# PQG short course

## *Predictive Networks*: a new framework for inferring robust networks from gene expression data

Benjamin Haibe-Kains[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

February 22, 2011

## Contents

**1 Introduction** **2**

**2 Target molecular pathway and gene expression data** **3**
2.1 PIK3CA pathway . . . 3
2.2 Breast cancer gene expression data . . . 3

**3 Priors from the biomedical literature and public structured databases** **5**

**4 Network inference from priors and gene expression data** **8**
4.1 Methodology . . . 8
4.2 Run your first network inference with *predictionet* . . . 11
4.3 Edge-specific stability . . . 16
4.4 Gene-specific predictive ability . . . 18

**5 Statistical comparison of network models** **23**
5.1 Comparison of edge-specific stability . . . 24
5.2 Comparison of gene-specific predictive ability . . . 26

**6 Future works** **31**

## Disclaimer

This work was supervised by John Quackenbush and it has been done in collaboration with Christopher Bouton, Erik Bakke, James Hardwick, Walter Gillett, Catharina Olsen, Gianluca Bontempi, Amira Djebbari, Niall Prendergast, and Renee Rubio.

## Materials

All the materials required for this course are available from `http://dvn.iq.harvard.edu/dvn/dv/netinf`.

# 1 Introduction

DNA microarrays and other high-throughput omics technologies provide large datasets that often include hidden patterns of correlation between genes reflecting the complex processes that underlie cellular processes. The challenge in analyzing large-scale expression data has been to extract biologically meaningful inferences regarding these processes – often represented as networks – in an environment where the datasets are complex and noisy. Although many techniques have been developed in an attempt to address these issues, to date their ability to extract meaningful and predictive network relationships has been limited.

In this PQG course[1], I will introduce a platform developed in John Quackenbush's lab, which enables inference of reliable gene interaction networks from prior biological knowledge, in the form of biomedical literature and structured databases, and from gene expression data. Using real data, I will show the benefit of using prior biological knowledge to infer networks and how to quantitatively assess the quality of such networks.

**Tutorial** This tutorial describes in detail all the necessary steps to infer a gene interaction network combining prior biological knowledge and gene expression data.

Our approach is the following:

1. Select a gene expression dataset and a set of genes of interest.
2. Extract priors from the biomedical literature and public structured databases using the *Predictive Networks* web application.
   (a) Read the priors and put them in a form that is convenient for further analyses.
3. Use the *predictionet* R package to infer a gene interaction network from priors and gene expression data.
   (a) Infer a network using the main function `netinf`.
   (b) Explore and display the topology of the resulting network.
   (c) Assess the stability of the network inference in cross-validation (function `netinf.cv`).
   (d) Assess quantitatively the predictive ability of the network in cross-validation (function `netinf.cv`).
   (e) Assess quantitatively the predictive ability of the network in a fully independent dataset (functions `netinf.predict` and `pred.score`).
4. Use *predictionet* to statistically compare multiple gene interaction networks.

Using a breast cancer gene expression dataset as an example (see Section 2) we go step by step and provide all the R code necessary to perform the entire analysis.

[1] `http://www.hsph.harvard.edu/pqg/events/tutorials-short-courses.html`

# 2 Target molecular pathway and gene expression data

For this tutorial, we focus on breast cancer and more specifically the PIK3CA pathway.

## 2.1 PIK3CA pathway

Deregulated PI3K-AKT signaling has been shown to be implicated in many aspects of carcinogenesis. Many genomic alterations have been found to activate this pathway, which can contribute to tumor progression, metastases and resistance to treatment. In particular, somatic mutations of the PIK3CA gene, encoding the p110$\alpha$ catalytic subunit of PI3K, have been shown to activate AKT and induce oncogenic transformation in vitro and in vivo. Mutations of the PIK3CA gene have been found in 18% to 40% of human breast cancers, making them one of the most common genetic aberrations in breast cancer.

Loi et al. identified a list of genes being differentially expressed between breast tumors carrying the PIK3CA mutation in exon 20 and those with the wild-type PIK3CA gene [Loi et al., 2010]. This gene list is provided in `files/loi2010_pik3cags_signature_278.csv` and will serve as the core set of genes involved in the PIK3CA pathway.

## 2.2 Breast cancer gene expression data

We use two large gene expression datasets of primary breast tumors collected before any adjuvant therapy. The first dataset, provided in `files/marray_breast1/minn2007_frma.RData`, was published in [Wang et al., 2005, Minn et al., 2007] and consists of 344 breast tumors hybridized on the Affymetrix GeneChip HG-U133A, composed of 22,283 probesets. The second dataset, provided in `files/marray_breast2/transbig2006affy_frma.RData`, was published in [Desmedt et al., 2007] and consists of 198 breast tumors hybridized on the same Affymetrix GeneChip (HG-U133A). The raw data have been collected from GEO[2], series accession numbers GSE2034/GSE5327 and GSE7390 for the first and the second dataset respectively, and normalized using `frma`[3].

