Vignette : the Patterns package

Frédéric Bertrand, Nicolas Jung, Laurent Vallat, Myriam Maumy-Bertrand

June 16, 2013

Contents

1 Overview

In a cell, after a specific activation, a gene contained in the DNA can be expressed as RNA molecules that are later traduced in proteins that will sustain the cell response Crick et al. (1970).

Cells are in continuous contact with their environment within the organism and display an adapted response to its modifications Barabási and Oltvai (2004). For this, each transient environmental modification activates surface cell receptors (and co-receptors) that induce multiple integrated signaling cascades whose ultimate events are expression of specific genes and proteins (transcriptional factors). These first transcriptional factors (TF) induce the expression of other genes within the cell. Some of these genes code themselves for TF or transcriptional regulators (TR) that induce sequential activation of other genes. At the end, concerted expression of these multiple genes induces protein expressions that are the substratum of the adapted cellular reaction to the initial stimulus.

One Common tool to analyze such complex systems is regulatory networks (RN). When studying transcriptional data, this RN is called a gene regulatory network (GRN) in which the vertex represent genes and edges represent potential (orientated) interactions between these genes.

Since the emergence of high-throughput technologies that allow measuring simultaneously expression of thousands of genes, many tools have been developed to learn gene expression profiles and reverse-engineer their underlying gene regulatory network (GRN) (Bansal et al., 2007). These tools are either based on static co-expression methods or, if the biological phenomenon shows any temporality, time dependent methods. While the former relies on the assumption that co-expressed genes share some biological characteristics, the latter is grounded on a directed network with temporal dependencies.

The `Patterns` package is a tool designed to infer such directed networks with temporal dependencies.

Another important distinction should be made between exogenous stress (e.g., growth response) and endogenous phenomenon (e.g., cell cycle) (Zhu et al., 2007 and Luscombe et al., 2004). This leads to different network topologies: in exogenous stress, networks' topologies seem to have larger hubs and shorter paths through temporal dependant transcriptional waves. This results in a quick response to environmental modifications (Luscombe et al., 2004) and those networks are often called "cascade networks".

The `Patterns` package is based on the idea that genes may be assigned to temporal clusters which are used to enforce temporal causality in the network. It has been designed to analyze temporal microarray datasets, allowing gene selection, temporal cluster assignment, reverse-engineering of the GRN and predicting the effect of biological intervention experiments. This package also features a temporal synthetic simulation tool. The biological interpretations are made easier and comprehensive thanks to graphical outputs.

The `Patterns` package is a tool to analyze microarray data and model directed networks with temporal dependencies.

2 Installation requirements

Following software is required to run the `Patterns` package:

- R (> 2.14.2). For installation of R, refer to http://www.r-project.org.
- R-packages: `abind` ; `animation` ; `cluster` ; `datasets` ; `graphics` ; `grDevices` ; `igraph` ; `lars` ; `lattice` ; `limma`* ; `magic` ; `methods` ; `nnls` ; `plotrix` ; `splines` ; `stats` ; `stats4` ; `survival`* ; `tnet` ; `utils` ; `VGAM`.

To install them :

- without stars:

```
> install.packages("name_of_the_package")
```

- with one star:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("name_of_the_package")
```

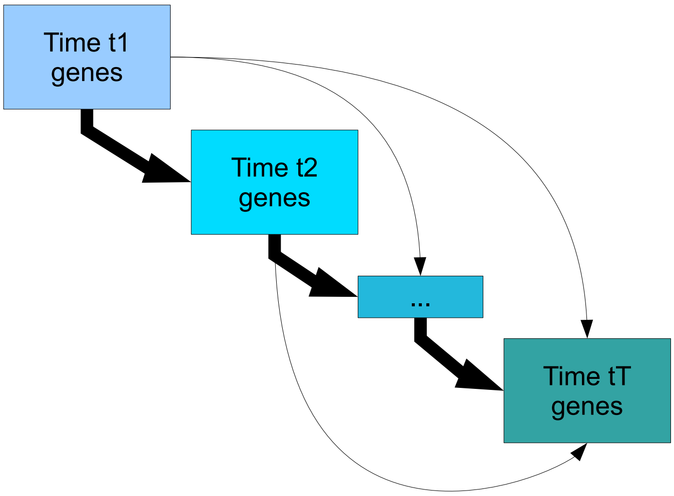


Figure 1: Cascade networks are temporal nested networks

Once the `Patterns` package is installed, you can load the package by:

```
> library(Patterns)
```

3 Data pre-processing

To illustrate our approach we will analyze a microarray data set. This data data set has initially be published in Vallat et al. (2007). Our data set is separated in two files: the first, `micro_S`, corresponds to the stimulated gene expressions while the second, `micro_US`, corresponds to the unstimulated gene expressions. In other words, `micro_US` is the control data set. You can load these data by:

```
> data(micro_S)
> data(micro_US)
```

Each of the these data sets corresponds to 54613 genes measured through 4 time points and 6 subjects (we have repeated longitudinal data). These data have have for biological model chronic lymphocytic leukemia ; for further details, see Vallat et al. (2007) or Vallat et al. (2013).

These data need to be coerced into a `micro_array` class. The matrix with the microarray measurements has to be of size $N \times K$ where N is the number of genes and $K = T \times P$ where T stands for the number of time points and P for the number of subjects. The first T colomns are the gene expressions for subject 1, the following T are the gene expressions for subject 2... In our case:

```
> colnames(micro_S)
```

```
 [1] "N1_S_T60"  "N1_S_T90"  "N1_S_T210" "N1_S_T390"
 [5] "N2_S_T60"  "N2_S_T90"  "N2_S_T210" "N2_S_T390"
 [9] "N3_S_T60"  "N3_S_T90"  "N3_S_T210" "N3_S_T390"
[13] "N4_S_T60"  "N4_S_T90"  "N4_S_T210" "N4_S_T390"
[17] "N5_S_T60"  "N5_S_T90"  "N5_S_T210" "N5_S_T390"
[21] "N6_S_T60"  "N6_S_T90"  "N6_S_T210" "N6_S_T390"
```

To coerce the data toward a `micro_array` class, you may just use the `as.micro_array` function:

```
> micro_S<-as.micro_array(micro_S,time=c(60,90,210,390),subject=6)
> micro_US<-as.micro_array(micro_US,time=c(60,90,210,390),subject=6)
```

In addition of the matrix of microarray measurements, this class also contains the name of genes, their group, the first time at which they are expressed, the time points at which they are measured, and the number of subjects. Primarily, method `print` summarizes these informations:

```
> print(micro_S)
```

```
This is a micro_array S4 class. It contains :
 - (@microarray) a matrix of dimension  54613 * 24
          .... [gene expressions]
 - (@name) a vector of length  54613  .... [gene names]
 - (@group) a vector of length  1  .... [groups for genes]
 - (@start_time) a vector of length  1
          .... [first differential expression for genes]
 - (@time)a vector of length  4  .... [time points]
 - (@subject) an integer  .... [number of subject]
```

While method `print` gives the structure of the object, method `head` gives an overview of the data:

```
> head(micro_S)
```

```
The matrix :

          N1_S_T60 N1_S_T90 N1_S_T210
1007_s_at    136.1    116.6     127.6
1053_at       32.0     43.3      31.3
117_at        78.0     63.5      57.9
121_at       201.8    209.2     208.8
1255_g_at     16.3      8.0      15.8
1294_at      196.8    198.7     163.9
...

Vector of names :
[1] "1007_s_at" "1053_at"   "117_at"    "121_at"
[5] "1255_g_at" "1294_at"
...
Vector of group :
[1] 0
```

```
...
Vector of starting time :
[1] 0
...
Vector of time :
[1]  60  90 210 390

Number of subject :
[1] 6
```

Entries `Vector of group` and `Vector of starting time` are set to 0 because they are no yet defined. They will be completed automatically when using gene selection functions of this package. Otherwise, it should be completed by the user.

Once data coerced into the `micro_array` class, this package allows doing gene selection and reverse-engineering ; note that gene selection requires two sets of data and will select genes that are differentially expressed in one condition against the other.

4 Gene selection

Gene selection requires two sets of data and will select genes that are differentially expressed in one condition against the other ; if unstimulated control dataset is omitted, it is remplaced with a null data set.

In this package gene selection mainly relies on the R-bioconductor `limma` package Smyth (2005). The `limma` package allows selecting genes that are differentially expressed between two conditions. In our case, these two conditions are "*stimulated*" and "*unstimulated*". The method relies on linear models and on improved bayesian t-tests ; refer to Smyth (2005) for details. Basically, to find the 50 more significant expressed genes you will use:

```
> Selection<-geneSelection(M1=micro_S,M2=micro_US,tot.number=50,data_log=TRUE)
```

The `data_log` option (default to TRUE) indicates that the data are logged before analysis. This function returns an object of class `micro_array`, with the difference "stimulated" (S) minus 'unstimulated" (US) of the 50 more significant expressed genes ; as the `data_log` option is here activated, we get:

$$\log(S) - \log(US) = \log\left(\frac{S}{US}\right).$$

Notice that the `group` and `start_time` are filled out automatically.

Applying the `summary` method prints the structure of Pearson linear correlation for subjects (see graphic 2) and the structure of Pearson linear correlation for genes (see graphic 3):

```
> summary(Selection)
```

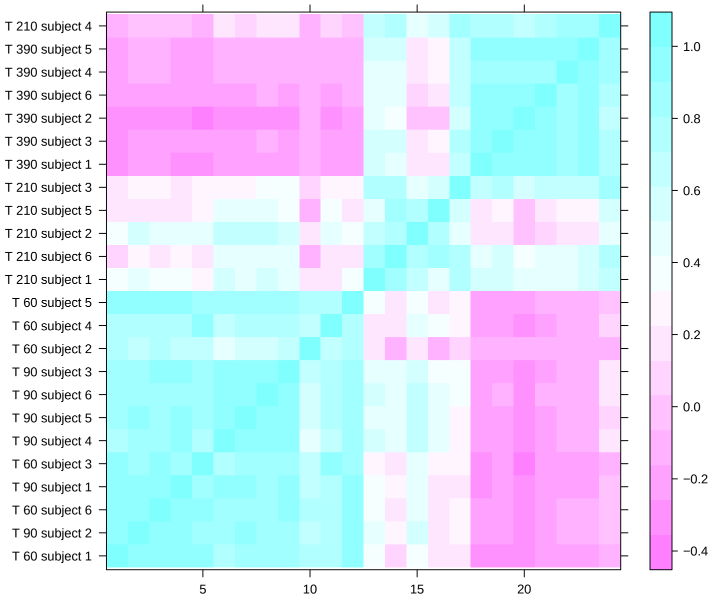


Figure 2: Correlation between subjects

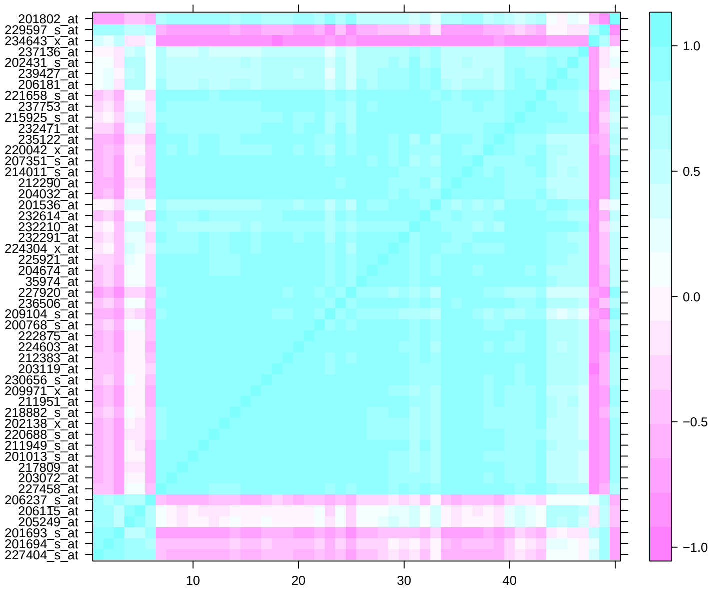


Figure 3: Correlation between genes

Note that a hierarchical clustering (function agnes of package cluster) is performed before plotting the result. This is necessary to point out some structures, as correlated objects will be close in the graph.

If we want to select genes that are differentially expressed at specific time points we use the option wanted.patterns:

```
> #If we want to select genes that are differentially
> #at time t60 or t90 :
> Selection<-geneSelection(M1=micro_S,M2=micro_US,tot.number=30,
   wanted.patterns=
   rbind(c(0,1,0,0),c(1,0,0,0),c(1,1,0,0)))
```

You may want forbid some patterns thanks to the forbidden.patterns option.

If we wish select genes that have a differential maximum of expression at a specific time point, we may use the genePicSelection method. Basically, this function selects genes that are differentially expressed at desired time point, and which differential expression is significantly higher at this time point:

```
> Selection<-genePicSelection(M1=micro_S,M2=micro_US,1,
   abs_val=FALSE,alpha_diff=0.01)
```

We can now compute a effective selection. As shown in graphic 4, the early time points (t_1 =60 and t_2 =90) are correlated together and the later time points (t_3 =210 and t_4 =390) are correlated together ; this is a fact that is well known in the literature Yosef and Regev (2011). As early genes expressions are lower than later gene expressions, we select them separately:

```
> #Select early genes (t1 or t2)
> Selection1<-geneSelection(M1=micro_S,M2=micro_US,20,
   wanted.patterns=
   rbind(c(0,1,0,0),c(1,0,0,0),c(1,1,0,0)))
> #Section genes with first significant differential
> #expression at t1:
>
> Selection2<-geneSelection(M1=micro_S,M2=micro_US,20,
   pic=1)
> #Section genes with first significant differential
> #expression at t2:
>
> Selection3<-geneSelection(M1=micro_S,M2=micro_US,20,
   pic=2)
> #Select later genes (t3 or t4)
> Selection4<-geneSelection(M1=micro_S,M2=micro_US,50,
   wanted.patterns=
   rbind(c(0,0,1,0),c(0,0,0,1),c(1,1,0,0)))
```

We then make the union between these different selection:

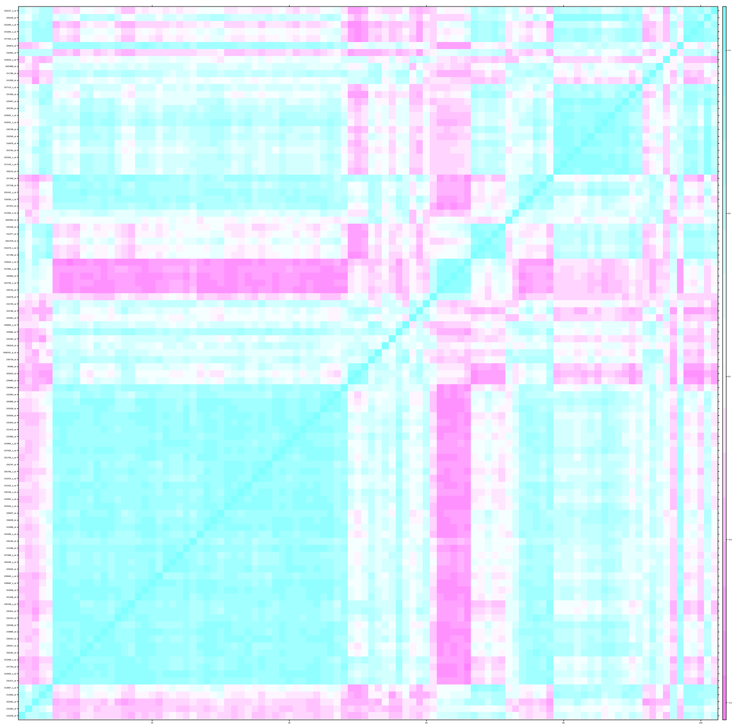


Figure 4: Correlation structure of the final selection

```
> Selection<-unionMicro(Selection1,Selection2)
> Selection<-unionMicro(Selection,Selection3)
> Selection<-unionMicro(Selection,Selection4)
> print(Selection)

This is a micro_array S4 class. It contains :
 - (@microarray) a matrix of dimension  102 * 24
          .... [gene expressions]
 - (@name) a vector of length  102  .... [gene names]
 - (@group) a vector of length  102  .... [groups for genes]
 - (@start_time) a vector of length  102
          .... [first differential expression for genes]
 - (@time)a vector of length  4  .... [time points]
 - (@subject) an integer  .... [number of subject]

> #Prints the correlation graphics Figure 4:
> summary(Selection,3)


