Additional application of the Cascade package to E-MTAB-1475 dataset

Nicolas Jung, Frédéric Bertrand,
Seiamak Bahram, Laurent Vallat and Myriam Maumy-Bertrand

March 24, 2014

The Cascade package has two vignettes and a manual:

- Introduction to the Cascade package with application to the GSE39411 dataset, available thanks to the R-command: `vignette("Cascade")`
- Additional application of the Cascade package to E-MTAB-1475 dataset, available thanks to the R-command: `vignette("E-MTAB-1475_re-analysis")`
- The manual for the Cascade package is available thanks to the R-command: `vignette("Cascade-manual")`

Contents

1 Introduction

We perform a reanalysis of "E-MTAB-1475" dataset (van den Ham *et al.*, 2013). This dataset is composed of temporal gene expressions from stimulated lymphocytes. The aim of the paper is to uncover the specific transcription that sustains the specific cellular T-lymphocyte differentiation in Th1 or Th2 T-lymphocyte cells. The experimental design is a follow:

- Gene expression of T-lymphocyte naïve cells (without any stimulation). Not time-repeated (this measure is replicated four times).
- Gene expression of T-lymphocyte cells after a neutral stimulation. These measurements are done after one, two, three and four days (each measure is replicated three times).
- Gene expression of T-lymphocyte cells differentiating themselves in Th1 T-lymphocyte cells following a neutral stimulation + a Th1 oriented stimulation. These measurements are done after one, two, three and four days (each measure is replicated three times).
- Gene expression of T-lymphocyte cells differentiating themselves in Th2 T-lymphocyte cells following a neutral stimulation + a Th2 oriented stimulation. These measurements are done after one, two, three and four days (each measure is replicated three times).

First, we configured our gene selection function with the same parameters as in the original paper (van den Ham *et al.*, 2013): false discovery rate (FDR) set to 0.05, minimal log-fold change (LFC) to 1.5 and we discarded the use of patterns in our package. As expected, we found the differentially expressed genes already highlighted in this paper.

Next, we focus on the cellular differentiation transcriptional program of the Th1 T-lymphocyte cells by studying the differentially expressed genes between the Neutral stimulation + Th1 versus the Neutral stimulation. We then reanalyzed the dataset taking advantage from the gene selection function of our package, according to their temporal patterns and differential expressions.

2 Importing the "E-MTAB-1475" dataset

The microarray datasets are stored online under the reference "E-MTAB-1475" at:

`http://www.ebi.ac.uk/arrayexpress/`

Before running the following code lines, you first have to download the appropriate dataset in your current working folder.

```
> library(ArrayExpress)
> MTAB1475 = getAE("E-MTAB-1475", type = "full",local=TRUE)
> cnames = getcolproc(MTAB1475);
> MTAB1475proc = procset(MTAB1475, cnames[2])
> #The columns order should follow the Cascade format order:
>
>
> exprsNeu <- exprs(MTAB1475proc)[,c(8,17,25,33,9,2,26,34,10,18,27,35)]
> exprsTh1 <- exprs(MTAB1475proc)[,c(11,19,28,36,12,20,3,4,13,21,29,37)]
> exprsTh2 <- exprs(MTAB1475proc)[,rep(c(14,15,16),rep(4,3))+c(0,8,16,24)]
> exprsNaive <- exprs(MTAB1475proc)[,c(1,5,6,7)]
```

Data are then converted in the Cascade format:

```
> library(Cascade)
> Neu_ma<-as.micro_array(exprsNeu,c(1,2,3,4),3)
> Th1_ma<-as.micro_array(exprsTh1,c(1,2,3,4),3)
> Th2_ma<-as.micro_array(exprsTh2,c(1,2,3,4),3)
> Naive<-as.micro_array(exprsNaive,c(1),4)
> M<-list(Naive,Neu_ma,Th1_ma,Th2_ma)
```

M is a list with all "micro_array" data corresponding to the four different conditions: naïve (control without any stimulation), neutral stimulation, neutral+TH1 oriented stimulation, neutral+TH2 oriented stimulation.

3 Gene selections and reverse-engineering of the networks

3.1 Note on the "geneSelection" function

The geneSelection function may be used specifying either a stimulated `micro_array` object and an unstimulated `micro_array` object (the control), either a list of `micro_array` objects and a list specifying the contrasts. In the second case, contrasts are specified using a list with the following specific form:

First element: "condition", "condition&time" or "pattern". The "condition" specification is used when the overall goal is to compare two conditions. The "condition&time" specification is used when comparing two conditions at two precise time points. The "pattern" specification allows to choose at which time points selected genes should be expressed or not.

Second element: a vector of length 2, corresponding to the two conditions that should be compared. If a non-temporal dataset is used as control, it should be the first element of the `micro_array` list and the option "`cont`=TRUE" should be used.

Third element: depends on the first element. This element is not needed if "condition" has been specified. If "condition&time" has been specified, then this is a vector containing the time point at which the comparison should be done. If "pattern" has been specified, then this is a vector of 0 and 1 of length T, where T is the number of time points. Time points where differential expression is wanted are provided with 1.