Each `RData` file contains three R objects:

`data` matrix of gene expression data, tumors in rows, probes in columns.

`annot` data frame of probe annotations, probes in rows, annotations in columns.

`demo` data frame of clinical information of the breast cancer patients, patients in rows, clinical variables in columns.

Let's now run R in the working directory. We have to format the datasets for further analysis, using the following commands:

```
> ## read the gene list with the corresponding annotations
> pik3ca.list <- read.csv("files/loi2010_pik3cags_signature_278.csv",
+     stringsAsFactors = FALSE)
> rownames(pik3ca.list) <- pik3ca.list[, "probe"]
> ## remove special characters in the gene symbols
```

[2] `http://www.ncbi.nlm.nih.gov/geo/`

[3] Do you know the benefits of using `frma` over `rma`?

```
> nn <- gsub(pattern = "[-]|[+]|[*]|[%]|[$]|[#]|[{]|[}]|[[]|[]]|[|]|[\\^]",
+     replacement = ".", x = pik3ca.list[, "Gene.symbol"])
> pik3ca.list <- cbind(pik3ca.list, Gene.symbol.curated = toupper(nn))
> ## remove unknown and duplicated genes by respecting the
> #   order of the list
> pik3ca.list <- pik3ca.list[is.na(pik3ca.list[, "Gene.symbol.curated"]) |
+     !duplicated(pik3ca.list[, "Gene.symbol.curated"]), , drop = FALSE]
> ## select only the top genes
> genen <- 20
> if (genen < 2 || genen > nrow(pik3ca.list)) {
+     genen <- nrow(pik3ca.list)
+ }
> pik3ca.list <- pik3ca.list[1:genen, , drop = FALSE]
> ## load the first dataset
> load("files/marray_breast1/minn2007_frma.RData")
> data1 <- data[, rownames(pik3ca.list), drop = FALSE]
> annot1 <- annot[rownames(pik3ca.list), , drop = FALSE]
> colnames(data1) <- rownames(annot1) <- pik3ca.list[, "Gene.symbol.curated"]
> demo1 <- demo
> ## save the data in a CSV file
> write.csv(t(data1), file = "files/marray_breast1_pik3ca.csv")
> ## free memory
> rm(list = c("data", "data.bin", "annot", "demo"))
> gc()
> ## load the second dataset
> load("files/marray_breast2/transbig2006affy_frma.RData")
> data2 <- data[, rownames(pik3ca.list), drop = FALSE]
> annot2 <- annot[rownames(pik3ca.list), , drop = FALSE]
> colnames(data2) <- rownames(annot2) <- pik3ca.list[, "Gene.symbol.curated"]
> demo2 <- demo
> ## save the data in a CSV file
> write.csv(t(data2), file = "files/marray_breast2_pik3ca.csv")
> ## free memory
> rm(list = c("data", "data.bin", "annot", "demo"))
> gc()
```

So the objects `data1`, `annot1`, `demo1` and `data2`, `annot2`, `demo2` contain the information for the first and second dataset respectively, the probe names being replaced by their corresponding gene symbols. The object `pik3ca.list` contains the annotations of the unique genes related to PIK3CA.

# 3 Priors from the biomedical literature and public structured databases

In order to extract previously published gene interactions, Prof John Quackenbush initiated the development of the *Predictive Networks* web application [`https://compbio.dfci.harvard.edu/predictivenetworks/`] implemented by Dr Christopher Bouton and his team at Entagen[4]. This tool enables users to easily retrieve a large number of high quality gene interactions reported in the biomedical literature (full-text open-access PubMed articles and/or MEDLINE abstracts) and/or structured biological databases (e.g., Pathway Commons) by focussing on a core set of genes.

In our case we use the *Predictive Networks* web application to retrieve a list of interactions involving at least one gene included in our list of PIK3CA-related genes.To do so we first have to create an account to login in the webapp. Then go to "My Page" and create a new gene list by cutting-and-pasting the gene symbols[5] included in `files/loi2010_pik3cags_signature_278.csv`. Lastly, we have to export all the resulting triples (e.g., "$gene_a$ regulates $gene_b$") in a CSV file by clicking on "View Triples" and then "Download w/ Sentences As". Name this CSV file as `priors_pik3ca.csv` in the `files/` directory.

