predictionet: a package for inferring predictive networks from high-dimensional genomic data

Benjamin Haibe-Kains[1,2], Catharina Olsen[3], Gianluca Bontempi[3], and John Quackenbush[1,2]

[1]Computational Biology and Functional Genomics Laboratory, Dana-Farber Cancer Institute, Harvard School of Public Health
[2]Center for Cancer Computational Biology, Dana-Farber Cancer Institute
[3]Machine Learning Group, Université Libre de Bruxelles

February 16, 2011

Contents

1 Introduction

This document describes an R package for inferring *predictive* networks from high-dimensional genomic data. The goal is to develop a an inference method able to:

- combine prior information retrieved from full-text pubmed articles and structured biological databases, with genomic data (gene expression for example)
- deal with high-dimensional data (hundreds to thousand of variables)
- make predictions
- handle data perturbations, i.e., gene knockdowns or over-expression

2 Getting started

After starting R, the package should be loaded using the following command:

```
> library(predictionet)
```

This will load *predictionet* as well as the dependencies.

The example data named `exp0.colon.ras`, which includes gene expressions (`mydata`), annotations (`myannot`), clinical data (`mydemo`), and priors (`mypriors`), is loaded with the following data command.

```
> data(exp0.colon.ras)
```

In order to illustrate the use of the *predictionet* packge to infer a network, we select the top 10 genes most differentially expressed with respect to KRAS mutation status, as published in [Bild et al., 2006].

```
> ## number of genes to select for the analysis
> genen <- 10
> ## select only the top genes
> goi <- dimnames(annot.ras)[[1]][order(abs(log2(annot.ras[, "fold.change"])),
+     decreasing = TRUE)[1:genen]]
> mydata <- data.ras[, goi, drop = FALSE]
> myannot <- annot.ras[goi, , drop = FALSE]
> mypriors <- priors.ras[goi, goi, drop = FALSE]
> mydemo <- demo.ras
```

The `exp0.colon.ras` dataset is purely observational, the human colon tumors come from a cross-sectional study called exp0[1]. Therefore there is no perturbations in the data (no genes have been artificially knocked down or over-expressed); the matrix of perturbations will then be full of '0'.

```
> ## create matrix of perturbations (no perturbations in our
> #   case)
> mypert <- matrix(0, nrow = nrow(mydata), ncol = ncol(mydata),
+     dimnames = dimnames(mydata))
```

Let's now infer a network from the entire dataset (gene expression data and priors). The maximum number of parents is set to three (`maxparents` = 5), the priors and gene expression data are equally weighted (`priors.weight` = 0.5) and the regression based method is used to infer the network (`method` = "regrnet").

```
> ## infer global network from data and priors
> mynet <- netinf(data = mydata, perturbations = mypert, priors = mypriors,
+     priors.count = TRUE, priors.weight = 0.5, maxparents = 3,
```

[1] `http://www.intgen.org/expo/`

```
+     method = "regrnet", seed = 54321)
```

Once the network is inferred, its topology can be easily displayed using the *network* package. The resulting topology is illustrated in Figure 1.

```
> ## network topology
> mytopoglobal <- net2topo(net = mynet, coefficients = FALSE)
> library(network)
> xnet <- network(x = mytopoglobal, matrix.type = "adjacency",
+     directed = TRUE, loops = FALSE, vertex.attrnames = dimnames(mytopoglobal)[[1]])

> mynetlayout <- plot.network(x = xnet, displayisolates = TRUE,
+     displaylabels = TRUE, boxed.labels = FALSE, label.pos = 0,
+     arrowhead.cex = 1.5, vertex.cex = 3, vertex.col = "royalblue",
+     jitter = FALSE, pad = 0.5)
```

