X, type="steady-state"), output=c("path.penalty",'ROC'),
ref.graph=g$A)

##
## 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
```



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



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

Autre simulation à la main:

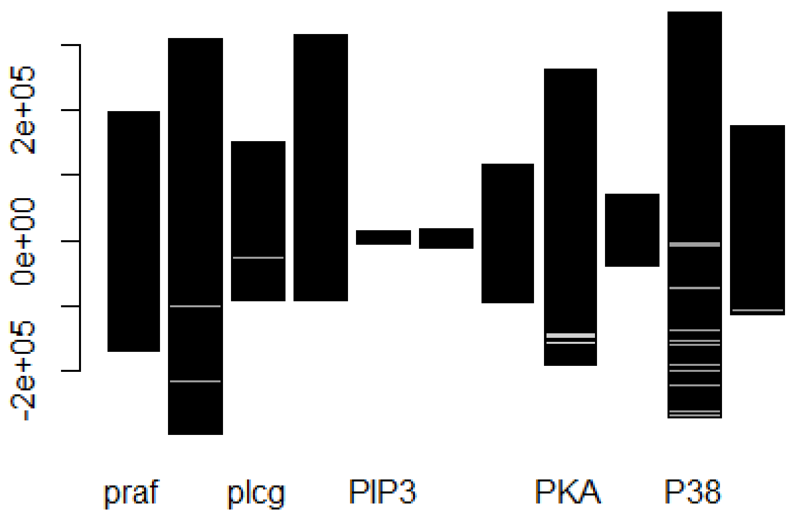
```
courbe=function(data){
net.simone=c()
theta.simone=c()
a=c()
matrix_conf=list()
prec=vector()
rappel=vector()
for (i in seq(0,1,length.out=100)){
  control <- setOptions(penalties=i, edges.steady="graphical.lasso")
  theta.simone <- simone(data, control=control)$networks[[1]]
  net.simone <- structure(list(A        = sign(theta.simone),
                             Theta     = theta.simone,
                             directed = FALSE,
                            name      = "simone package"),
class="simone.network")

 a=abs(net.simone$A)
 matrix_conf=table(graphe_infere=a[upper.tri(a,
diag=F)],graphe_theorique=g$A[upper.tri(g$A, diag=F)])
 if (length( matrix_conf)==4){
  TP= matrix_conf[2,2]
  TN= matrix_conf[1,1]
  FN= matrix_conf[1,2]
  FP= matrix_conf[2,1]
  prec1= TP/(TP+FP)
  rappel1=TP/(TP+FN)
  prec=append(prec,prec1)
  rappel=append(rappel,rappel1)
  }
  else{
  prec1=0
  rappel1=0
  }
 #plot(net.simone)
 #print(matrix_conf)
 #print(prec)
 #print(rappel)
}
return(plot(prec~rappel,type ="b",pch = 16))
}
```

4. Télécharger les données “Protein flow cytometry data” du site https://web.stanford.edu/~hastie/ElemStatLearn/data.html . Inférer et afficher les réseaux d’indépendances conditionnelles obtenus pour des pénalités différentes.

```
data1=read.table("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/sac
hs.data",sep="",header = FALSE)

names(data1)=c("praf","pmek","plcg","PIP2","PIP3","p44/42","pakts473","PKA","
PKC","P38","pjnk")
```

Description des données

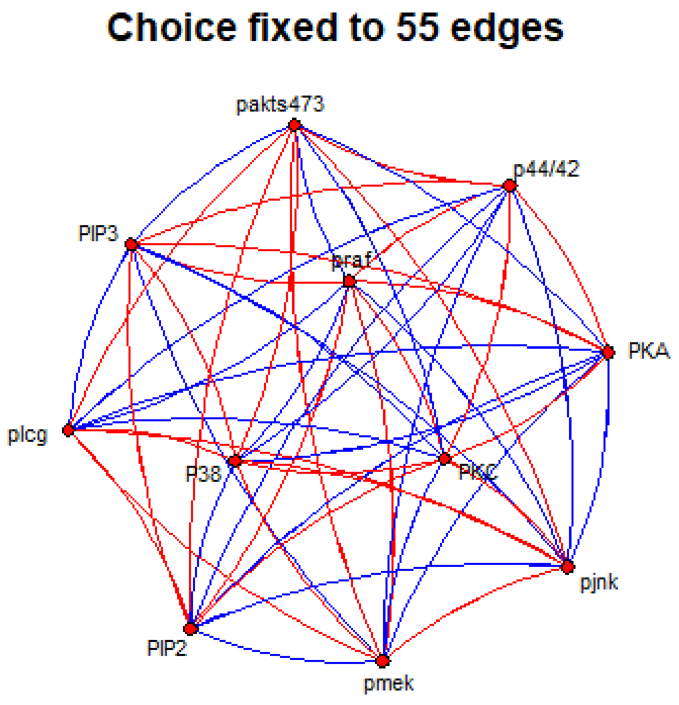
```
#View(data1)
str(data1)