Let's now read the newly generated priors into our R session.

```
> ## read priors generated by the Predictive Networks web
> #   application
> pn.priors <- read.csv("files/priors_pik3ca.csv", stringsAsFactors = FALSE)
> ## the column names should be: subject, predicate, object,
> #   direction, evidence, sentence, article, network
> ## remove special characters in the gene symbols
> pn.priors[, "subject"] <- gsub(pattern = "[-]|[+]|[*]|[%]|[$]|[#]|[{]|[}]|[[]|[]]|[|]|[\
+     replacement = ".", x = pn.priors[, "subject"])
> pn.priors[, "object"] <- gsub(pattern = "[-]|[+]|[*]|[%]|[$]|[#]|[{]|[}]|[[]|[]]|[|]|[\\
+     replacement = ".", x = pn.priors[, "object"])
> ## missing values
> pn.priors[!is.na(pn.priors) & (pn.priors == "" | pn.priors ==
+     " " | pn.priors == "N/A")] <- NA
> ## select only the interactions in which the genes are
> #   comprised in our gene expression dataset
> myx <- is.element(pn.priors[, "subject"], pik3ca.list[, "Gene.symbol.curated"])
&
+     is.element(pn.priors[, "object"], pik3ca.list[, "Gene.symbol.curated"])
> pn.priors <- pn.priors[myx, , drop = FALSE]
```

The R object `pn.priors` is a matrix that contains the triples $gene_a \rightarrow gene_b$ that have been reported in the literature and the structured biological databases.

```
> ## print the first rows of 'pn.priors'
> print(head(pn.priors))
```

[4] http://www.entagen.com

[5] Be extremely careful if you use Microsoft Excel since it will automatically interpret SEPT6, that is the gene "SEPTIN 6", as the 6th of September!

```
      subject              predicate  object direction evidence
108       MYC          expression by     MYC      left positive
2246    FGFR2 functional_interaction  PIK3R3     right positive
3040     IRS2 functional_interaction  PIK3R3     right positive
4320    RPL14 functional_interaction   RPL38     right positive
4466  SCGB1D2 functional_interaction SCGB2A2     right positive
10137   FGFR2            reacts with   FGFR2     right positive

108   Furthermore, Knockdown of MYC expression by MYC-specific siRNA significantly inhibit
2246
3040
4320
4466
10137

             article                network
108   PubMed:18809243      Medline Abstracts
2246             <NA> Functional Interaction
3040             <NA> Functional Interaction
4320             <NA> Functional Interaction
4466             <NA> Functional Interaction
10137            <NA>        Pathway Commons
```

As we will see later it is convenient to transform this long list of triples in a square matrix of counts containing the number of times an interaction $gene_a \rightarrow gene_b$ has been reported in the literature and the structured biological databases.

```
> ## build prior counts
> priors.count <- matrix(0, nrow = nrow(pik3ca.list), ncol = nrow(pik3ca.list),
+     dimnames = list(pik3ca.list[, "Gene.symbol.curated"], pik3ca.list[,
+         "Gene.symbol.curated"]))
> 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(!is.na(pn.priors[i,
+             "evidence"]), ifelse(tolower(pn.priors[i, "evidence"]) ==
+             "positive", +1, -1), 0)
+     }, left = {
+         priors.count[pn.priors[i, "object"], pn.priors[i, "subject"]] <- pri-
ors.count[pn.priors[i,
+             "object"], pn.priors[i, "subject"]] + ifelse(!is.na(pn.priors[i,
+             "evidence"]), ifelse(tolower(pn.priors[i, "evidence"]) ==
+             "positive", +1, -1), 0)
+     }, {
+         priors.count[pn.priors[i, "subject"], pn.priors[i, "object"]] <- pri-
ors.count[pn.priors[i,
+             "subject"], pn.priors[i, "object"]] + ifelse(!is.na(pn.priors[i,
```

```
+             "evidence"]), ifelse(tolower(pn.priors[i, "evidence"]) ==
+             "positive", +1, -1), 0)
+         if (pn.priors[i, "object"] != pn.priors[i, "subject"]) {
+             priors.count[pn.priors[i, "object"], pn.priors[i,
+                 "subject"]] <- priors.count[pn.priors[i, "object"],
+                 pn.priors[i, "subject"]] + ifelse(!is.na(pn.priors[i,
+                 "evidence"]), ifelse(tolower(pn.priors[i, "evidence"]) ==
+                 "positive", +1, -1), 0)
+         }
+     })
+ }
> ## negative count represent evidence for ABSENCE of an
> #   interaction, positive otherwise
```

As can be seen, few interactions have been previously reported; this is due to the fact that, to date, the genes related to PIK3CA published in [Loi et al., 2010] have been only poorly characterized.

```
> print(table(priors.count))
```

```
priors.count
  0   1   2   4   5
380  13   4   1   2
```

Out of the 400 possible interactions, only 20 have been reported at least once.

# 4 Network inference from priors and gene expression data

Many methods have been developed for network inference: Bayesian networks, nested $q$-partial graphs, information-theoretic networks, regression-based networks,... These methods differ greatly in their way of dealing with the potentially high dimensionality of the problem, handling missing values, learning from observations and/or interventions, combining genomic data and prior knowledge, assessing the quality of fitted networks, and their ability to make predictions.

In this course, I present a regression-based network inference approach we specifically developed

- to deal efficiently with large number of genes (potentially $\geq 100$),
- to avoid discretization of the input values,
- to combine prior knowledge and gene expression data,
- to infer stable gene interaction networks,
- to make predictions about the expression of genes of interest in independent data.