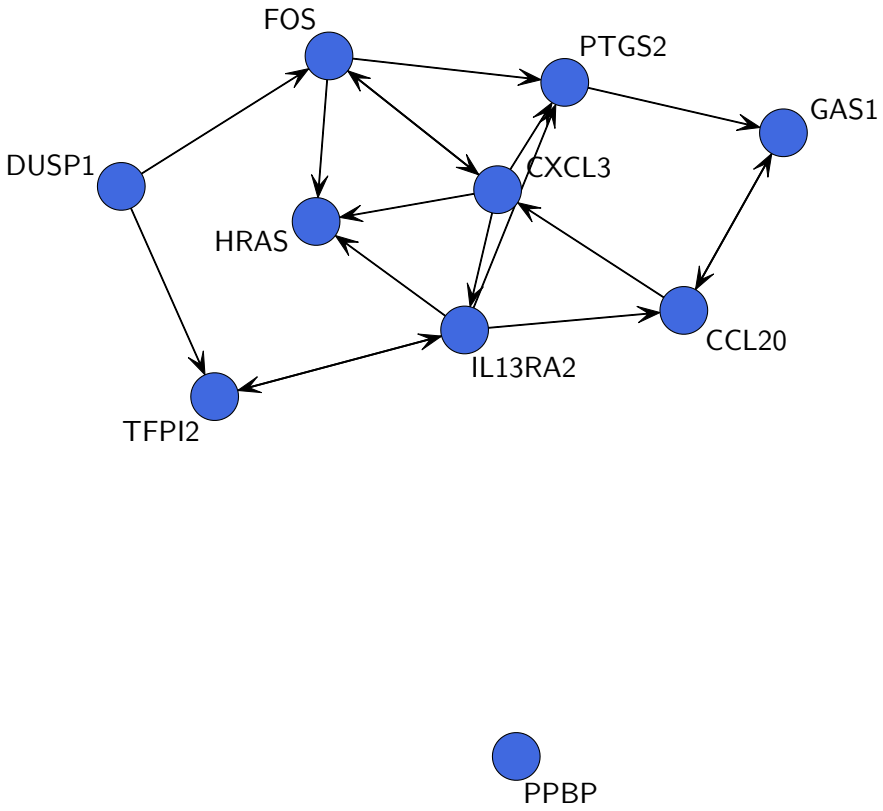

Figure 1: Directed graph representing the topology of the network inferred from priors and gene expression data.

3 Predictive ability

The network that is inferred is able to predict the expression of a gene given the expression of its parents. We argue that interactions with genes accurately predicted in the network are biologically relevant. However, as for all prediction models, such predictive ability should be assess using independent data. In this example, predictive ability will be assessed in 10-fold cross-validation.

Two performance criteria have been implemented to assess the predictive ability of a network, depending on the type of model and prediction. The first performance criterion is a continuous estimate of the predictive ability of each node in the network: the normalized root mean squared error (NMSE) and the loglikelihood for the regression-based and bayesian network inference respectively.The second performance criterion is the classification accuracy, as estimated using the Matthew's correlation coefficient (MCC). Although such an criterion is easier to interpret, it relies on the discretization of the data, what is a non trivial task.