> Selection2gp<-unionMicro(Selection1,Selection2)
> Selection2gp<-unionMicro(Selection2gp,Selection3)
```

5 Gene regulatory network reverse engineering

5.1 Theorical background

Gene regulatory network reverse engineering relies on a lasso penalized regression Tibshirani (1996). The Lasso estimation is given by:

$$\hat{\boldsymbol{\beta}}^L(\lambda) = \underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\operatorname{argmin}} \left[\sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \|\boldsymbol{\beta}\|_1 \right], \tag{1}$$

with λ a non negative scalar that determines the level of the constraints. We remark that:

- When $\lambda = 0$, $\hat{\boldsymbol{\beta}}^L$ is ordinary least square estimation.
- When $\lambda = +\infty$, we get $\hat{\boldsymbol{\beta}}^L = \mathbf{0}_p$.

The Lasso regression has two main advantages :

1. it allows dealing with ill posed problems, where the number of observations is inferior to the number of variables,
2. it allows performing variable selection.

The Lasso regression can also be written in the following form:

$$\hat{\boldsymbol{\beta}}^L(\lambda) = \underset{\boldsymbol{\beta}\in\mathbb{R}^p \quad \|\boldsymbol{\beta}\|_1 \leqslant \tilde{\lambda}}{\operatorname{argmin}} \left[\sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right]. \tag{2}$$

These two forms (equation (1) and (2)) are equivalent in the sense that for each non negative λ there exists a non negative $\tilde{\lambda}$ leading to the same solution.

Based on the Lasso regression (equation 2), our model is very close to the model proposed in Vallat et al. (2013). It can be written:

$$\underset{\omega_{ij}\in\mathbb{R},\ 1\leqslant i,j\leqslant N_{sel}}{\operatorname{argmin}} \left[\sum_{j=1}^{N_{sel}} \left(\tilde{\boldsymbol{x}}_{jp.} - \sum_{i=1}^{N_{sel}} F_{m(i)m(j)} \omega_{ij} \boldsymbol{x}_{ip.} \right)^2 \right],$$

with the constraint :

$$\forall j = 1, ..., N_{sel}, \qquad \sum_{i=1}^{N_{sel}} \omega_{ij} \leqslant \lambda_j,$$

where:

$$\tilde{\boldsymbol{x}}_{jp.} = \begin{pmatrix} x_{jpt_1} \\ \vdots \\ x_{jpt_T} \end{pmatrix} \quad \text{and} \quad \boldsymbol{x}_{ip.} = \begin{pmatrix} x_{jpt_1} \\ \vdots \\ x_{jpt_T} \end{pmatrix},$$

with:

- x_{jpt_k} is the expression of gene j for patient p at time point t_k,
- $m(\bullet)$ is the function that maps a gene to its categorical label,
- $F_{m(i)m(j)}$ is a N_{gp} squared matrix that describes the action of genes,
- ω_{ij} is the strength of the connection from gene i toward gene j,
- λ_1,...,λ_j are non negative constants.

So, $\tilde{\boldsymbol{x}}_{jp.}$ is the regulated gene and $\boldsymbol{x}_{ip.}, i = 1..N$ are the regulators. Notice that matrix $F_{m(i)m(j)}$ permits to the link between genes i and j to evolves across time.For the time being, we do not allow auto-regulations for genes $(m(i) = m(j) \Rightarrow F_{m(i)m(j)} = 0)$.

We solve this problem to a coordinate ascent approach, by iteratively supposing the F matrices or the ω_{ij} matrices known. The result of the optimization is a connectivity network described by the nonzero elements of ω_{ij} combined with a set of cluster-dependent interaction models described by the set $F_{m(i)m(j)}$.

However, if clusters are sufficiently homogeneous, inference of matrices $F_{m(i)m(j)}$ doesn't depend on which genes are active (i.e. which $\omega_{ij} \neq 0$). That's why a non iterative algorithm is proposed in which estimation of of matrices $F_{m(i)m(j)}$ precedes estimation of matrix Ω.

To get a more robust result, at each step, the estimation of matrices $F_{m(i)m(j)}$ is done several times throughout cross-validation. Furthermore, to avoid computational issues, the new solution is chosen by a linear combination between the old and the new solution.

The three functions `replaceBand`, `replaceUp` and `replaceDown` can be used to create F matrices with custom band values. The two functions `CascadeFinit` and `CascadeFshape` will provide the users with band F matrices useful for cascade networks.

5.2 Performing the algorithm

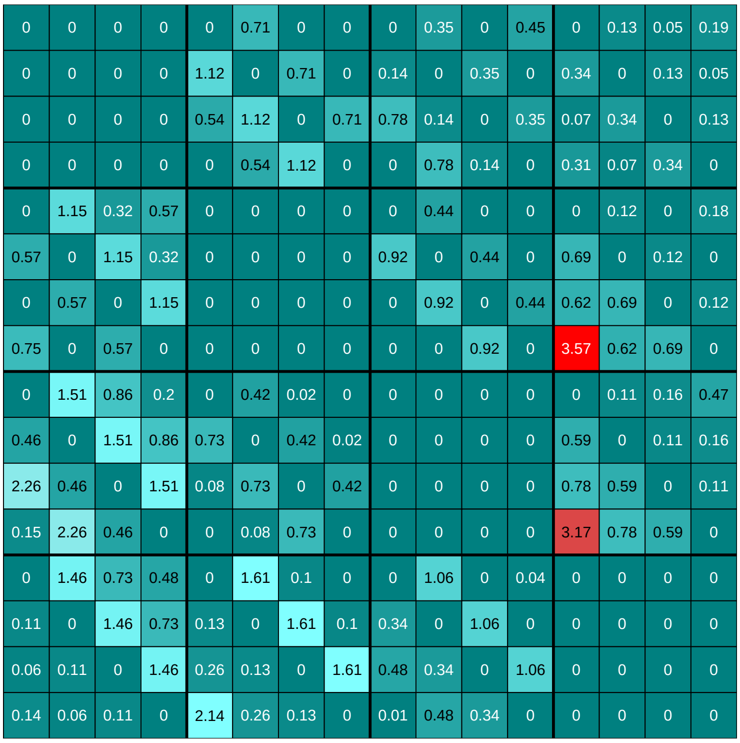
To perform this algorithm on our data:

```
> network<-inference(Selection)
> networkCascade<-inference(Selection,Finit=CascadeFinit(4,4),Fshape=CascadeFshape(4,4))
> network2gp<-inference(Selection2gp)
```

We can plot the resulting network (figure 8) and a representation of F matrices (figure 5) simply using the `plot` method:

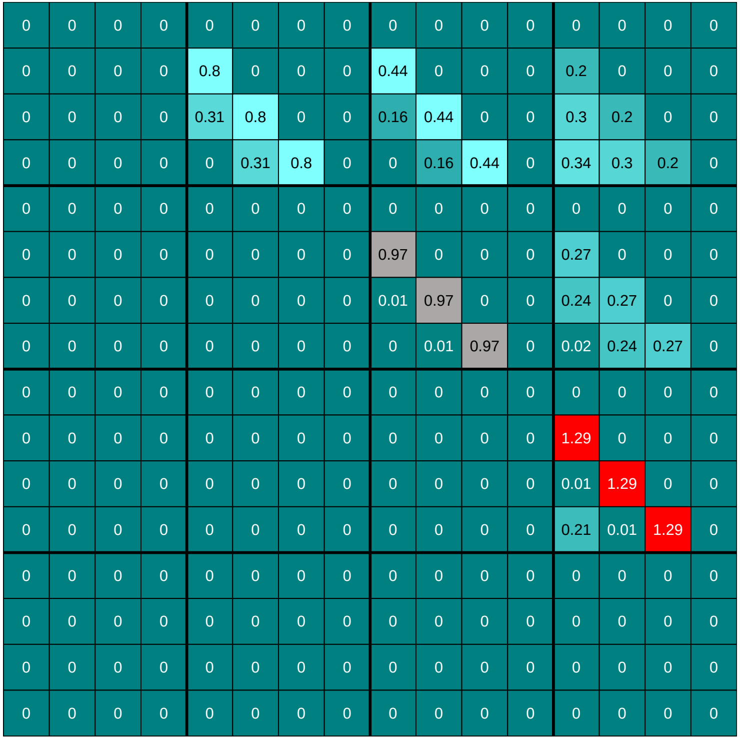
```
> plot(network,choice="F")
> plot(networkCascade,choice="F")
> plot(network2gp,choice="F")
> plot(network,choice="network",gr=Selection@group)
> plot(network2gp,choice="network",gr=Selection2gp@group)
```

Note that all network plots are computed using the Igraph R package Csardi and Nepusz (2006).



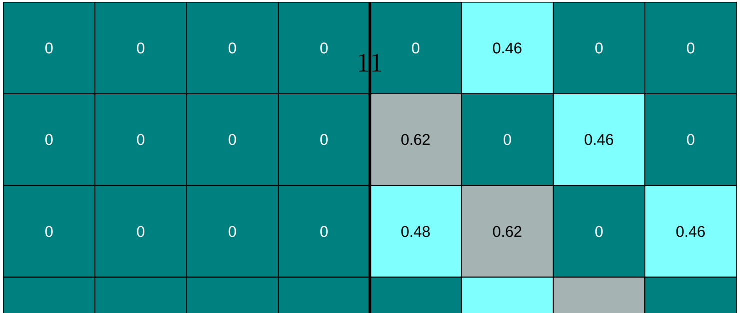
0	0	0	0	0	0.71	0	0	0	0.35	0	0.45	0	0.13	0.05	0.19
0	0	0	0	1.12	0	0.71	0	0.14	0	0.35	0	0.34	0	0.13	0.05
0	0	0	0	0.54	1.12	0	0.71	0.78	0.14	0	0.35	0.07	0.34	0	0.13
0	0	0	0	0	0.54	1.12	0	0	0.78	0.14	0	0.31	0.07	0.34	0
0	1.15	0.32	0.57	0	0	0	0	0	0.44	0	0	0	0.12	0	0.18
0.57	0	1.15	0.32	0	0	0	0	0.92	0	0.44	0	0.69	0	0.12	0
0	0.57	0	1.15	0	0	0	0	0	0.92	0	0.44	0.62	0.69	0	0.12
0.75	0	0.57	0	0	0	0	0	0	0	0.92	0	3.57	0.62	0.69	0
0	1.51	0.86	0.2	0	0.42	0.02	0	0	0	0	0	0	0.11	0.16	0.47
0.46	0	1.51	0.86	0.73	0	0.42	0.02	0	0	0	0	0.59	0	0.11	0.16
2.26	0.46	0	1.51	0.08	0.73	0	0.42	0	0	0	0	0.78	0.59	0	0.11
0.15	2.26	0.46	0	0	0.08	0.73	0	0	0	0	0	3.17	0.78	0.59	0
0	1.46	0.73	0.48	0	1.61	0.1	0	0	1.06	0	0.04	0	0	0	0
0.11	0	1.46	0.73	0.13	0	1.61	0.1	0.34	0	1.06	0	0	0	0	0
0.06	0.11	0	1.46	0.26	0.13	0	1.61	0.48	0.34	0	1.06	0	0	0	0
0.14	0.06	0.11	0	2.14	0.26	0.13	0	0.01	0.48	0.34	0	0	0	0	0

Figure 5: The F matrices



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0.8	0	0	0	0.44	0	0	0	0.2	0	0	0
0	0	0	0	0.31	0.8	0	0	0.16	0.44	0	0	0.3	0.2	0	0
0	0	0	0	0	0.31	0.8	0	0	0.16	0.44	0	0.34	0.3	0.2	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.97	0	0	0	0.27	0	0	0
0	0	0	0	0	0	0	0	0.01	0.97	0	0	0.24	0.27	0	0
0	0	0	0	0	0	0	0	0	0.01	0.97	0	0.02	0.24	0.27	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1.29	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0.01	1.29	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0.21	0.01	1.29	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6: The F matrices



0	0	0	0	0	0.46	0	0
0	0	0	0	0.62	0	0.46	0
0	0	0	0	0.48	0.62	0	0.46

The number of edges in the network makes the message difficult to interpret ; and as we shall see in the next section, results in term of predictive positive value and F-score can be improved when choosing a right cutoff level. Using the `nv` option, we shall choose a cutoff under which the regression coefficients (ω_{ij}) are set to 0. In figure 10 a cutoff of 0.2 is chosen.

5.3 Choosing the cutoff

The difficulty is now to choose the best cutoff. As a starting point, we propose method `evolution`, that allows the user to see, in a html page, the evolution of the network when the cutoff is growing up. When the `fix` option is set to `FASLE`, at each step the position of the genes are re-calculated.

```
> evolution(network,seq(0,0.4,by=0.01),gr=Selection@group,fix=TRUE)
> evolution(network,seq(0,0.4,by=0.01),gr=Selection@group,fix=FALSE)
```

To see the result of these functions, go to :

- `http://www-irma.u-strasbg.fr/~njung/evolution_fix_true/evol.html` : here the `fix` option is set to `TRUE`.
- `http://www-irma.u-strasbg.fr/~njung/evolution_fix_false/evol.html`: here the `fix` option is set to `FALSE`.

As it is well known, gene regulatory networks are scale-free Jeong et al. (2000). The notion of scale freeness in networks relies on the probability distribution of the number of outgoing edges. A network is called scale free when this distribution is a power law distribution Clauset et al. (2009). As this family of law is large, it is difficult to test such an hypothesis. We used the test proposed in Clauset et al. (2009):