3.2 Selecting differentially expressed genes without specifying patterns

In the first part of our analysis, we configured our method with the same parameters as in the paper of van den Ham et al. (van den Ham *et al.*, 2013): FDR set to 0.05, minimal log-fold change to 1.5 and we discarded the use of patterns.

We define the following contrasts:

- neutral versus naive
- Th2 versus naive
- Th2 versus neutral
- Th1 versus naive
- Th1 versus neutral
- Th1 versus Th2

```
> naive_neutre<- list("condition",c(1,2))
> naive_th1<- list("condition",c(1,3))
> naive_th2<- list("condition",c(1,4))
> neutre_th1<- list("condition",c(2,3))
> neutre_th2<- list("condition",c(2,4))
> th2_th1<- list("condition",c(4,3))
```

We now use these same contrasts, but we focus on each day separately. Note: as a convention, when the control is not time-measured, it is considered that measures are done at day 0 (e.g. naïve condition).

```
> naive_neutre_d1<- list("condition&time",c(1,2),c(0,1))
> naive_neutre_d2<- list("condition&time",c(1,2),c(0,2))
> naive_neutre_d3<- list("condition&time",c(1,2),c(0,3))
> naive_neutre_d4<- list("condition&time",c(1,2),c(0,4))
> naive_th1_d1<- list("condition&time",c(1,3),c(0,1))
> naive_th1_d2<- list("condition&time",c(1,3),c(0,2))
> naive_th1_d3<- list("condition&time",c(1,3),c(0,3))
> naive_th1_d4<- list("condition&time",c(1,3),c(0,4))
> naive_th2_d1<- list("condition&time",c(1,4),c(0,1))
> naive_th2_d2<- list("condition&time",c(1,4),c(0,2))
> naive_th2_d3<- list("condition&time",c(1,4),c(0,3))
> naive_th2_d4<- list("condition&time",c(1,4),c(0,4))
> neutre_th1_d1<- list("condition&time",c(2,3),c(1,1))
> neutre_th1_d2<- list("condition&time",c(2,3),c(2,2))
> neutre_th1_d3<- list("condition&time",c(2,3),c(3,3))
> neutre_th1_d4<- list("condition&time",c(2,3),c(4,4))
> neutre_th2_d1<- list("condition&time",c(2,4),c(1,1))
> neutre_th2_d2<- list("condition&time",c(2,4),c(2,2))
> neutre_th2_d3<- list("condition&time",c(2,4),c(3,3))
> neutre_th2_d4<- list("condition&time",c(2,4),c(4,4))
> th2_th1_d1<- list("condition&time",c(4,3),c(1,1))
> th2_th1_d2<- list("condition&time",c(4,3),c(2,2))
> th2_th1_d3<- list("condition&time",c(4,3),c(3,3))
> th2_th1_d4<- list("condition&time",c(4,3),c(4,4))
```

We proceed to the corresponding gene selections, setting the option "`cont`" to TRUE (because the control is not time-measured) and lfc (minimal log fold change) to 1.5:

```
> S_naive_neutre<-geneSelection(M,naive_neutre,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_th1<-geneSelection(M,naive_th1,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_th2<-geneSelection(M,naive_th2,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th1<-geneSelection(M,neutre_th1,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th2<-geneSelection(M,neutre_th2,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_neutre_d1<-geneSelection(M,naive_neutre_d1,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_neutre_d2<-geneSelection(M,naive_neutre_d2,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_neutre_d3<-geneSelection(M,naive_neutre_d3,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_neutre_d4<-geneSelection(M,naive_neutre_d4,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_th1_d1<-geneSelection(M,naive_th1_d1,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_th1_d2<-geneSelection(M,naive_th1_d2,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_th1_d3<-geneSelection(M,naive_th1_d3,-1,cont=TRUE,alpha=0.05,
```

```
   data_log=FALSE,lfc=1.5)
> S_naive_th1_d4<-geneSelection(M,naive_th1_d4,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_th2_d1<-geneSelection(M,naive_th2_d1,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_th2_d2<-geneSelection(M,naive_th2_d2,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_th2_d3<-geneSelection(M,naive_th2_d3,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_naive_th2_d4<-geneSelection(M,naive_th2_d4,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th1_d1<-geneSelection(M,neutre_th1_d1,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th1_d2<-geneSelection(M,neutre_th1_d2,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th1_d3<-geneSelection(M,neutre_th1_d3,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th1_d4<-geneSelection(M,neutre_th1_d4,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th2_d1<-geneSelection(M,neutre_th2_d1,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th2_d2<-geneSelection(M,neutre_th2_d2,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th2_d3<-geneSelection(M,neutre_th2_d3,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_neutre_th2_d4<-geneSelection(M,neutre_th2_d4,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_th2_th1_d1<-geneSelection(M,th2_th1_d1,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_th2_th1_d2<-geneSelection(M,th2_th1_d2,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_th2_th1_d3<-geneSelection(M,th2_th1_d3,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> S_th2_th1_d4<-geneSelection(M,th2_th1_d4,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
```