## 4.1 Methodology

The design of our regression-based approach for network inference is illustrated in Figure 1. The extraction of prior knowledge about gene interactions and its transformation to obtain a directed graph have been explained in Section 3. In order to infer a directed graph from gene expression data we first build an undirected graph using the *maximum relevance minimum redundancy* (MRMR) feature selection technique [Ding and Peng, 2005, Meyer et al., 2007] and we then infer causality [Cheng et al., 2002, Olsen et al., 2009]. The network topology is determined by combining the directed graphs inferred from priors and gene expression data. Using the resulting topology, we can fit regression models in order to assess the predictive ability of the network model.

### MRMR

To infer an undirected graph from gene expression data we use the maximum relevance minimum redundancy (MRMR) feature selection technique [Ding and Peng, 2005, Meyer et al., 2007]. For this iterative forward selection technique, at each step a different gene is used as the target gene. The algorithm selects then the genes that exhibits the highest difference between mutual information with the target gene and redundancy with the previously selected genes. This difference acts as a score for the possible interactions. The interactions which are represented by a score lower than a given threshold will be deleted in order to infer the undirected graph. The idea is that direct interactions should receive a high score whereas indirect interactions a low one.

In the first step, the gene $X_i$ which has the highest mutual information to the target gene $Y$ is selected. The second selected variable $X_j$ will be the one with a high information $I(X_j, Y)$ to the target and at the same time a low information $I(X_j, X_i)$ to the previously selected variable.

In the next steps, given a set $X_{\mathcal{S}}$ of selected variables, the criterion updates $X_{\mathcal{S}}$ by choosing the next gene $X_q$ that maximizes the score

$$s_q = I(X_q, Y) - \frac{1}{|\mathcal{S}|} \sum_{X_k \in X_{\mathcal{S}}} I(X_q, X_k)$$

which can be described as a relevance term minus a redundancy term.

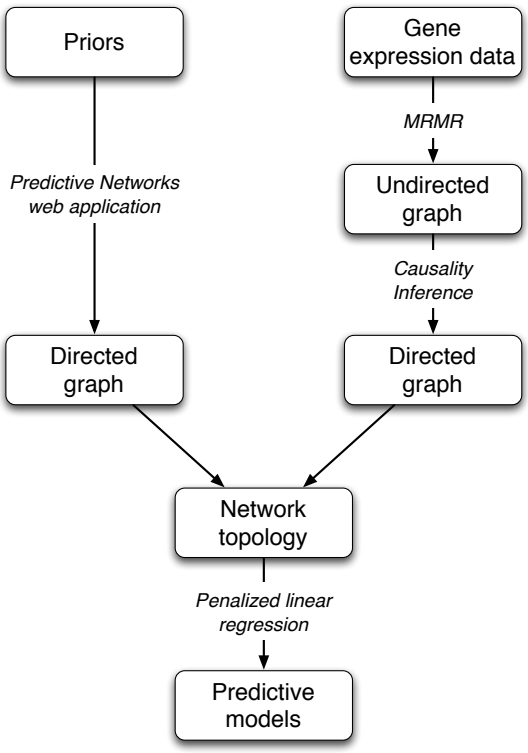


Figure 1: Design of the regression-based network inference approach.

The feature selection stop when the number of genes connected to the target is the maximum number of parents allowed by the user. The resulting undirected graph is locally acyclic since the target gene cannot be selected.

### Causality inference

Once the undirected graph is built, we have to perform the arc orientation. Based on previous works on causality (see [neo, 2003] for a review), we can actually infer causality by using the "explaining away effect" which states that a given common effect creates a dependency between two variables (this can be graphically represented by the *v*-structure, where the value of the collider is given). In [Ding and Peng, 2005, Meyer, 2008] this effects has been related to the property

$$I(X, Z|Y) > I(X, Z) \Leftrightarrow C(X, Y, Z) < 0$$

where $I(X, Z)$ is the mutual information of $X$ and $Z$, $I(X, Z|Y)$ is the mutual information of $X$ and $Z$ given $Y$, and $C(X, Y, Z)$ is the interaction information of $X$, $Y$ and $Z$ such that

$$\begin{aligned} C(X, Y, Z) &= I(x, Y) - I(X, Y|Z) \\ &= I(X, Z) - I(X, Z|Y) \\ &= I(Y, Z) - I(Y, Z|X) \end{aligned}$$

If the undirected (acyclic) graph is given, then for every triple of variables $X - Y - Z$ the property (5.6) corresponds to a v-structure $X \rightarrow Y \leftarrow Z$. On the contrary, knowing only that the

interaction information is less than zero does not help with inferring the network because the three possible differences between mutual information and conditional mutual are equivalent. Thus, it is not evident which variable should be the collider.

Therefore, the only possibility for orienting the arcs using the interaction information is to use the following property:

> Given three variables $X$, $Y$ and $Z$ and a structure $X - Y - Z$, then if $I(X, Z|Y) > I(X, Z)$, i.e., $C(X, Y, Z) < 0$, then $X \rightarrow Y \leftarrow Z$

Depending on the data, we can now identify some of the genes as parents.