## 'data.frame':    7466 obs. of  11 variables:
##  $ praf    : num  -97.7 -88.2 -64.7 -51.1 -90.4 ...
##  $ pmek    : num  -132.2 -128.9 -101.3 -62.6 -125.6 ...
##  $ plcg    : num  -46 -42.6 -40.3 -31.8 -49.7 ...
##  $ PIP2    : num  -133 -134 -141 -138 -141 ...
##  $ PIP3    : num  31.77 -18.9 -14.03 -25.74 -2.23 ...
##  $ p44/42  : num  -20.02 -8.03 -11.73 -20.8 -5.53 ...
##  $ pakts473: num  -64.2 -48.7 -48.7 -69.4 -35.1 ...
##  $ PKA     : num  -211.8 -273.8 -222.8 -97.8 -320.8 ...
##  $ PKC     : num  -13.3 -27 -18.9 -16.6 -25.7 ...
##  $ P38     : num  -90.1 -118.5 -103.1 -106.4 -109.3 ...
##  $ pjnk    : num  -33.27 -11.77 -53.77 -50.17 8.03 ...

barplot(as.matrix(data1))
```



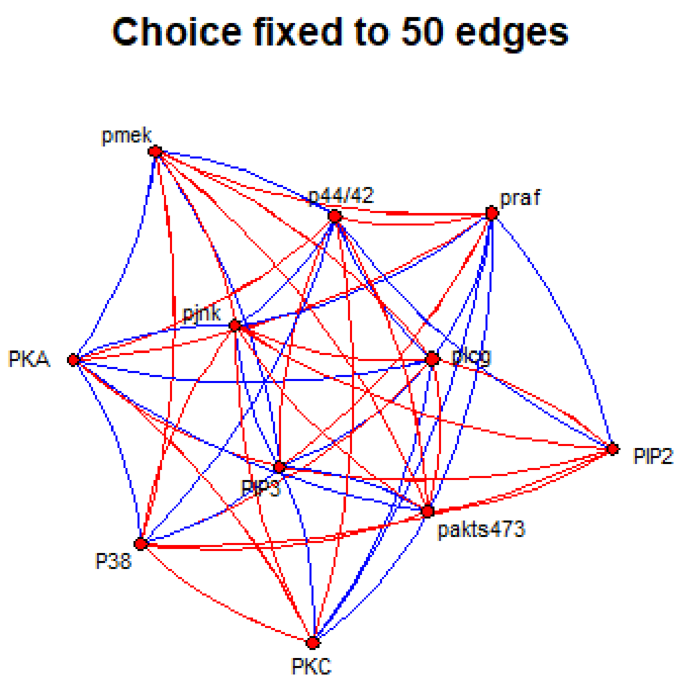
```
cor(data1)

##                 praf        pmek        plcg      PIP2         PIP3
## praf      1.00000000  0.99023837  0.24196393 0.2152472 -0.010557508
## pmek      0.99023837  1.00000000  0.27110331 0.2414428 -0.017134897
## plcg      0.24196393  0.27110331  1.00000000 0.9262334  0.086955158
## PIP2      0.21524724  0.24144281  0.92623340 1.0000000  0.195057111
## PIP3     -0.01055751 -0.01713490  0.08695516 0.1950571  1.000000000
## p44/42    0.02825065  0.03961065  0.15302681 0.1406634 -0.032356084
## pakts473  0.27380339  0.30944768  0.44297345 0.4088492 -0.049050981
## PKA      -0.14710175 -0.15999377 -0.19325159 -0.1771301  0.004951507
## PKC       0.19404499  0.21872487  0.35548599 0.3370470 -0.032547192
## P38       0.21869831  0.24455473  0.37796747 0.3566153 -0.030728772
## pjnk      0.19268871  0.22140895  0.41023973 0.3826924 -0.040549798
##               p44/42    pakts473          PKA         PKC         P38
## praf      0.02825065  0.27380339 -0.147101752  0.19404499  0.21869831
## pmek      0.03961065  0.30944768 -0.159993773  0.21872487  0.24455473
## plcg      0.15302681  0.44297345 -0.193251590  0.35548599  0.37796747
## PIP2      0.14066340  0.40884923 -0.177130090  0.33704696  0.35661534
## PIP3     -0.03235608 -0.04905098  0.004951507 -0.03254719 -0.03072877
## p44/42    1.00000000  0.68732514  0.221461335  0.13146887  0.13565132
## pakts473  0.68732514  1.00000000 -0.012466567  0.37935084  0.40111216
## PKA       0.22146134 -0.01246657  1.000000000 -0.15938167 -0.18148621
## PKC       0.13146887  0.37935084 -0.159381669  1.00000000  0.95892095
## P38       0.13565132  0.40111216 -0.181486211  0.95892095  1.00000000
## pjnk      0.15615874  0.44016188 -0.178505816  0.81399918  0.79847597
##                pjnk
## praf       0.1926887
## pmek       0.2214090
## plcg       0.4102397
## PIP2       0.3826924
## PIP3      -0.0405498
## p44/42     0.1561587
## pakts473  0.4401619
## PKA       -0.1785058
## PKC        0.8139992
## P38        0.7984760
## pjnk       1.0000000
```

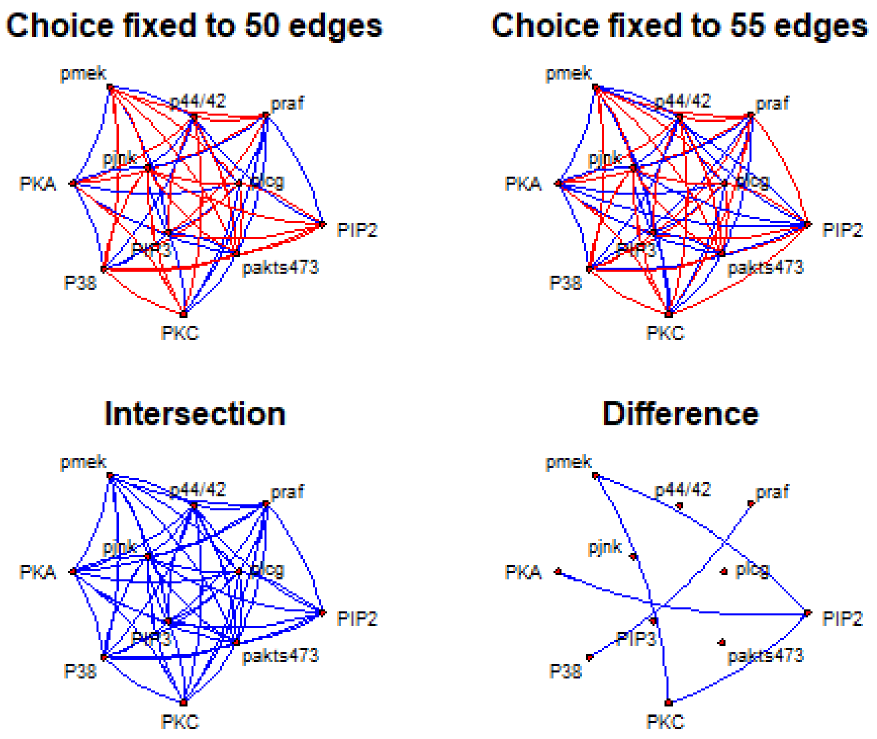
Ici on peut déjà voir une forte corrélation entre certaines protéines.