We can use the `org.Mm.eg.db` Bioconductor database which is provided as a package (Carlson, 2010) to find gene names:

```
> library(org.Mm.eg.db)
> #Here S is the union of all selections
>
> S<-unionMicro(list(S_naive_neutre,S_naive_th1,
   S_naive_th2,S_neutre_th1,S_neutre_th2,
   S_naive_neutre_d1,S_naive_neutre_d2,S_naive_neutre_d3,S_naive_neutre_d4,
   S_naive_th1_d1,S_naive_th1_d2,S_naive_th1_d3,S_naive_th1_d4,
   S_naive_th2_d1,S_naive_th2_d2,S_naive_th2_d3,S_naive_th2_d4,
   S_neutre_th1_d1,S_neutre_th1_d2,S_neutre_th1_d3,S_neutre_th1_d4,
   S_neutre_th2_d1,S_neutre_th2_d2,S_neutre_th2_d3,S_neutre_th2_d4,
   S_th2_th1_d1,S_th2_th1_d2,S_th2_th1_d3,S_th2_th1_d4
   ))
> ff<-function(x){substr(x, 1, nchar(x)-3)}
> ff<-Vectorize(ff)
```

```
> #Here is the function to transform the probeset names to gene ID.
>
>
>
> probe_to_id<-function(n){
   x <- (org.Mm.egENSEMBL2EG)
   mp<-mappedkeys(x)
   xx <- unlist(as.list(x[mp]))
   genes_all = xx[ff(n)]
   indi2<-!is.na(genes_all)
   genes_all=genes_all[!is.na(genes_all)]
   gns <- unlist(mget((unlist(genes_all)), org.Mm.egSYMBOL) )
   Name<-rep("na",length(n))
   Name[ indi2]<-gns
   return(toupper(Name))
   }
> Name<-probe_to_id(S@name)
```

As expected, we found the differentially expressed genes already highlighted in this paper (van den Ham *et al.*, 2013). Here is the list:

- Batf
- Batf2
- Batf3
- Gata3
- Ifng
- Il13
- Il4
- Tbx21 (Tbet)
- Ccl17
- Ccl24
- Egr2
- Foxp3
- Hopx
- Il10
- Il6
- Il9
- Mecom
- Mycn
- Nfil3
- Rbpj
- Spic
- Stad1
- Stat5a
- Tnf
- Tnfrsf4
- Vdr.

The whole list of genes contains 1.454 elements (see supplemental file genelists.xls available at `http://www-math.u-strasbg.fr/genpred/sites/genpred/IMG/xls/genelists.xls`).

3.3 Selecting differentially expressed genes with specific patterns

To reverse-engineer the TH1 specific network, we select genes that are differentially expressed with the neutral+TH1 stimulation versus the neutral stimulation. Here we use the same methodology than in the paper of Vallat *et al.* (Vallat *et al.*, 2013), that's why we select the following patterns:

- early differentially expressed genes (D1,D2,D1+D2),
- late differentially expressed genes (D3,D4,D3+D4),
- gene with high differential expression (D1+D2+D3+D4).

```
> neutre_th1_early<- list("patterns",c(2,3),
   rbind(c(1,1,0,0),c(1,0,0,0),c(0,1,0,0)))
> neutre_th1_late<- list("patterns",c(2,3),
   rbind(c(0,0,1,1),c(0,0,1,0),c(0,0,0,1)))
> neutre_th1_high<- list("patterns",c(2,3),
```

```
   rbind(c(1,1,1,0),c(1,1,1,1)))
> th1_early<-geneSelection(M,neutre_th1_early,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> th1_late<-geneSelection(M,neutre_th1_late,-1,cont=TRUE,alpha=0.05,
   data_log=FALSE,lfc=1.5)
> th1_high<-geneSelection(M,neutre_th1_high,-1,cont=TRUE,
   alpha=0.05,data_log=FALSE,lfc=1.5)
> dim(th1_early)
[1] 15 12
> dim(th1_late)
[1] 66 12
> dim(th1_high)
[1] 13 12
> selection_th1<-unionMicro(list(th1_early,th1_late,th1_high))
> S<-selection_th1
> plot(selection_th1)
> S_name<-probe_to_id(S@name)
> selection_th1@name<-S_name
```

Interestingly, many genes of interest that are specific to TH1 are in our short selection of 94 genes:

- IFNG
- IL4
- IL9
- GATA3
- FOXP3
- MECOM
- IL13
- HOPX
- MYCN

The Figure 1 shows the selected genes grouped within time-clusters.