The code below first discretizes the gene expressions into three equally frequent categories, and then performs a cross-validation to get an unbiased estimate of classification accuracy.

```
> ## discretize gene expression data
> ## categories for each variable (same number of categories
> #   for each variable in the current version)
> nbcat <- 3  ## each gene expressions will discretized in 3 equally frequent cat-
egories
> mycats <- NULL
> for (i in 1:ncol(mydata)) {
+     mycats <- c(mycats, list(1:nbcat))
+ }
> names(mycats) <- dimnames(mydata)[[2]]
> cuts.discr <- t(apply(mydata, 2, quantile, probs = seq(0, 1,
+     length.out = nbcat + 1)[-c(1, nbcat + 1)], na.rm = TRUE))
> cuts.discr <- lapply(apply(cuts.discr, 1, list), function(x) {
+     return(x[[1]])
+ })
> ## actual gene expression data
> mydata.discr <- data.discretize(data = mydata, cuts = cuts.discr)
> ## compute folds for cross-validation
> nfold <- 10
> if (nfold == 1) {
+     k <- 1
+     nfold <- nrow(mydata)
+ } else {
+     k <- floor(nrow(mydata)/nfold)
+ }
> ## randomize observations to prevent potential bias in the
> #   cross-validation
> smpl <- sample(nrow(mydata))
> ## perform cross-validation
> mypredscore <- mytopo <- NULL
> for (i in 1:nfold) {
```

```
+     ## fold of cross-validation
+     if (i == nfold) {
+         s.ix <- smpl[c(((i - 1) * k + 1):nrow(mydata))]
+     }
+     else {
+         s.ix <- smpl[c(((i - 1) * k + 1):(i * k))]
+     }
+     ## s.ix contains the indices of the test set
+     ## infer network from training data and priors
+     mynet <- netinf(data = mydata[-s.ix, , drop = FALSE], perturbations = mypert[-
s.ix,
+         , drop = FALSE], priors = mypriors, priors.count = TRUE,
+         priors.weight = 0.5, maxparents = 3, method = "regrnet",
+         seed = 54321)
+     ## compute predictions
+     mynet.pred <- netinf.predict(net = mynet, data = mydata[s.ix,
+         , drop = FALSE], perturbations = mypert[s.ix, , drop = FALSE])
+     ## performance estimation: R2
+     mynet.r2 <- pred.score(data = data[s.ix, , drop = FALSE],
+         pred = mynet.pred, method = "r2")
+     ## performance estimation: NRMSE
+     mynet.nrmse <- pred.score(data = data[s.ix, , drop = FALSE],
+         pred = mynet.pred, method = "nrmse")
+     ## performance estimation: MCC
+     ## discretize predicted gene expression data
+     pred.discr <- data.discretize(data = mynet.pred, cuts = cuts.discr)
+     mynet.mcc <- pred.score(data = mydata.discr[s.ix, , drop = FALSE],
+         pred = pred.discr, categories = categories, method = "mcc")
+     ## adjacency matrix
+     topol <- net2topo(net = mynet, coefficients = FALSE)
+     ## save results
+     myr2 <- rbind(myr2, mynet.r2)
+     mynrmse <- rbind(mynrmse, mynet.nrmse)
+     mymcc <- rbind(mymcc, mynet.mcc)
+     mytopo <- c(mytopo, list(topol))
+ }
> dimnames(myr2)[[1]] <- dimnames(mynrmse)[[1]] <- dimnames(mymcc)[[1]] <- names(mytopo)
<- names(mynets) <- paste("fold",
+     1:nfold, sep = ".")
```

Alternatively, you can use the function `netinf.cv`

```
> netinf.cv(data = mydata, categories = 3, perturbations = mypert,
+     priors = mypriors, maxparents = 3, method = "regrnet", nfold = 10,
+     seed = 54321)
```

The R^2 estimated for each gene in each fold of the cross-validation can be represented in a

boxplot (Figure 2).

```
> par(las = 3)
> boxplot(mypredscore.r2, ylim = c(0, 1), ylab = "R2 in cross-validation")
```

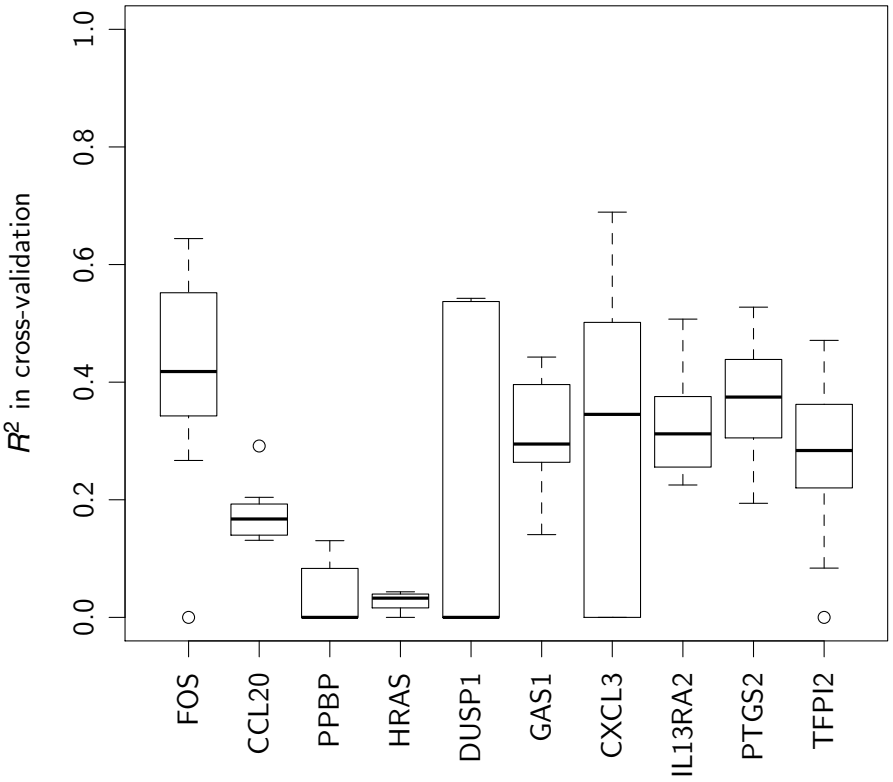

Figure 2: R^2 estimated in 10-fold cross-validation for each gene separately. As can be seen, some of the genes yielded prediction accuracy close to random ($R^2 \approx 0$), the analyst may then want to focus on the genes yielding better prediction accuracy.