```
control=setOptions(penalties=0,edges.steady="graphical.lasso")
nr<- getNetwork(simone(data1, control=control))

##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |     edges | criteria
##
##      1e-05          55       1210
##
## Found a network with 55 edges.

plot(nr)
```



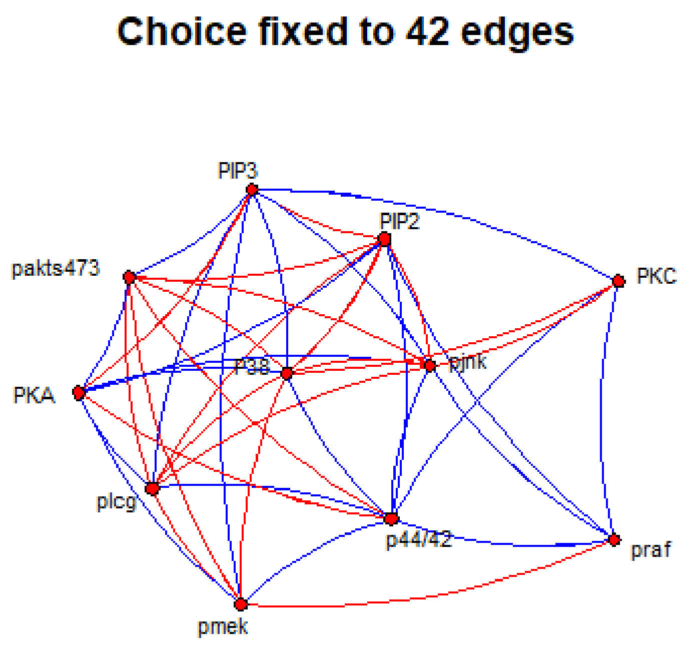
```
for (i in c(0.001,0.01,0.02,0.03,0.20,0.6,0.7,0.8,0.9)){
  m=c()

m=getNetwork(simone(data1,control=setOptions(penalties=i,edges.steady="graphi
cal.lasso")))
  plot(m)
  plot(m,nr)
}