3.4 Reverse-engineering the TH1 specific network

Using our selection of 94 genes, we reverse-engineer the network.

```
> network_th1<-inference(selection_th1)
> cutoff(network_th1)
> #To save the plot of the network in a convenient way,
> #we use the Cairo package (Simon Urbanek and Jeffrey Horner, 2012)
>
> library(Cairo)
> Cairo(40, 20, file="th1.pdf",type="pdf",units="cm",bg="white")
> par(mar=c(0,0,0,0))
> pos<-position(network_th1,nv=0.12)
> plot(network_th1,nv=0.12,gr=selection_th1@group,label_v=S_name,ini=pos,
   edge.arrow.size=0.85,edge.thickness=2)
> dev.off()
```

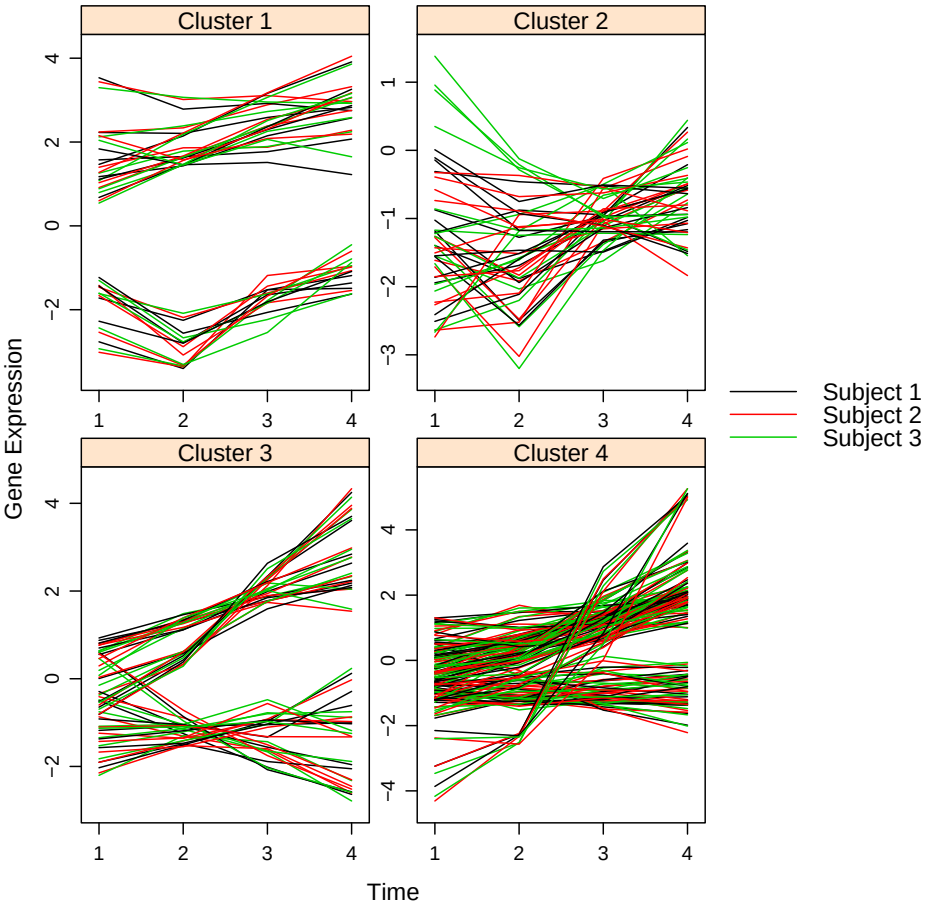

Figure 1: Selected genes for inference of the TH1 specific network

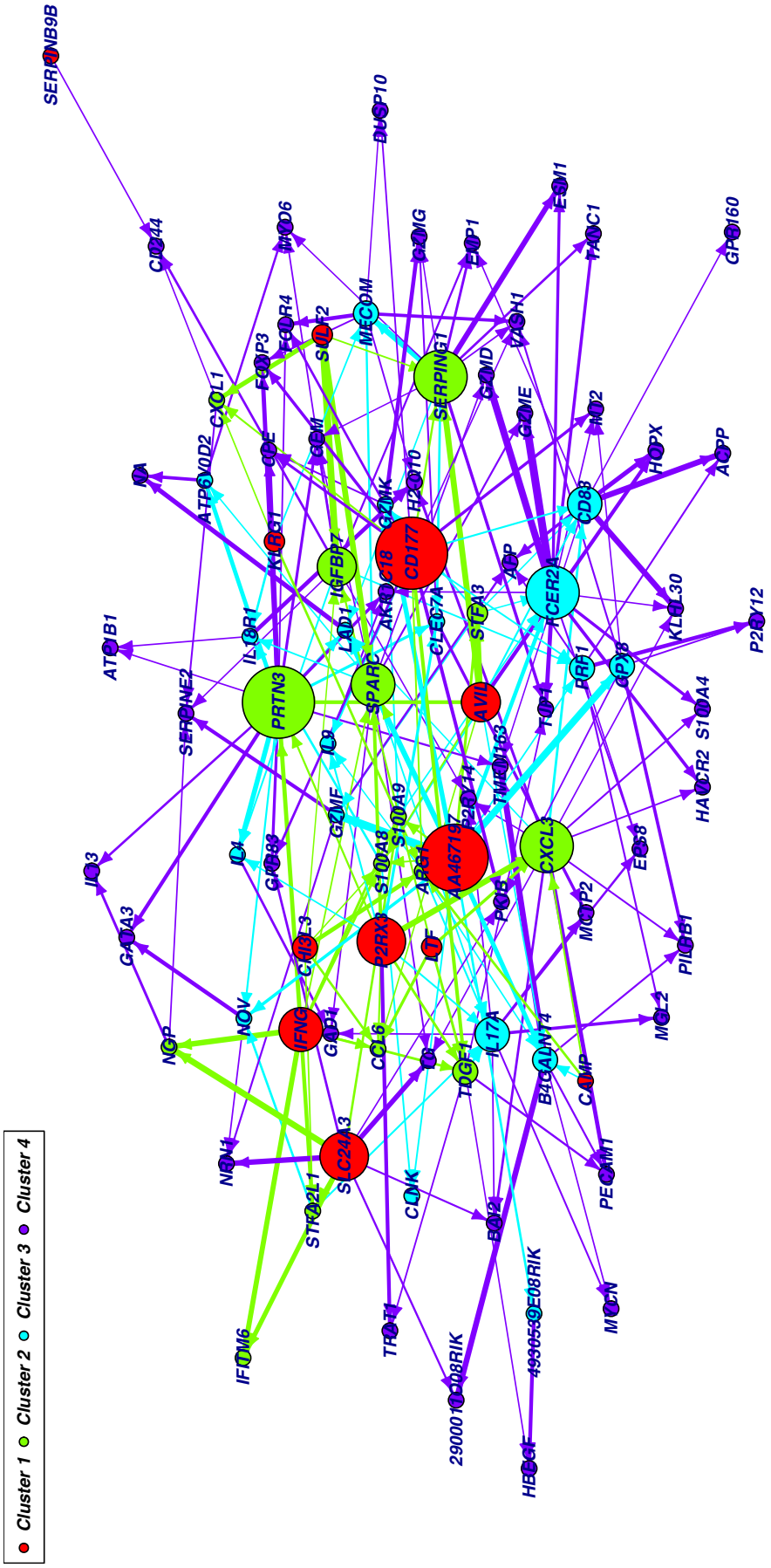

Figure 2: Result of the reverse-engineering of the TH1 specific network.

The TH1 specific network is represented in Figure 2.

We note that the key TH1 gene, IFNG, is a hub in our network. Furthermore, the analyze of the network shows that IFNG is the third most important gene in our network by its closeness score.

```
> D.influence<-analyze_network(network_th1,nv=0.12,network_th1@name)
> head(D.influence[rev(order(D.influence$closeness)),])
```

3.5 Using very specific patterns and lower the log-fold change threshold

Up to now, we decrease the minimal log fold change parameter from 1.5 to 0.5. To enhance the gene selection, we decide to look for very specific patterns. Such a procedure is known to reduce the false-positive rate (Di Camillo *et al.*, 2012).

```
> contrastTH1d1<-  list("patterns",c(1,3),rbind(c(1,0,0,0)))
> contrastTH1d2<-  list("patterns",c(1,3),rbind(c(0,1,0,0)))