#### Combination with priors

To infer the final network topology, we combine, for each interaction between $gene_i$ and $gene_j$, the score based on priors ($P_{ij}$) and MRMR+causality inference ($M_{ij}$). We let users control the weight put on the priors, i.e., their confidence on the prior knowledge gathered on internet:

$$wPij + (1 - w)M_{ij}$$

where $w \in [0, 1]$.

Users can then easily infer networks using prior knowledge only ($w = 1$) or using gene expression data only ($w = 0$). However, we will see in our real case study that a combination of priors and gene expression data usually yields inference of better networks.

#### Predictive ability of the network model

The last step of our network inference approach is the fitting of local regression models to enable prediction of gene expressions. To do so, we fit a penalized linear regression model [Park and Hastie, 2007] for each gene in networks using its parents as predictors.

This novel approach is implemented in the *predictionet* R package and is also integrated in the *Predictive Networks* web application (see "Analysis" panel). Even if the package is not yet available from BioConductor[6], a public Git repository is accessible from `https://github.com/bhaibeka/predictionet`; any non-violent and constructive feedback is welcome. ☺

First we should install the *predictionet* package using the following command

```
> install.packages(pkgs = "predictionet_0.7.tar.gz", repos = NULL)
```

If some dependencies are missing, such as the *penalized* package, you can install them using the `install.packages` and specify the Comprehensive R Archive Network (CRAN) as repository

```
> install.packages("penalized", repos = "http://cran.r-project.org",
+     dependencies = TRUE)
```

If the package has to be installed on a Windows machine, the compilation may fail because you lack of a proper C compiler. Then we should use the R GUI and install the file `predictionet_0.7.zip` that was precompiled specifically for Windows.

[6] `http://www.bioconductor.org`

Let's load the library into our R session.

```
> library(predictionet)
```

We are now able to run all the functions implemented in the *predictionet* package. These function are listed in the ocumentation of the package itself.

```
> library(help = predictionet)
```

## 4.2 Run your first network inference with predictionet

Let's infer our first network and go through the main options of the package. The main function of the *predictionet* package is `netinf` with the following parameters: .

**data** Matrix of continuous or categorical values with observations in rows and features in columns.

**categories** If this parameter is missing, `data` should be already discretized; otherwise the parameter should be either a single integer or a vector of integers specifying the number of categories used to discretize each variable (data are then discretized using equal-frequency bins) or a list of cutoffs to use in order to discretize each of the variables in `data` matrix. If method='bayesnet', this parameter should be specified by the user.

**perturbations** Matrix of $\{0, 1\}$ specifying whether a gene has been perturbed (e.g., knockdown, over-expression) in some experiments. Dimensions should be the same than `data`.

**priors** Matrix of prior information available for gene-gene interaction (parents in rows, children in columns). Values may be probabilities or any other weights (citations count for instance).

**predn** Indices or names of variables to fit during network inference. If missing, all the variables will be used for network inference.

**priors.count** `TRUE` if priors specified by the user are number of citations (count) for each interaction, `FALSE` if probabilities or any other weight lying in $[0, 1]$ are reported instead.

**priors.weight** Real value in [0, 1] specifying the weight to put on the priors (0=only the data are used, 1=only the priors are used to infer the topology of the network).

**maxparents** Maximum number of parents allowed for each gene.

**subset** Vector of indices to select only subset of the observations.

**method** 'bayesnet' for Bayesian network inference with the `catnet` package (not implemented yet), 'regrnet' for regression-based network inference.

**causal** `TRUE` if the causality should be inferred from the data, `FALSE` otherwise

**seed** Set the seed to make the network inference deterministic.

You can easily access this description by consulting the help page of the `netinf`

```
> help(netinf)
```

You can infer a gene interaction network using the entire dataset `data1`, by equally balancing the importance of priors and data in the network inference process (`priors.weight` = 0.5).

```
> mynet <- netinf(data = data1, priors = priors.count, priors.count = TRUE,
+     priors.weight = 0.5, maxparents = 3, method = "regrnet",
+     seed = 54321)
```

NB: You may get some warning messages, this is due to the *penalized* package which is used internally; it will be fixed in a next release.

Now let's display the topology of the network we just inferred (see Figure 2) using the `plot` function of the *network* package.

```
> ## 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 = 2, vertex.col = "royalblue",
+     jitter = FALSE, pad = 0.5)
```

Unfortunately, such kind of plots are not interactive and we cannot change the position of the genes as we would like. There exist other packages to display networks in R, some of them are interactive: *igraph*, *Graphviz*,...

A quick look at the topology in Figure 2 allows us to identify PIK3R3 and RPL14 as highly connected genes (also referred to as *hub* in the literature). By searching for these two genes in GeneSigDB[7], a manually curated database of published gene signatures developed by Dr Aedin Culhane, we can see that PIK3R3 and RPL14, in addition to be included in the PIK3CA signature published in [Loi et al., 2010], are also part of other signatures published in breast, leukemia, ovarian and bladder cancers.