##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##      0.001          50     -29.83
##
## Found a network with 50 edges.
```

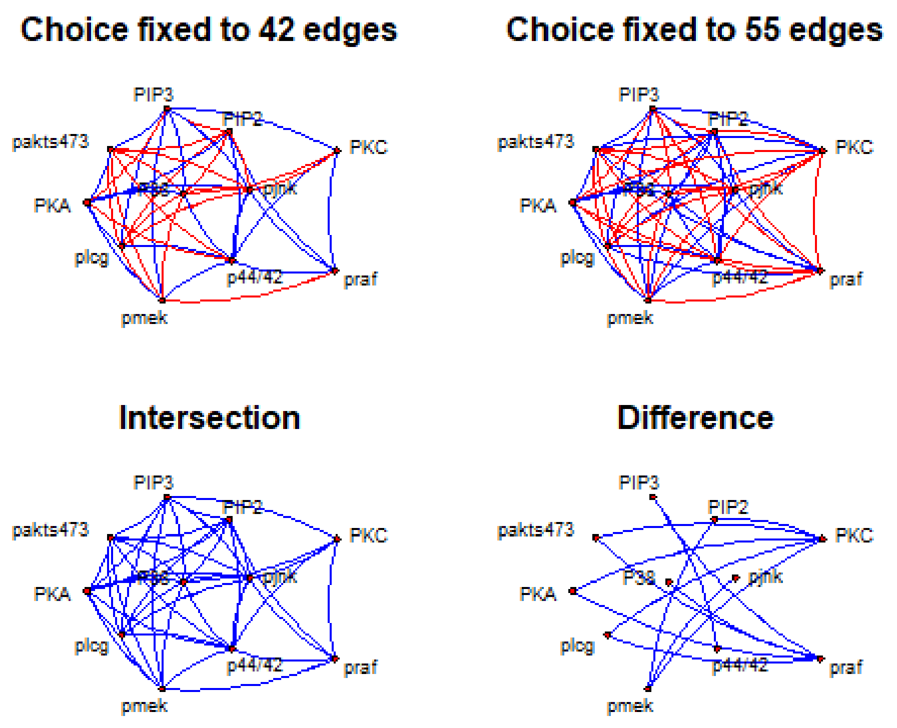




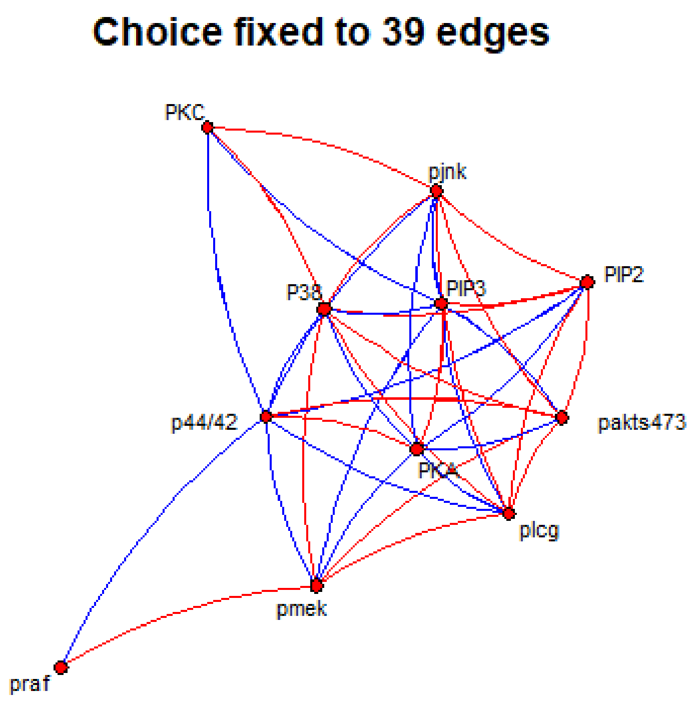
```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
```

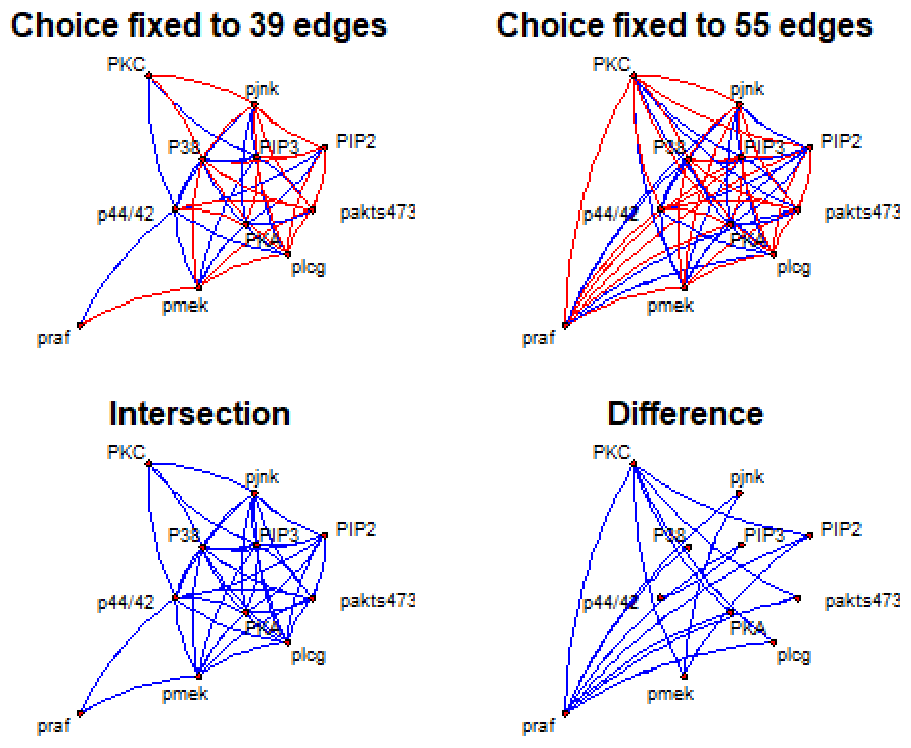
```
##
##       0.01         42      -6901
##
## Found a network with 42 edges.
```





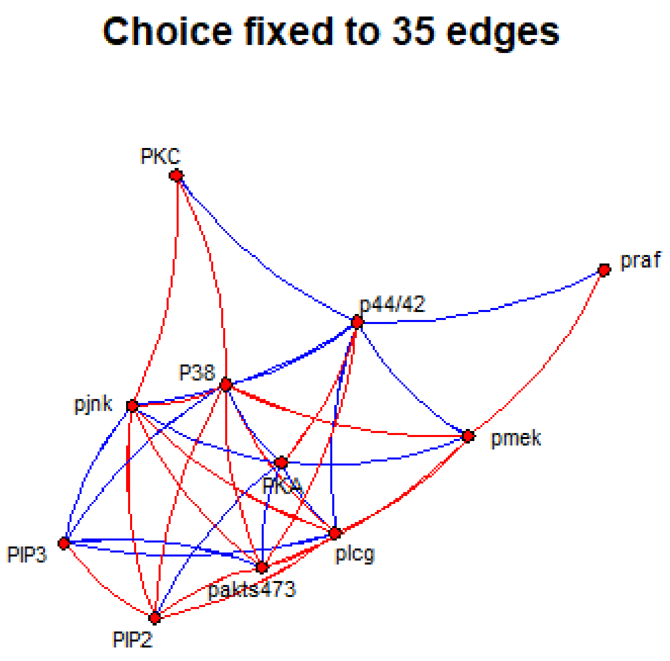
```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##       0.02         39     -11585