> contrastTH1d3<-  list("patterns",c(1,3),rbind(c(0,0,1,0)))
> contrastTH1d4<-  list("patterns",c(1,3),rbind(c(0,0,0,1)))
> S1<-geneSelection(M,contrastTH1d1,-1,cont=TRUE,
   alpha=0.05,data_log=FALSE,lfc=0.5)
> S2<-geneSelection(M,contrastTH1d2,-1,cont=TRUE,
   alpha=0.05,data_log=FALSE,lfc=0.5)
> S3<-geneSelection(M,contrastTH1d3,-1,cont=TRUE,
   alpha=0.05,data_log=FALSE,lfc=0.5)
> S4<-geneSelection(M,contrastTH1d4,-1,cont=TRUE,
   alpha=0.05,data_log=FALSE,lfc=0.5)
> S<-unionMicro(list(S1,S2,S3,S4))
```

As shown in Figure 3, this new selection presents very specific time patterns in which each gene has a peak of differential expression at only one time point.

Additionally, we can analyze the predominant biological gene functions (GO) of the selected genes using two more libraries (geneListPie and org.Mm.eg.db):

```
> library(geneListPie)
> library(org.Mm.eg.db)
> data(goslim.mouse.BP)
> genes_all<-probe_to_id(S@name)
> pdf("functions_TH1.pdf",width=14)
> r1<-geneListProfile(goslim.mouse.BP, genes_all, threshold=10)
> labels<-sub("_", "__", r1$labels)
> labels<-sub(".*__", "", labels)
> barplot(r1$sizes, main="GO Slim Biological Process Mapping  : BP",
   col=rainbow(length(r1$sizes)),names.arg=1:length(r1$sizes),
   ylab="Number of genes")
> legend("topright",legend=paste(paste(1:15,"-"),labels),
   pch=15,col=rainbow(length(r1$sizes)))
> dev.off()
```

We notice that this allows selecting supplemental genes harboring "significant" patterns, enriching the gene selection with transcriptional factors and key genes in the lymphocyte biology (IL15, IL6, IL7R, JAK2, IKZF3 or ITGAL for example) (see Figure 4).