We could also highlight the interactions that were already known (**blue**), the new interactions supported by the gene expression (**green**) and the interactions both known and supported by the data (**red**), see Figure 3.

```
> ## preparing colors
> mycols <- c("blue", "green", "red")
> names(mycols) <- c("prior", "data", "both")
> myedgecols <- matrix("white", nrow = nrow(mytopoglobal), ncol = ncol(mytopoglobal),
```

[7] `http://compbio.dfci.harvard.edu/genesigdb/`

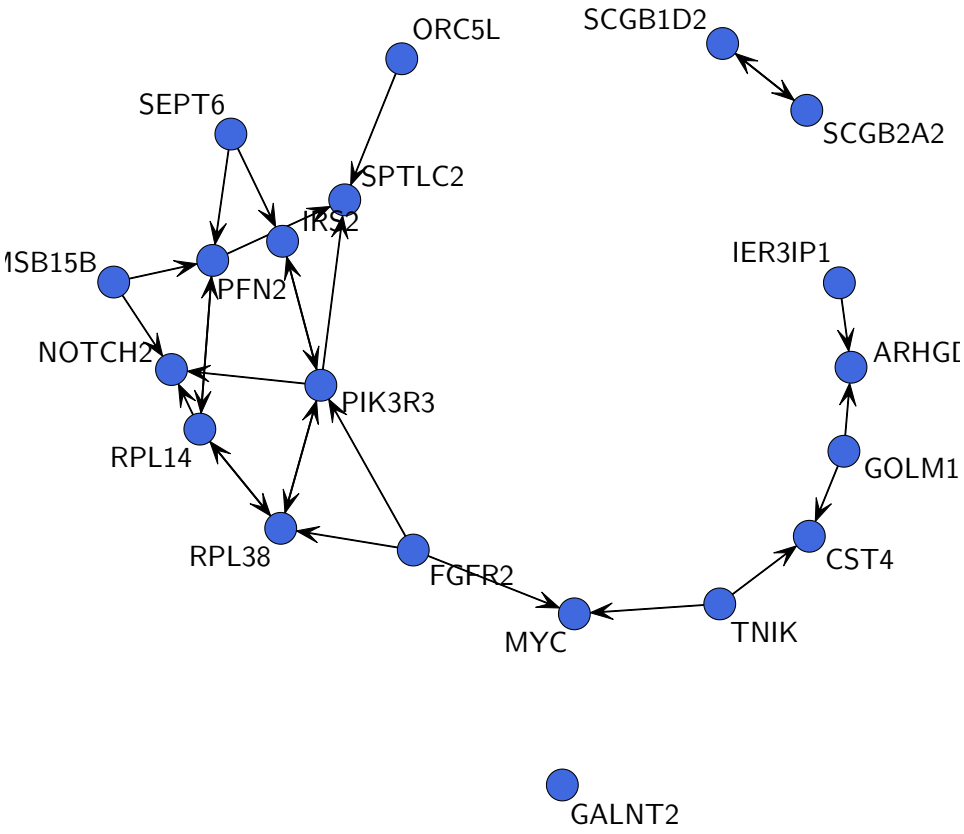


Figure 2: Directed graph representing the topology of the network inferred from priors and gene expression data.

```
+     dimnames = dimnames(mytopoglobal))
> ## prior only
> myedgecols[mytopoglobal == 0 & priors.count >= 1] <- mycols["prior"]
> ## data only
> myedgecols[mytopoglobal == 1 & priors.count < 1] <- mycols["data"]
> ## both in priors and data
> myedgecols[mytopoglobal == 1 & priors.count >= 1] <- mycols["both"]


> mytopoglobal2 <- (myedgecols != "white") + 0
```

```
> ## network topology
> xnet2 <- network(x = mytopoglobal2, matrix.type = "adjacency",
+     directed = TRUE, loops = TRUE, vertex.attrnames = dimnames(mytopoglobal2)[[1]])

> plot.network(x = xnet2, displayisolates = TRUE, displaylabels = TRUE,
+     boxed.labels = FALSE, label.pos = 0, arrowhead.cex = 1.5,
+     vertex.cex = 2, vertex.col = "royalblue", jitter = FALSE,
+     pad = 0.5, edge.col = myedgecols, coord = mynetlayout)
```