```
> evol_cutoff<-cutoff(network)
> nv<-0.07
```

We prefer plotting the smooth interpolation rather than the exact values, as our interest relies mostly on the trend. In figure 11, if we apply some heuristic scree test, we will choose a cutoff of $nv = 0.07$.

5.4 Analyzing the network

One may want to know which genes are important in the network. In our representation, the bigger the vertex the larger the number of outgoing edges. Indeed, genes with many outgoing edges, the hubs, are important in the network. But what about genes that control these hubs ? The `analyze_network` method allows computing different indicators:

- betweenness : it is a measure of the node centrality. It is calculated, for node n, by the following formula:

  $$\sum_{s\neq t\neq n} \frac{\sigma_{st}(n)}{\sigma_{st}}$$

  where σ_{st} is the number of shortest way between s and t, and $\sigma_{st}(n)$ is the number of shortest way between s and t passing by n ;

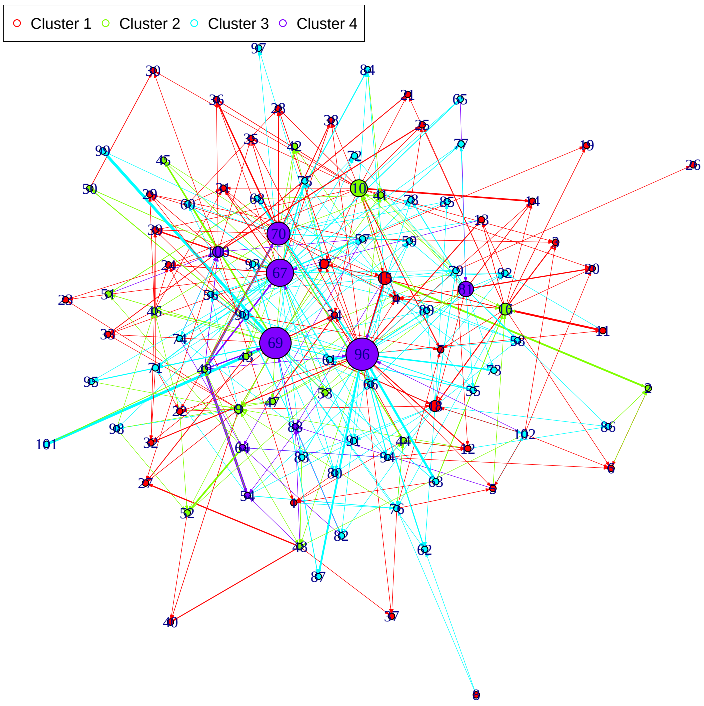


Figure 10: The resulting network with a cutoff of 0.07

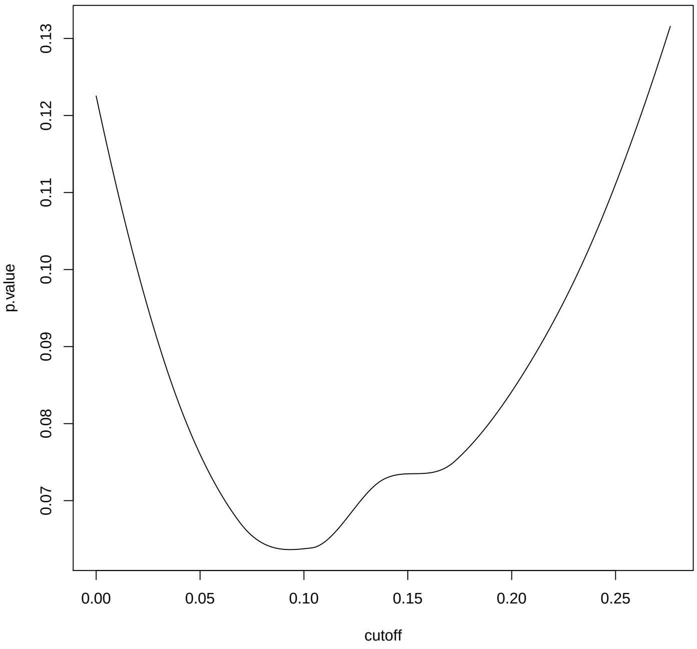


Figure 11: Evolution of scale freeness of the network in function of the cutoff. The p-value corresponds to the adequacy of the data to a power law distribution.

- degree : the number of outgoing edges ;
- output : the sum of weights of outgoing genes ;
- closeness : it is a measure of the distance (in terms of shortest path) of a gene to others.

As our network is weighted we used specific measures developed in Opsahl (2009).

```
> analyze<-analyze_network(network,nv)
> head(analyze)

     node betweenness degree     output  closeness
[1,]    1         174      2 0.15671244 21.2920924
[2,]    2           7      1 0.08785171  0.6077887
[3,]    3           0      0 0.00000000  0.0000000
[4,]    4          69      6 0.67694653 30.7597692
[5,]    5         235      3 0.32073348 29.0064119
[6,]    6          11      1 0.09393366  0.6498658
```

Note that one can plot the network and modulate the size of the vertex following one of this measure, using the `weight.node` option.

Using again the package `animation`, we can see how the signal spreads in the network by turning to `TRUE` the option `ani`:

```
> plot(network,nv=nv,gr=Selection@group,ani=TRUE)
```

Result is available at `http://www-irma.u-strasbg.fr/~njung/network_spread/spread.html`.

The method `plot` has basically two steps: 1- it calculates the position of the vertex, 2- it plots the graph. In some case, it is interesting to produce two plots of a same network without changing vertex positions. Here is a way to do that, using the `ini` option of method plot:

```
> P<-position(network,nv=nv)
> #plotting the network with the group coloring:
> plot(network,nv=nv,gr=Selection@group,ini=P)
> #plotting the network without the group coloring:
> plot(network,nv=nv,ini=P)
```

However, we didn't develop all possibilities of the `plot` option ; for more possibilities, please refer to the manual.

6 Prediction

Once the network reverse-engineered, we want to be able to know the impact of perturbation in this network. For example, what would happen if gene 16 is perturbed ? First the `geneNeighborhood` method allows determining which are the neighborhood of gene 16.

```
> geneNeighborhood(network,targets=16,nv=nv,ini=P,
          label.hub=TRUE,label_v=Selection@name)
> #label.hub: only hubs vertex should have a name
> #label_v: name of the vertex
```

We then can predict the changes in the gene expression. Suppose gene 16 is knocked-out

```
> prediction_ko16<-predict(Selection,network,nv=nv,targets=16)
> prediction_ko16_2gp<-predict(Selection2gp,network2gp,nv=nv,targets=16)
```

We can then plot the result:

```
> #We plot the results ; here for example we see changes at time point t2
> plot(prediction_ko16,time=2,ini=P,label.hub=TRUE,label_v=Selection@name)
> plot(prediction_ko16_2gp,time=2,ini=P,label.hub=TRUE,label_v=Selection2gp@name)
```

7 Simulation

To simulate gene expressions based on a gene regulatory network, we first have to generate the network. Here, we implemented an algorithm that is inspired by the *preferential attachment* from Barabasi Jeong et al. (2007). We adapted this algorithm in our case of temporal nested networks.

We then use our linear model to make some simulations, using Laplace laws to initiate the algorithm.