The *NRMSE* estimated for each gene in each fold of the cross-validation can be represented in a boxplot (Figure 3).

```
> par(las = 3)
> tt <- 1 - mypredscore.nrmse
> tt[tt < 0] <- 0
> boxplot(tt, ylim = c(0, 1), ylab = "(1 - NRMSE) in cross-validation")
```

The *MCC* estimated for each gene in each fold of the cross-validation can be represented in a boxplot (Figure 4).

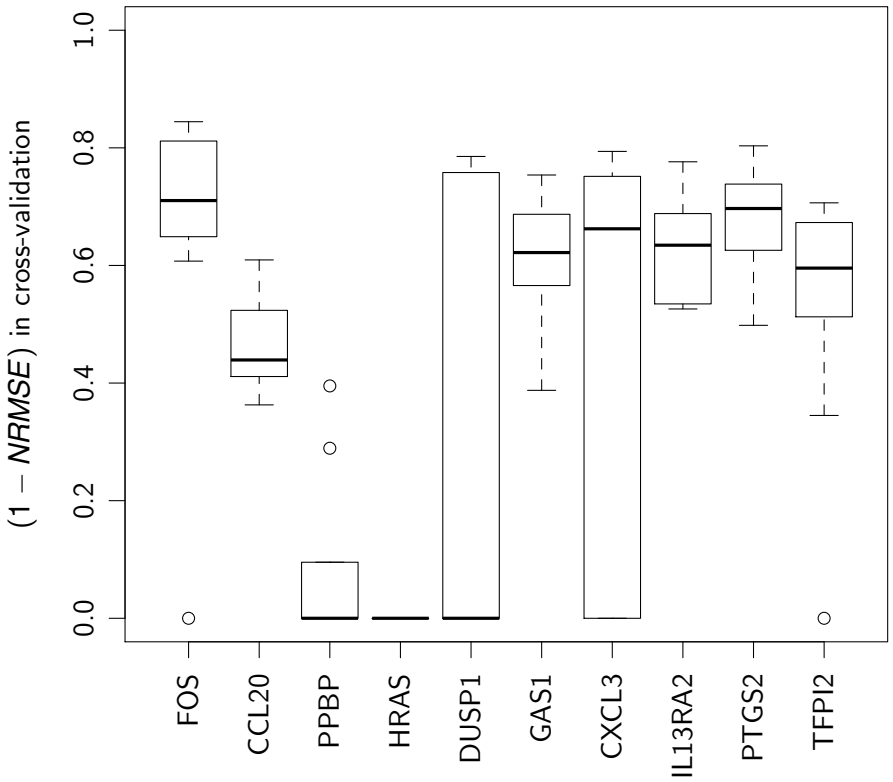

Figure 3: $1 - NRMSE$ estimated in 10-fold cross-validation for each gene separately. As can be seen, some of the genes yielded prediction accuracy close to random ($(1 - NRMSE) \approx 0$), the analyst may then want to focus on the genes yielding better prediction accuracy.

```
> par(las = 3)
> boxplot(mypredscore.mcc, ylim = c(0, 1), ylab = "MCC in cross-validation")
> abline(h = 0.5, lwd = 1, lty = 2, col = "red")
```