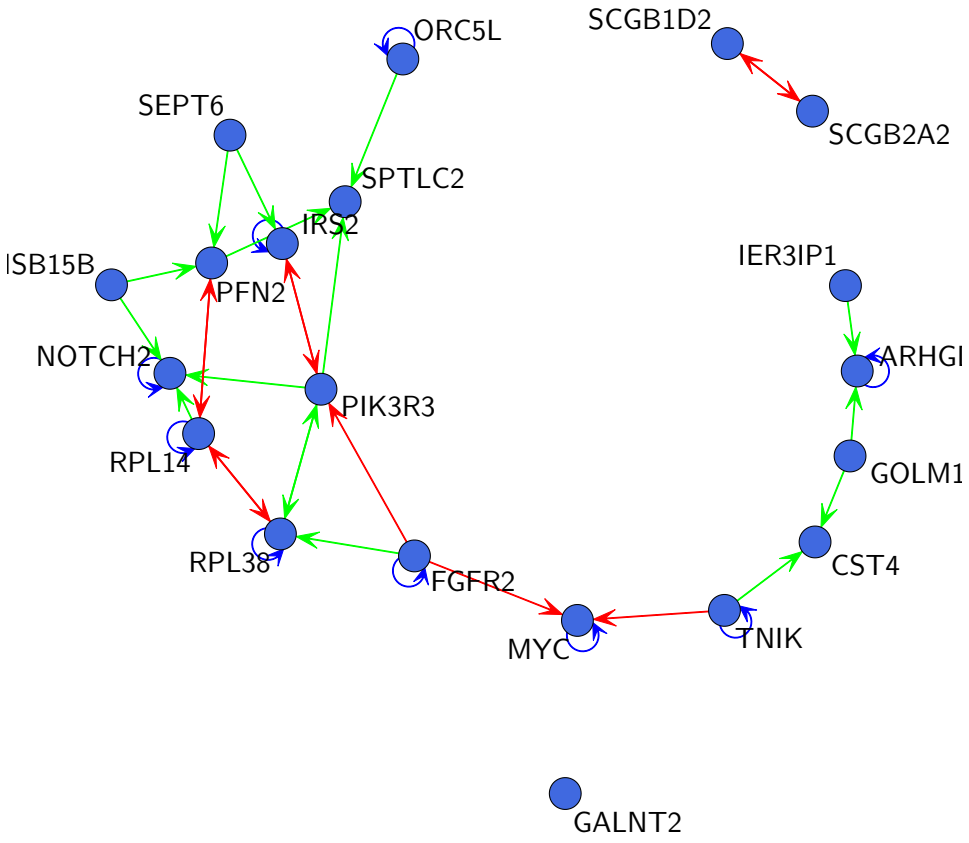


Figure 3: Network topology where the edges are colored according to their source of evidence: **blue** for edges supported by the priors only, **green** for edges supported by the data only, and **red** for edges supported by both the priors and the data.

In Figure 3 we can see that most of the interactions included in the priors have been also supported by the data (**red**) and have been inferred in the final network; only the self loops (**blue**), which are not allowed by our regression-based network inference method, have been discarded. Many new interactions (**green**) have been inferred from the gene expression data.

Although of interest, the topology does not tell us much about the quality of the network inference. Therefore we implemented two statistics to help us focus on the gene interactions that are well supported by the data:

- edge-specific stability.
- gene-specific predictive ability.

The idea is to use a cross-validation procedure[8] to infer multiple networks from different training datasets to assess both the stability of the inference and its predictive ability. The function `netinf.cv`, although computationally intensive, is doing all the work for us. The vast majority of the parameters are the same than for the `netinf` function, with some additions such as `nfold` that is the number of folds for the cross-validation procedure.

```
> myres.cv <- netinf.cv(data = data1, categories = 3, priors = priors.count,
+     maxparents = 3, priors.weight = 0.5, method = "regrnet",
+     nfold = 10, seed = 54321)
```

The object `myres.cv` contains a lot of information related to the network inference process:

```
> print(str(myres.cv, 1))
```

```
List of 7
 $ net                 :List of 2
 $ net.cv              :List of 10
 $ topology            : num [1:20, 1:20] 0 0 0 0 0 1 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
 $ topology.cv         :List of 10
 $ prediction.score.cv:List of 3
 $ edge.stability      : num [1:20, 1:20] 0 0 0 0 0 1 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
 $ edge.stability.cv   : num [1:20, 1:20] 0 0 0 0 0.1 1 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
NULL
```

with

**net** Model object of the network inferred using the entire dataset.

**net.cv** List of models network models fitted at each fold of the cross-validation.

**topology** Topology of the model inferred using the entire dataset.

[8] http://en.wikipedia.org/wiki/Cross-validation_(statistics)

**topology.cv** Topology of the networks inferred at each fold of the cross-validation.

**prediction.score.cv** List of prediction scores ($R^2$, *NRMSE*, *MCC*) computed at each fold of the cross-validation.

**edge.stability** Stability of the edges inferred during cross-validation; only the stability of the edges present in the network inferred using the entire dataset is reported.

**edge.stability.cv** Stability of the edges inferred during cross-validation.

We can now extract these statistics to better quantify the robustness of the network inference.

### 4.3 Edge-specific stability

At each fold of the cross-validation, a network is inferred using a slightly different dataset. This variation in the set of observations used to fit the network model induces some variation at the level of the inference. Some edges may be poorly supported by the data and therefore their inference strongly depends on the training dataset and is not generalizable. Since we performed a 10-fold cross-validation, we can calculate for each edge its presence frequency in the inferred networks, the most robust edge being inferred 10 times, the less robust ones only once or twice.