```
> #We set the seed to make the results reproducible
> set.seed(1)
> #We create a random scale free network
> Net<-network_random(
          nb=100,
          time_label=rep(1:4,each=25),
          exp=1,
          init=1,
          regul=round(rexp(100,1))+1,
          min_expr=0.1,
          max_expr=2,
          casc.level=0.4
          )
> #We change F matrices
> ngrp<-4
> T<-4
> F<-array(0,c(T,T,ngrp*ngrp))
> for(i in 1:(ngrp*ngrp)){diag(F[,,i])<-1}
> F[,,2]<-F[,,2]*0.2
> F[2,1,2]<-1
> F[3,2,2]<-1
> F[,,4]<-F[,,2]*0.3
> F[3,1,4]<-1
```

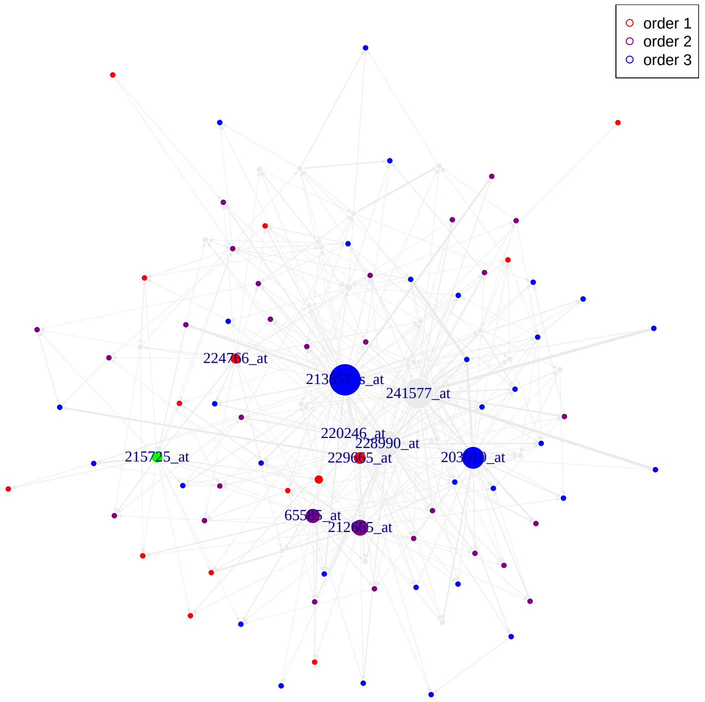


Figure 12: Neighborhood of gene 16

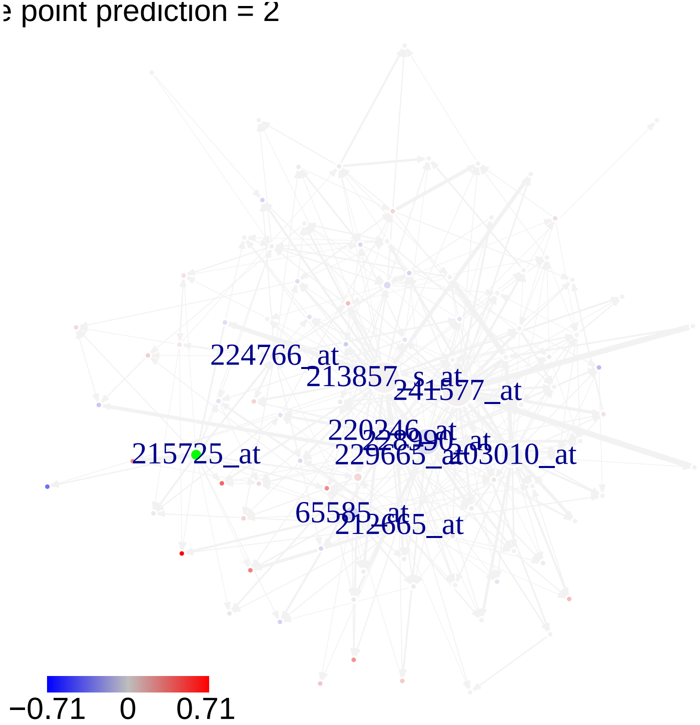
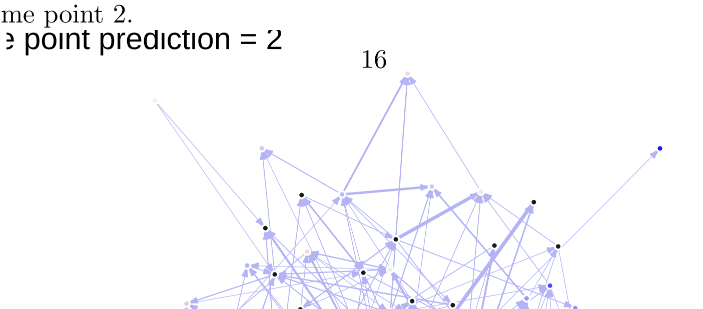


Figure 13: Perturbation of the network consecutively to the knock out of gene 16 at time point 2.



```
> F[,,5]<-F[,,2]
> Net@F<-F
> #We simulate gene expression according to the network Net
> M<-gene_expr_simulation(
          network=Net,
          time_label=rep(1:4,each=25),
          subject=5,
          level_pic=200)

> #We infer the new network
> Net_inf<-inference(M)

> #Comparing true and inferred networks
> F_score<-rep(0,200)
> #Here are the cutoff level tested
> test.seq<-seq(0,max(abs(Net_inf@network*0.9)),length.out=200)
> u<-0
> for(i in test.seq){
          u<-u+1
          F_score[u]<-compare(Net,Net_inf,i)[3]
   }