3.6 Reverse-engineering an additional network

For readability, we only use 5% of the selected genes from each previous sub-selection (S1, S2, S3, S4):

```
> S1_res<-geneSelection(M,contrastTH1d1,0.05,cont=TRUE,
   alpha=0.05,data_log=FALSE,lfc=0.5)
> S2_res<-geneSelection(M,contrastTH1d2,0.05,cont=TRUE,
   alpha=0.05,data_log=FALSE,lfc=0.5)
```

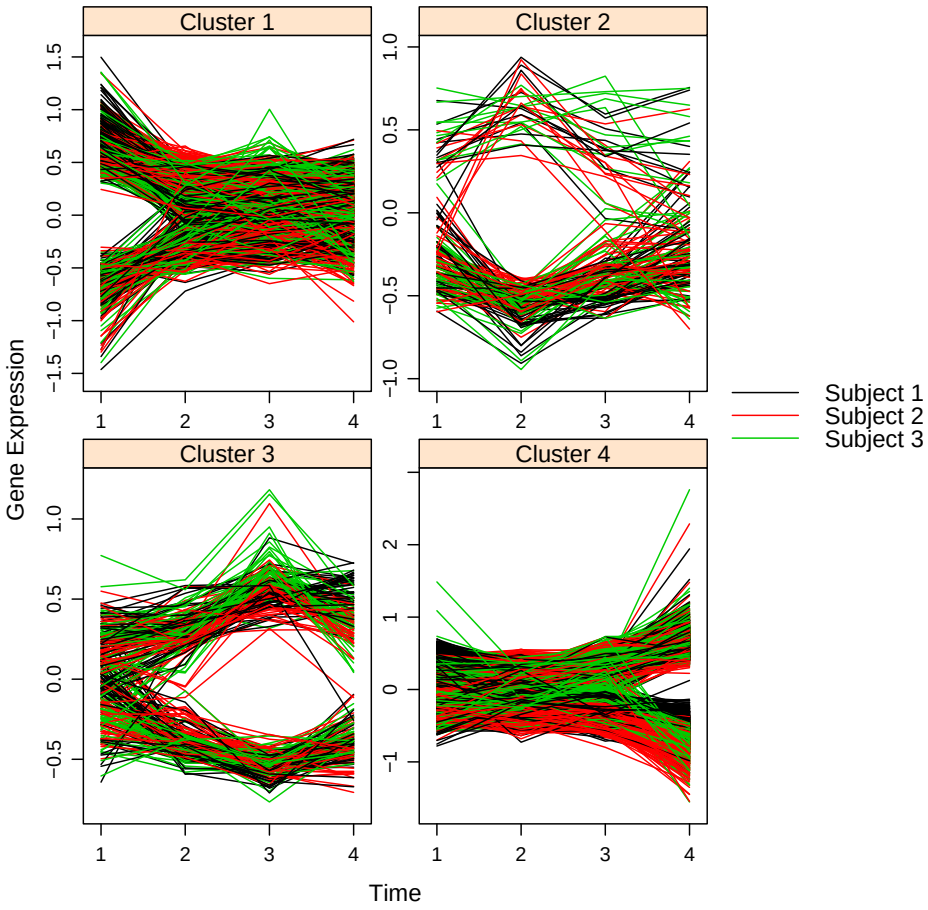

Figure 3: New gene selection with a minimal log fold-change set to 0.5 and very specific temporal patterns.

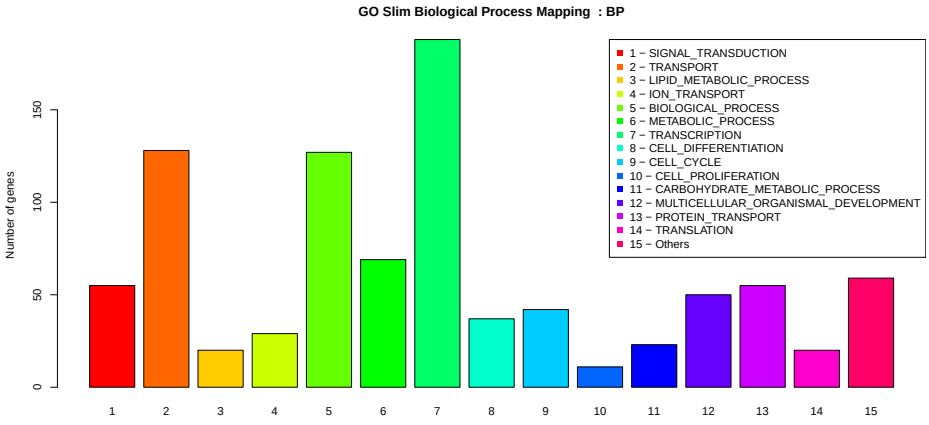

Figure 4: Biological functions of the selected genes.

```
> S3_res<-geneSelection(M,contrastTH1d3,0.05,cont=TRUE,
   alpha=0.05,data_log=FALSE,lfc=0.5)
> S4_res<-geneSelection(M,contrastTH1d4,0.05,cont=TRUE,
   alpha=0.05,data_log=FALSE,lfc=0.5)
> S_res<-unionMicro(list(S1_res,S2_res,S3_res,S4_res))
> network_resTH1<-inference(S_res)
> cutoff(network_resTH1)
```