##
## Found a network with 39 edges.
```

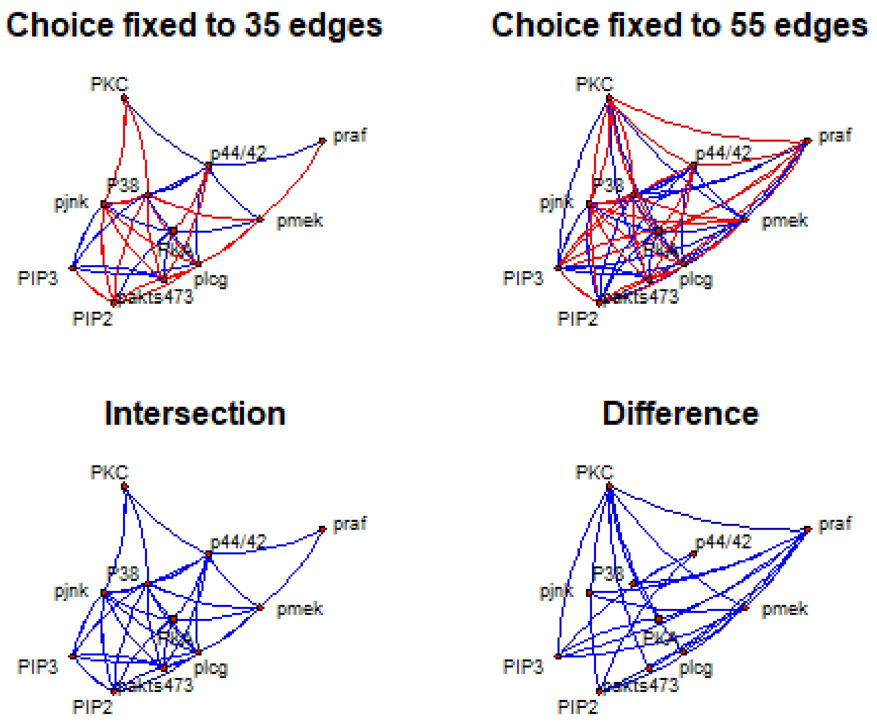




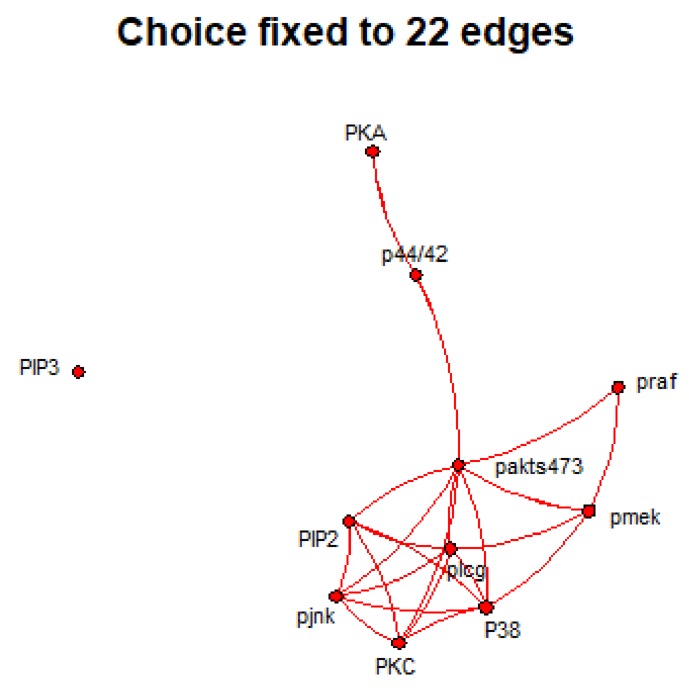
```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
```