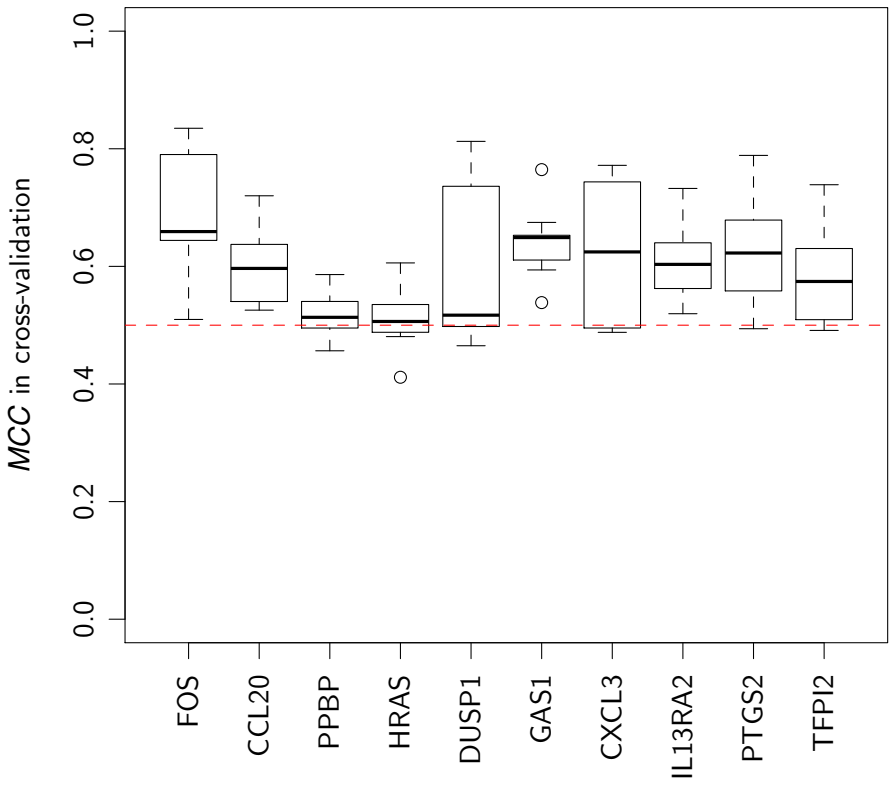

Figure 4: *MCC* estimated in 10-fold cross-validation for each gene separately. As can be seen, some of the genes yielded prediction accuracy close to random ($MCC \approx 0.5$), the analyst may then want to focus on the genes yielding better prediction accuracy.

4 Stability

Stability of network inference – existence of an (un)directed edge, conservation of order relation,... – can be assessed using resampling procedures. The function `net.stab` performs a bootstrap analysis to estimate such stability criteria. Since we already performed a 10-fold cross-validation, we will use these ten resamplings to compute the number of times the directed edges are selected during the network inference.

```
> ## compute stability for each edge
> edgestab <- matrix(0, nrow = ncol(mydata), ncol = ncol(mydata),
+     dimnames = list(dimnames(mydata)[[2]], dimnames(mydata)[[2]]))
> for (i in 1:length(mytopo)) {
+     edgestab <- edgestab + mytopo[[i]]
+ }
> edgestab <- edgestab/length(mytopo)
> ## report stability of edges present in the global network
> edgestab2 <- matrix(0, nrow = ncol(mydata), ncol = ncol(mydata),
+     dimnames = list(dimnames(mydata)[[2]], dimnames(mydata)[[2]]))
> edgestab2[mytopoglobal == 1] <- edgestab[mytopoglobal == 1]
```

The corresponding topology where the edges are colored according to their stability is illustrated in Figure 5.

```
> ## preparing colors
> ii <- 0:10
> mycols <- c("#BEBEBE", rev(rainbow(10, v = 0.8, alpha = 0.5)))
> names(mycols) <- as.character(ii/10)
> myedgecols <- matrix("#00000000", nrow = nrow(mytopoglobal),
+     ncol = ncol(mytopoglobal), dimnames = dimnames(mytopoglobal))
> for (k in 1:length(mycols)) {
+     myedgecols[edgestab2 == names(mycols)[k]] <- mycols[k]
+ }
> myedgecols[mytopoglobal != 1] <- "#00000000"


> layout(rbind(1, 2), heights = rbind(8, 1))
> par(mar = c(1, 1, 1, 1))
> ## network topology
> xnet <- network(x = mytopoglobal, matrix.type = "adjacency",
+     directed = TRUE, loops = FALSE, vertex.attrnames = dimnames(mytopoglobal)[[1]])

> plot.network(x = xnet, displayisolates = TRUE, displaylabels = TRUE,
+     boxed.labels = FALSE, label.pos = 0, arrowhead.cex = 1.5,
+     vertex.cex = 3, vertex.col = "royalblue", jitter = FALSE,
+     pad = 0.5, edge.col = myedgecols, coord = mynetlayout)
> par(mar = c(0, 3, 1, 3))
> plot(ii + 1, ii + 10/6 + 1, bty = "n", type = "n", yaxt = "n",
```

```
+     xaxt = "n", ylab = "", xlab = "", main = "Stability scale",
+     cex.main = 1)
> rect(xleft = ii + 0.5, ybottom = 3, xright = ii + 1.4, ytop = 10 +
+     3, col = mycols, border = "grey")
> text(ii + 1, y = 2.4, labels = names(mycols), pos = 3, cex = 1)
```