The resulting network is presented in Figure 5.

```
> Cairo(40, 20, file="th1_res.pdf",type="pdf",units="cm",bg="white")
> par(mar=c(0,0,0,0))
> pos2<-position(network_resTH1,nv=0.12)
> plot(network_resTH1,nv=0.12,gr=S_res@group,label_v=probe_to_id(S_res@name),
   edge.arrow.size=0.73,edge.thickness=2)
> dev.off()
```

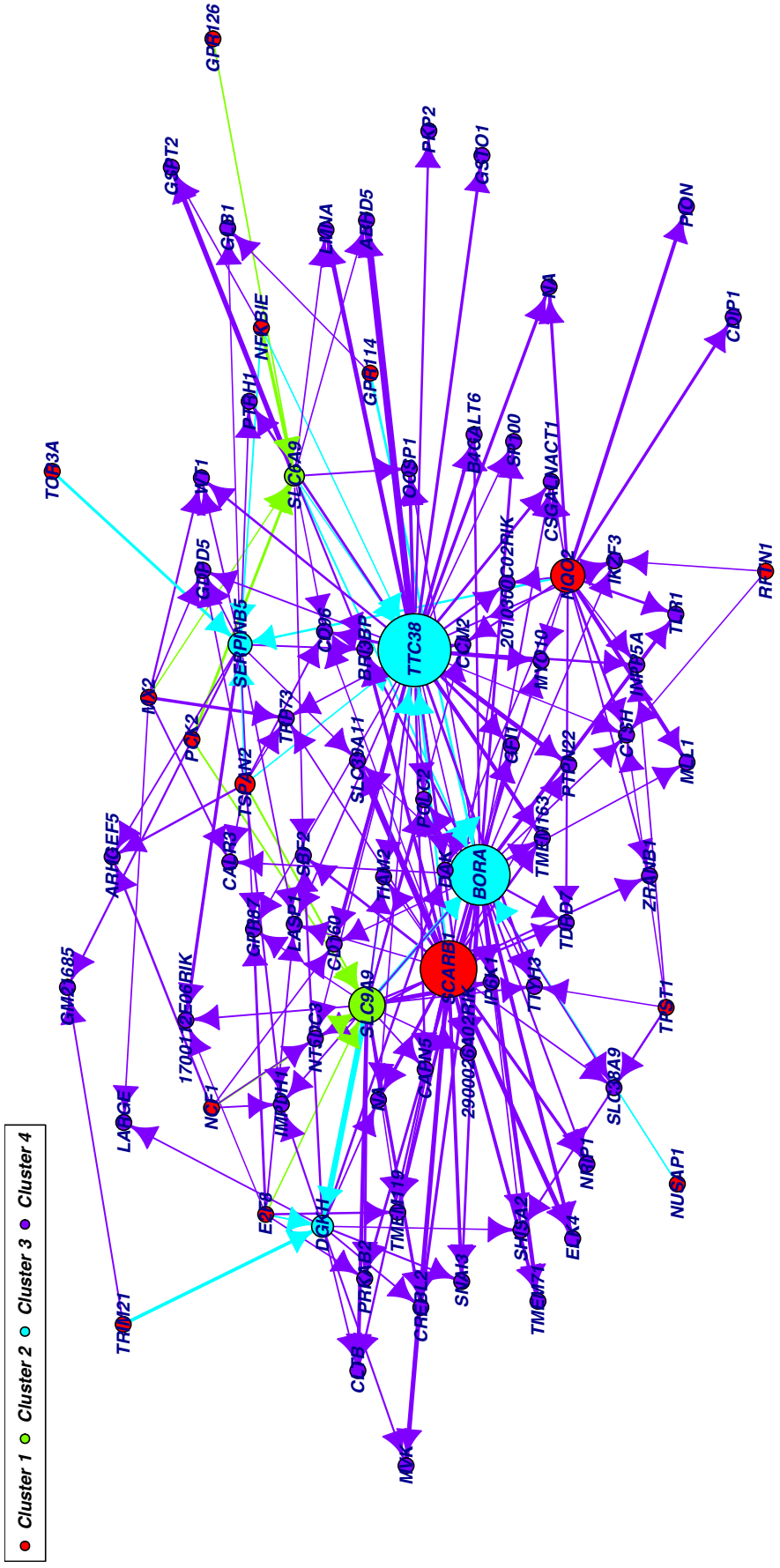

Figure 5: Result of the reverse-engineering of the second network.