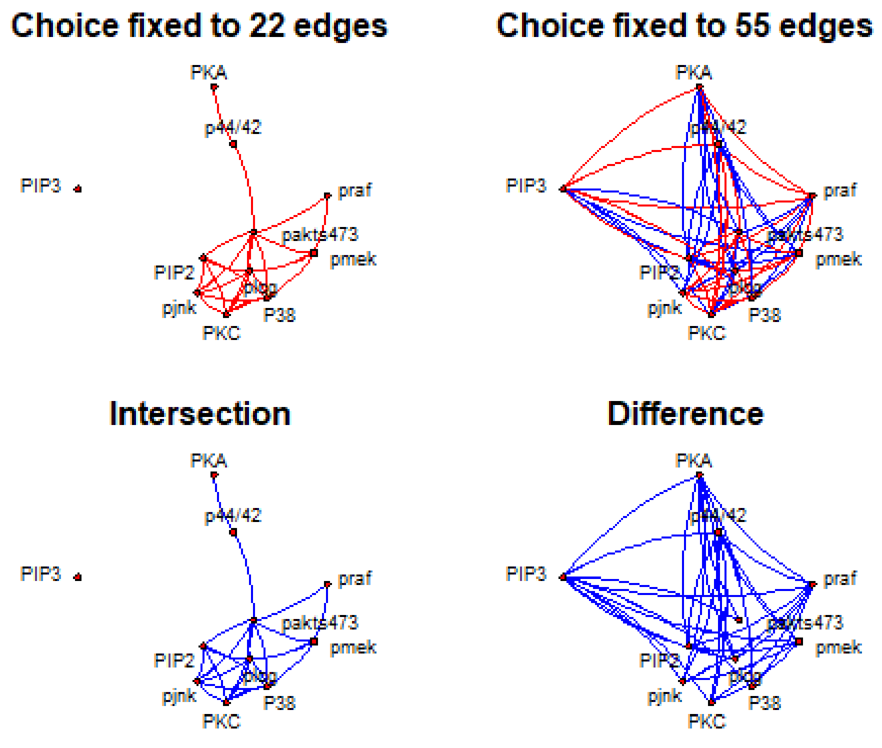
```
##
##       0.03         35      -15126
##
## Found a network with 35 edges.
```





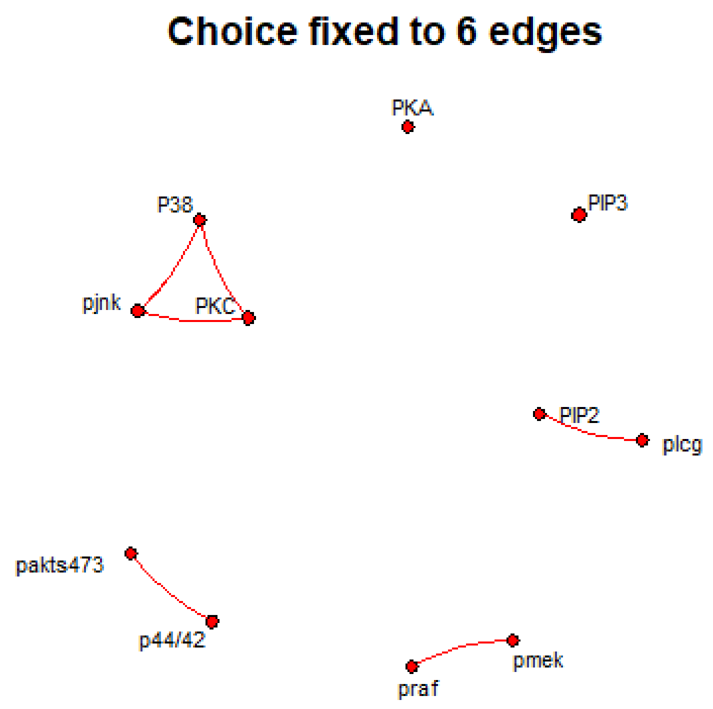
```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##        0.2         22     -40255
##
## Found a network with 22 edges.
```

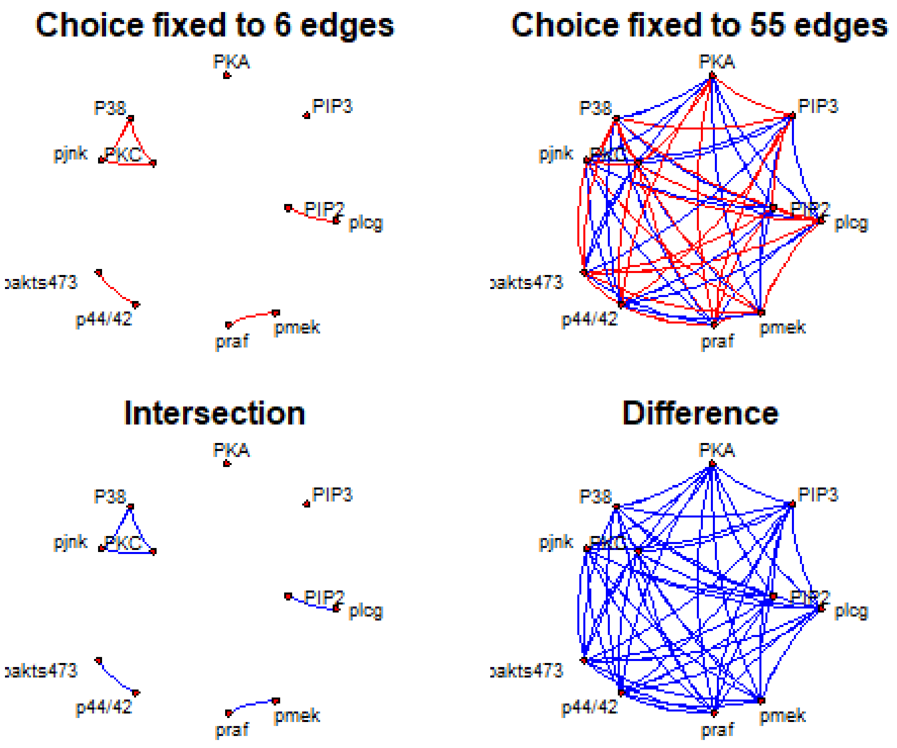




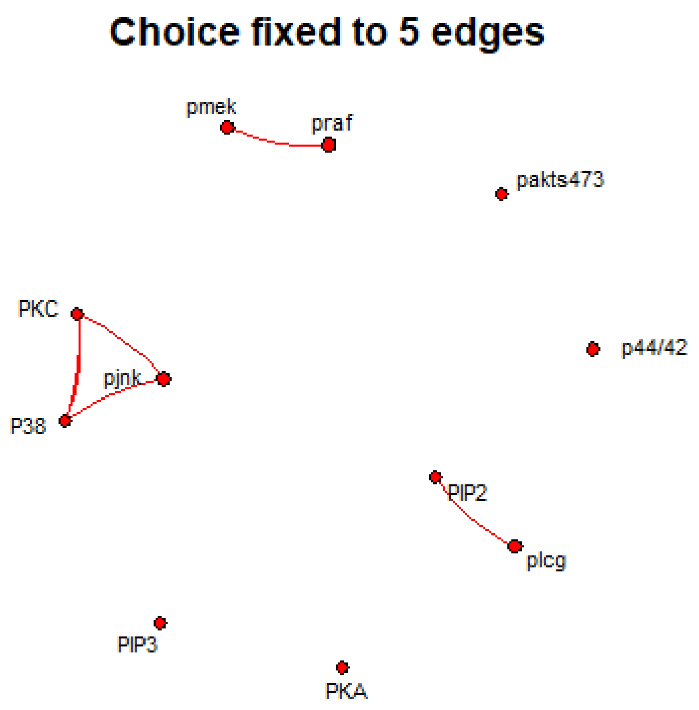
```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
```

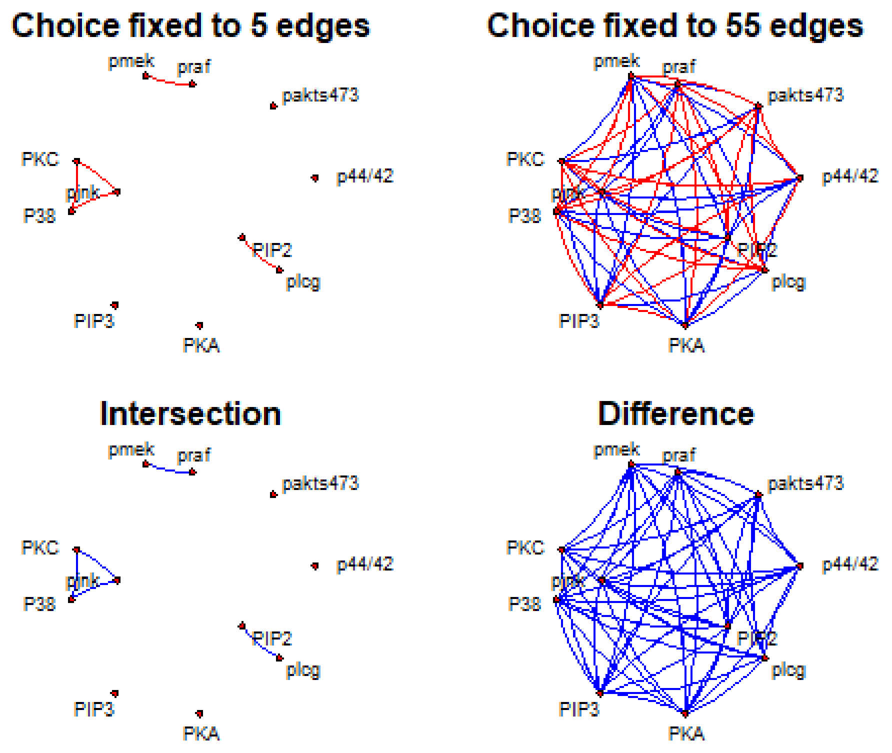
```
##
##        0.6          6     -59668
##
## Found a network with 6 edges.
```





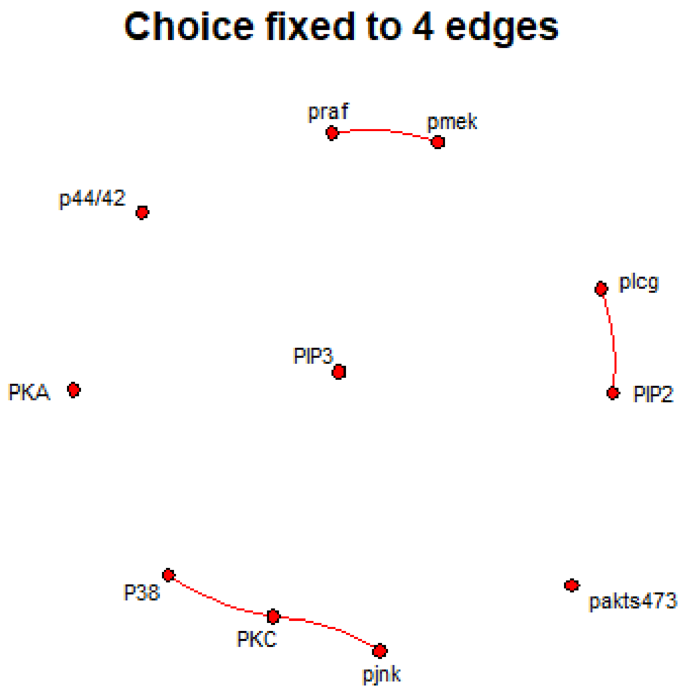
```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##        0.7          5     -62562
##
## Found a network with 5 edges.
```