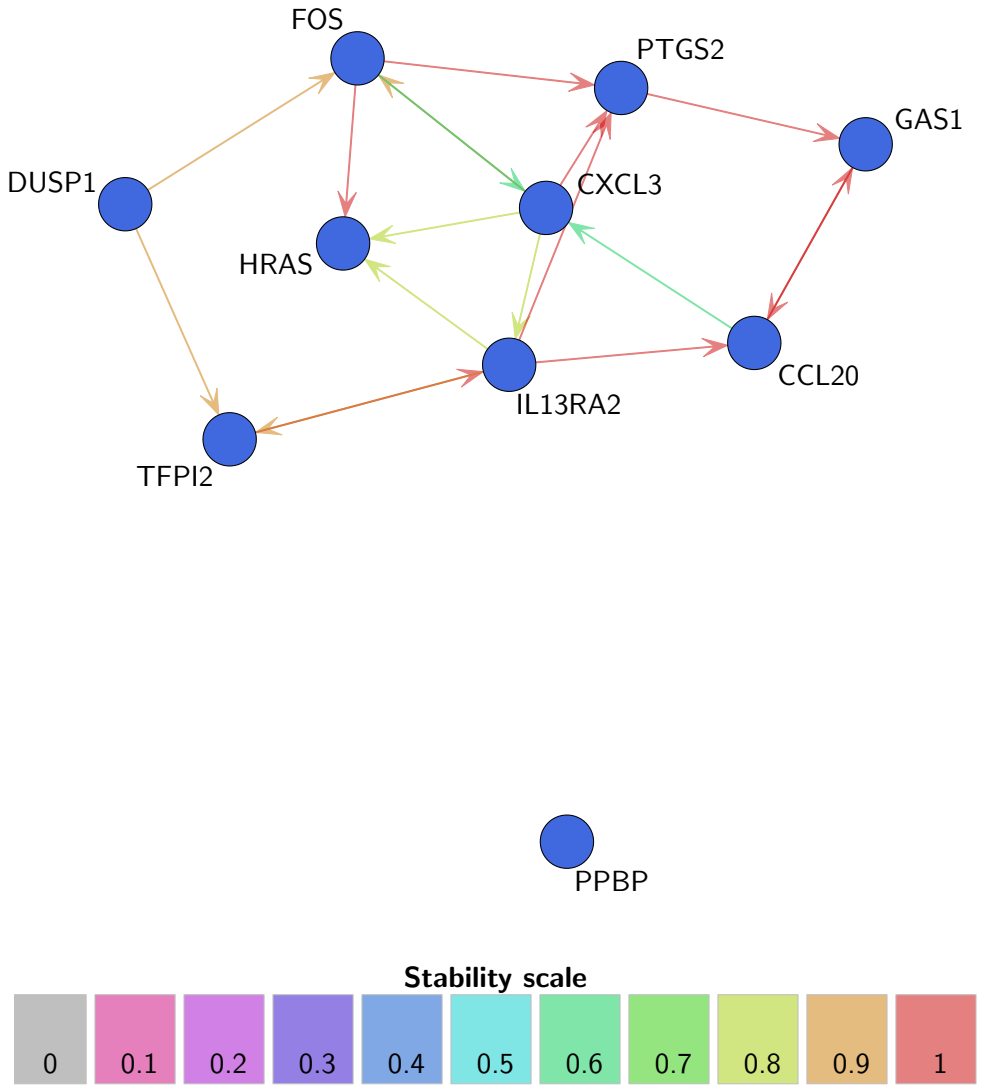

Figure 5: Network topology where the edges are colored according to their stability.