> #Choosing the cutoff
> cut.seq<-cutoff(Net_inf)
> plot(cut.seq$sequence,cut.seq$p.value.inter)
```

References

Bansal, M., Belcastro, V., Ambesi-Impiombato, A., and Di Bernardo, D. (2007). How to infer gene networks from expression profiles. *Molecular systems biology*, 3(1).

Barabási, A.-L. and Oltvai, Z. N. (2004). Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113.

Clauset, A., Shalizi, C. R., and Newman, M. E. (2009). Power-law distributions in empirical data. *SIAM review*, 51(4):661–703.

Crick, F. et al. (1970). Central dogma of molecular biology. *Nature*, 227(5258):561–563.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Jeong, H., Néda, Z., and Barabási, A.-L. (2007). Measuring preferential attachment in evolving networks. *EPL (Europhysics Letters)*, 61(4):567.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., and Barabási, A.-L. (2000). The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654.

Luscombe, N. M., Babu, M. M., Yu, H., Snyder, M., Teichmann, S. A., and Gerstein, M. (2004). Genomic analysis of regulatory network dynamics reveals large topological changes. *Nature*, 431(7006):308–312.

Opsahl, T. (2009). *Structure and Evolution of Weighted Networks*. University of London (Queen Mary College), London, UK.

Smyth, G. K. (2005). Limma: linear models for microarray data. In Gentleman, R., Carey, V., Dudoit, S., Irizarry, R., and Huber, W., editors, *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pages 397–420. Springer, New York.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Vallat, L., Kemper, C. A., Jung, N., Maumy-Bertrand, M., Bertrand, F., Meyer, N., Pocheville, A., Fisher, J. W., Gribben, J. G., and Bahram, S. (2013). Reverse-engineering the genetic circuitry of a cancer cell with predicted intervention in chronic lymphocytic leukemia. *Proceedings of the National Academy of Sciences*, 110(2):459–464.

Vallat, L. D., Park, Y., Li, C., and Gribben, J. G. (2007). Temporal genetic program following b-cell receptor cross-linking: altered balance between proliferation and death in healthy and malignant b cells. *Blood*, 109(9):3989–3997.

Yosef, N. and Regev, A. (2011). Impulse control: temporal dynamics in gene transcription. *Cell*, 144(6):886–896.

Zhu, X., Gerstein, M., and Snyder, M. (2007). Getting connected: analysis and principles of biological networks. *Genes & development*, 21(9):1010–1024.

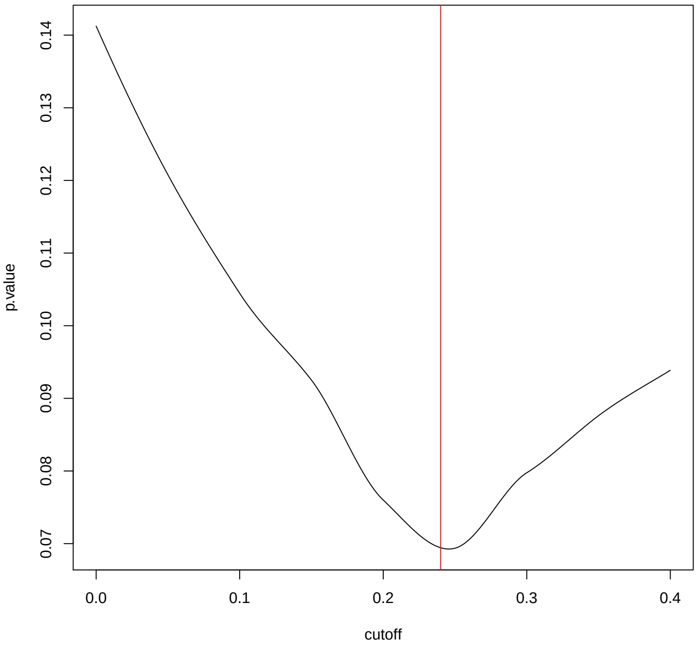


Figure 15: Evolution of the scale freeness of the network in function of the cutoff

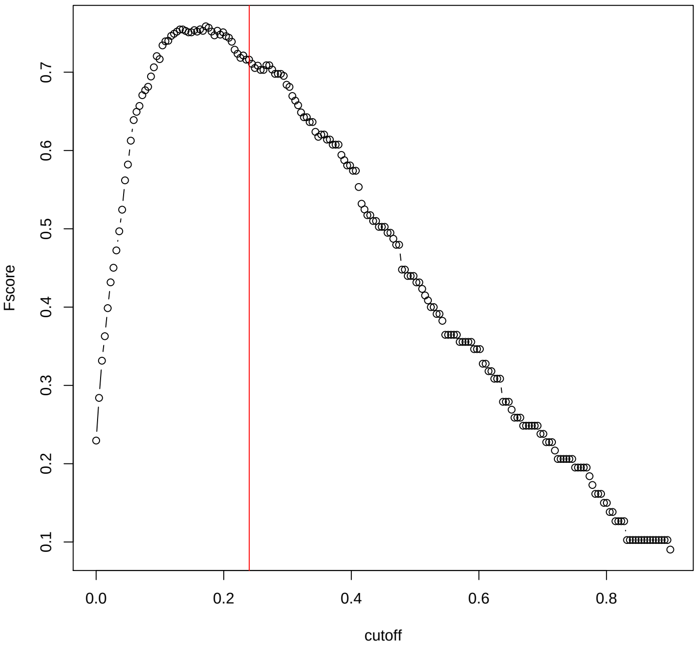


Figure 16: Evolution of F-score in function of the cutoff

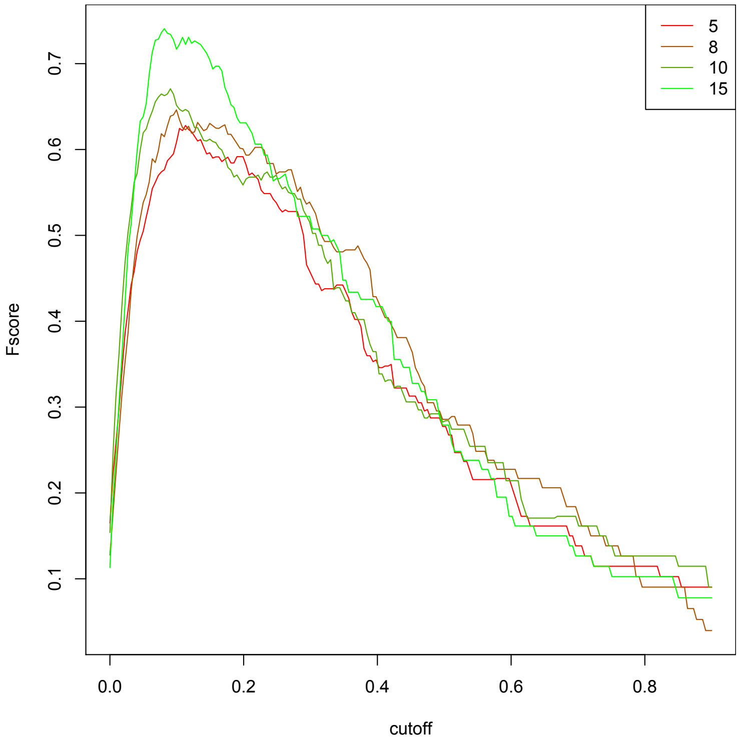


Figure 17: Evolution of F-score in function of the cutoff and the number of subject in the study