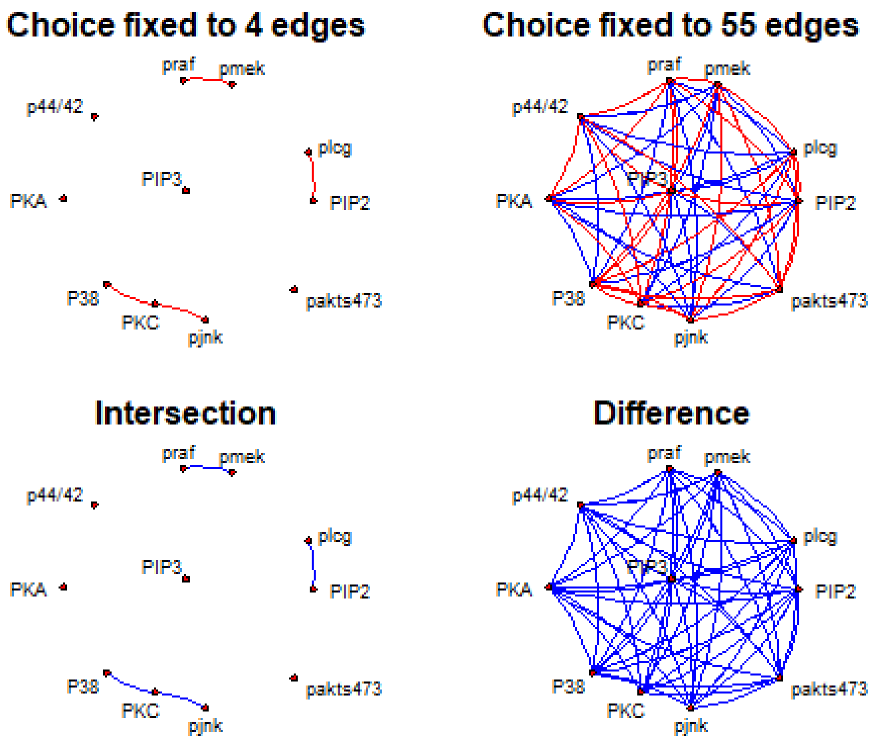




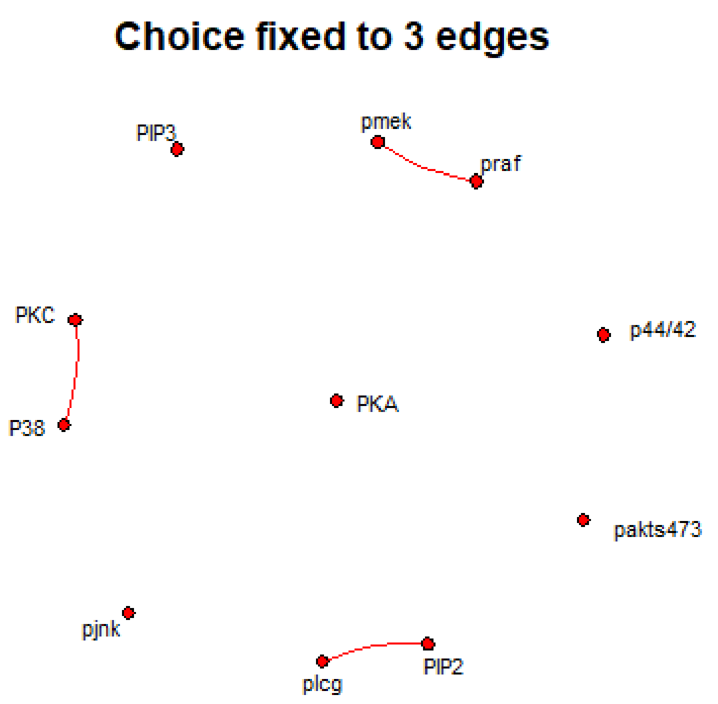
```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |     edges | criteria
```

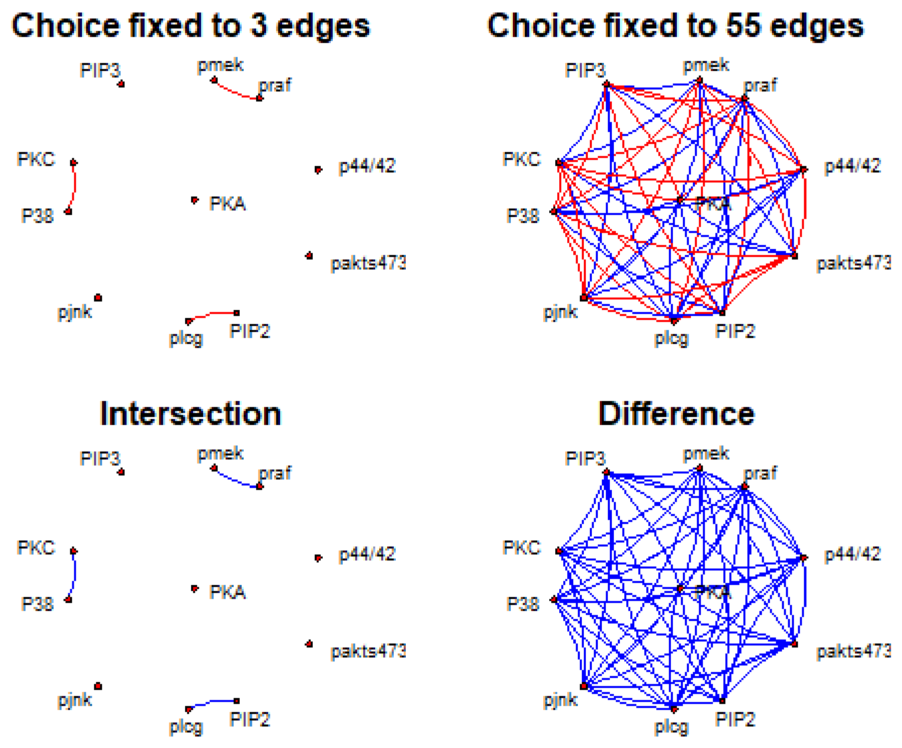
```
##
##         0.8           4      -65110
##
## Found a network with 4 edges.
```





```
##
## Network Inference: graphical.lasso with AND symmetrization rule applied
##
## |  penalty |    edges | criteria
##
##        0.9          3     -67407
##
## Found a network with 3 edges.
```





5. Discuter vos résultats.

Comparaison entre les graphes inférés de pénalité différentes.Les couleurs, rouge et bleu, représentent le signe de la corrélation partielle(négative ou positive).On peut endéduire que plus l'ecard est pétite entre les pénalités,plus il y'a de d'intersection communes. La complexité des réseaux reflète la multitude des fonctions qu'ils régulent et rend leur caractérisation difficile à interpreter.

Chercher la structure du graphe de concentration revient à chercher les coeffs non nuls de la regression de chaque variable sur les autres.Et d'autre part on cherche à trouver la pénalité qui permet de mieux représenter la structure de notre graphe

Quelle pénalité choisir? sélection d'un graphe parmi les familles construites par minimisation d'un critère pénalisé.

- Pour $\alpha \geq 1$, le problème de minimisation est convexe.
- Pour $\alpha \leq 1$, la solution du problème est sparse.

un autre critère de sélection de la pénalité est BIC et l'AIC.