5 Priors

The priors can be retrieved through the *Predictive Networks*[2] web application developed by the Dana-Farber Cancer institute[3] in collaboration with Entagen[4]. Once the list of genes of interest is uploaded in the web application, one can download a csv file with all the interactions reported in the biomedical literature and structured biological databases.

Let's assume that the name of suche a file is priors_ras_from_webapp.csv. One can easily read this file, count the number of times a citation is actually reported and create a matrix containing such counts.

```
> ## read priors generated by the Predictive Networks web
> #   application
> pn.priors <- read.csv("priors_ras_from_webapp.csv", stringsAsFactors = FALSE)

> ## the column names should be: subject, predicate, object,
> #   direction, evidence, sentence, article, source
> ## select only the interactions in which the genes are
> #   comprised in our gene expression dataset
> myx <- is.element(pn.priors[, "subject"], dimnames(mydata)[[2]]) &
+     is.element(pn.priors[, "object"], dimnames(mydata)[[2]])
> pn.priors <- pn.priors[myx, , drop = FALSE]
> ## build prior counts
> priors.count <- matrix(0, nrow = ncol(mydata), ncol = ncol(mydata),
+     dimnames = list(dimnames(mydata)[[2]], dimnames(mydata)[[2]]))
> for (i in 1:nrow(pn.priors)) {
+     switch(tolower(pn.priors[i, "direction"]), right = {
+         priors.count[pn.priors[i, "subject"], pn.priors[i, "object"]] <- pri-
ors.count[pn.priors[i,
+             "subject"], pn.priors[i, "object"]] + ifelse(tolower(pn.priors[i,
+             "evidence"]) == "positive", +1, -1)
+     }, left = {
+         priors.count[pn.priors[i, "object"], pn.priors[i, "subject"]] <- pri-
ors.count[pn.priors[i,
+             "object"], pn.priors[i, "subject"]] + ifelse(tolower(pn.priors[i,
+             "evidence"]) == "positive", +1, -1)
+     }, undirected = {
+         priors.count[pn.priors[i, "subject"], pn.priors[i, "object"]] <- pri-
ors.count[pn.priors[i,
+             "subject"], pn.priors[i, "object"]] + ifelse(tolower(pn.priors[i,
+             "evidence"]) == "positive", +1, -1)
+         priors.count[pn.priors[i, "object"], pn.priors[i, "subject"]] <- pri-
ors.count[pn.priors[i,
+             "object"], pn.priors[i, "subject"]] + ifelse(tolower(pn.priors[i,
+             "evidence"]) == "positive", +1, -1)
```

[2] http://www.predictivenetworks.org
[3] http://www.dfci.org
[4] http://www.entagen.com

```
+     })
+ }
> ## negative count represent evidence for ABSENCE of an
> #   interaction, positive otherwise
```

In order to prevent some very frequently cited interactions to skew the distribution of priors counts, one can truncate the counts to some quantile.

```
> ## quantile of maximum count
> maxcit <- 0.01
> ## truncate the number of frequently cited interactions
> maxcitq <- ceiling(quantile(abs(priors.count[priors.count !=
+     0]), probs = 1 - maxcit))
> if (maxcitq > 0) {
+     priors.count[priors.count > maxcitq] <- maxcitq
+     priors.count[priors.count < -maxcitq] <- -maxcitq
+ }
```

Session Info

- R version 2.12.1 (2010-12-16), `x86_64-apple-darwin9.8.0`
- Base packages: base, datasets, grDevices, graphics, methods, splines, stats, tools, utils
- Other packages: Rcpp 0.9.0, cacheSweave 0.4-5, codetools 0.2-7, filehash 2.1-1, formatR 0.1-9, getopt 1.15, highlight 0.2-5, network 1.6, parser 0.0-13, penalized 0.9-34, pgfSweave 1.1.3, predictionet 0.7, stashR 0.3-3, survival 2.36-2, tikzDevice 0.5.3
- Loaded via a namespace (and not attached): digest 0.4.2

References

A. H. Bild, G. Yao, J. T. Chang, Q. Wang, A. Potti, D. Chasse, M-B. Joshi, >D. Harpole, J. M. Lancaster, A. Berschuk, J. A. Olson Jr, J. R. Marks, H. K. Dressman, M. West, and J. R. Nevins. Oncogenic pathway signatures in human cancers as a guide to targeted therapies. *Nature*, 439: 353–356, 2006.