4 Prediction of gene expressions after a knock-out experiment

The package further allows predicting the result of a knock-out experiment on the downstream genes in the network. Here we choose to knock-out IFN-G in our first TH1 specific network. The Figure 6 shows the effects of the knock-out experiment on IFN-G.

```
> IFNG<-which(selection_th1@name %in% "IFNG")
> network.p<-predict(selection_th1,network_th1,nv=0.12,targets=IFNG)


> Cairo(40, 20, file="th1_pred.pdf",type="pdf",units="cm",bg="white")
> par(mar=c(0,0,0,0))
> plot(network.p,time=2,label_v=S_name,ini=pos,
   edge.arrow.size=0.85,edge.thickness=2)
> dev.off()
```

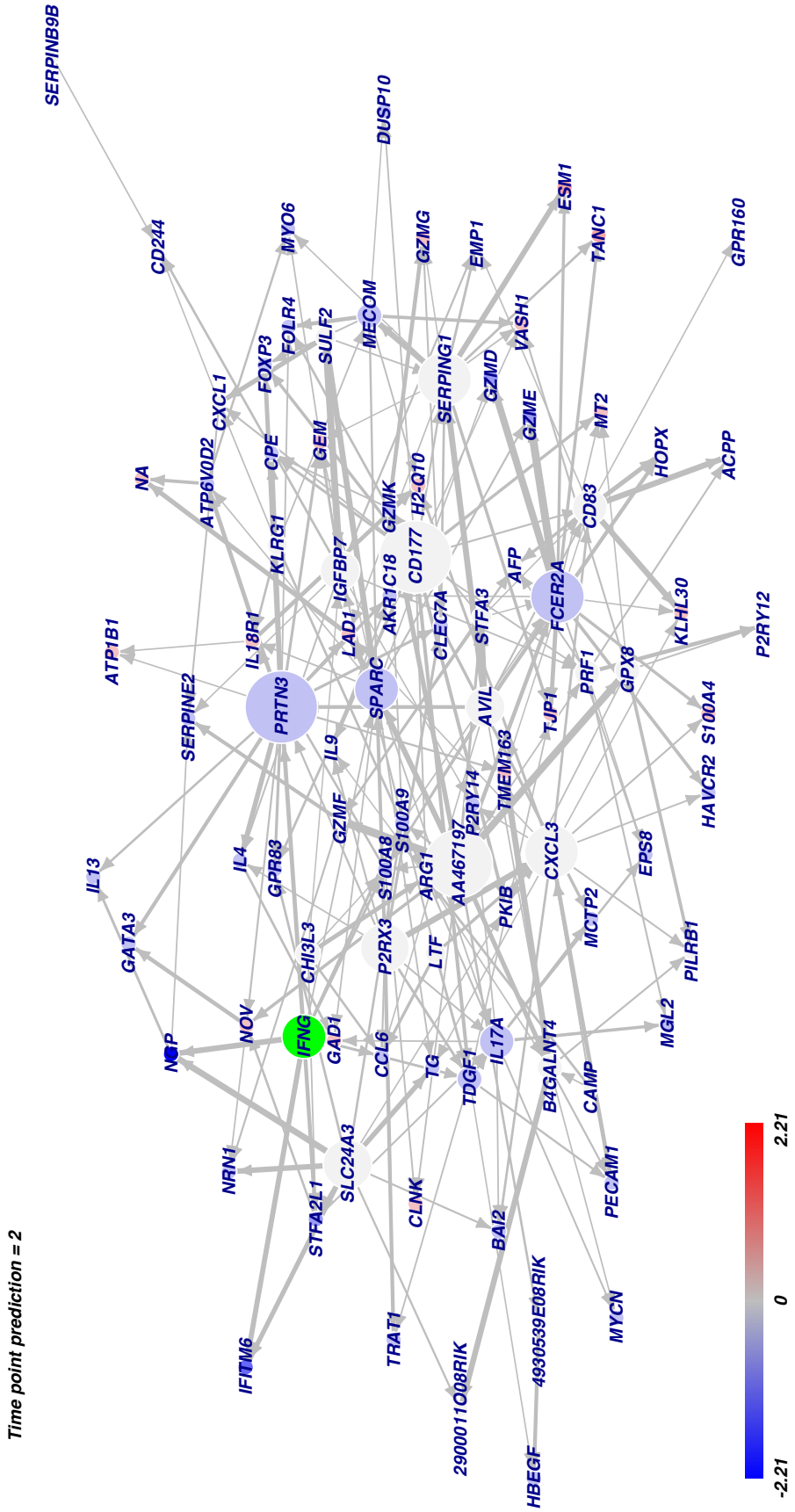

Figure 6: Predicted effects of the knock-out of IFN-G.

References

Carlson, M. (2010). *org.Mm.eg.db: Genome wide annotation for Mouse.* R package version 2.9.0.

Di Camillo, B., Irving, B. A., Schimke, J., Sanavia, T., Toffolo, G., Cobelli, C., and Nair, K. S. (2012). Function-based discovery of significant transcriptional temporal patterns in insulin stimulated muscle cells. *PLoS ONE*, 7(3):e32391.

Urbanek, S. and Horner, J. (2012). *Cairo: R graphics device using cairo graphics library for creating high-quality bitmap (PNG, JPEG, TIFF), vector (PDF, SVG, PostScript) and display (X11 and Win32) output.* R package version 1.5-2.

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.

van den Ham, H.-J., Waal, L., Zaaraoui-Boutahar, F., Bijl, M., IJcken, W. F., Osterhaus, A. D., Boer, R. J., and Andeweg, A. C. (2013). Early divergence of th1 and th2 transcriptomes involves a small core response and sets of transiently expressed genes. *European Journal of Immunology*, 43(4):1074–1084.