Let's display the topology of the network inferred from the entire dataset where the edges are colored according to their stability (Figure 4).

```
> ## 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[myres.cv$edge.stability == names(mycols)[k]] <- mycols[k]
+ }
> myedgecols[!mytopoglobal] <- "#00000000"


> def.par <- par(no.readonly = TRUE)
> layout(rbind(1, 2), heights = rbind(8, 1))
> par(mar = c(1, 1, 1, 1))
> ## network topology
> xnet3 <- network(x = mytopoglobal, matrix.type = "adjacency",
+     directed = TRUE, loops = FALSE, vertex.attrnames = dimnames(mytopoglobal)[[1]])

> plot.network(x = xnet3, displayisolates = TRUE, displaylabels = TRUE,
+     boxed.labels = FALSE, label.pos = 0, arrowhead.cex = 1.5,
+     vertex.cex = 2, 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)
> par(def.par)
```

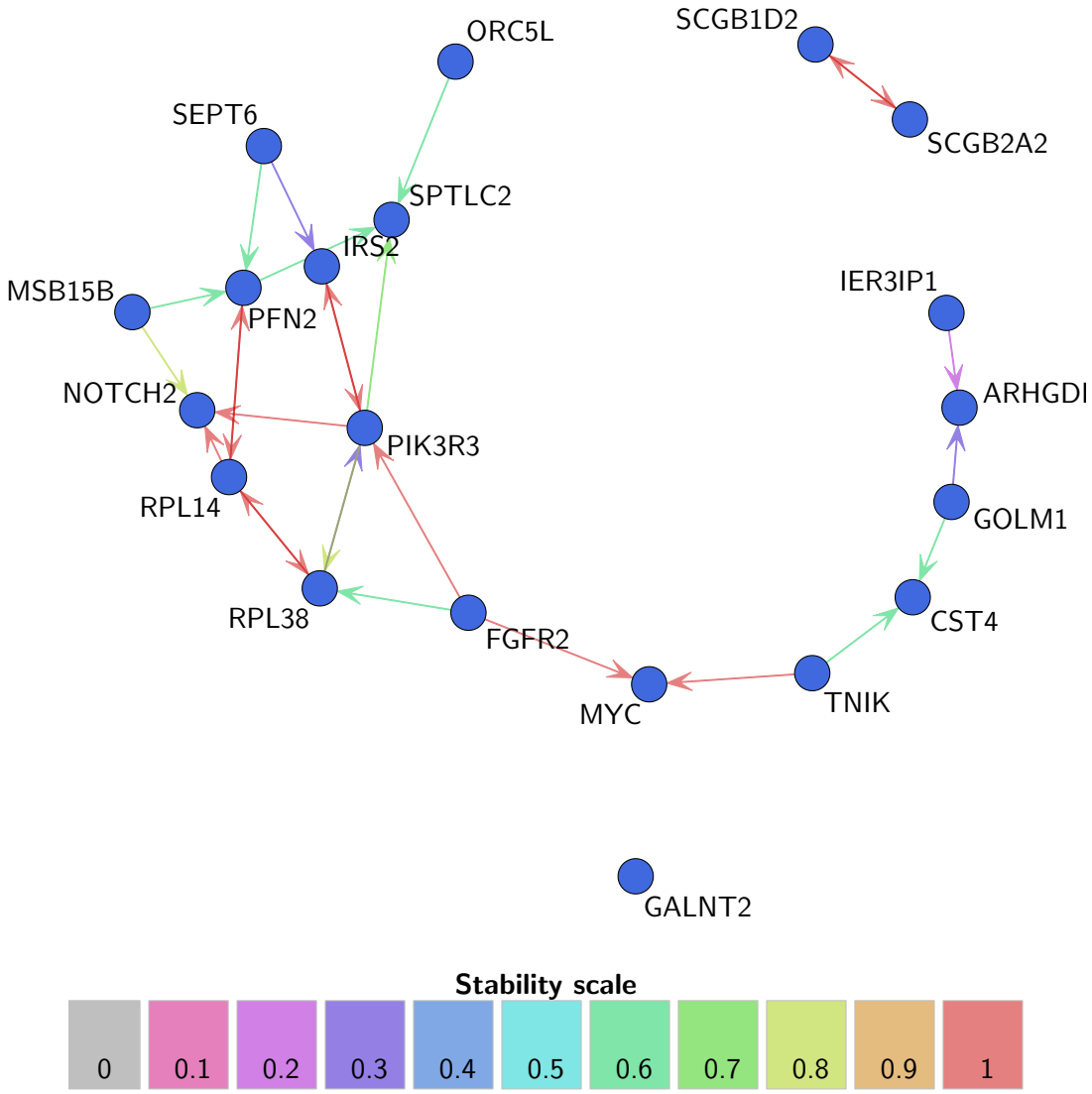


Figure 4: Network topology where the edges are colored according to their stability.

As can be seen, the inference of half of the edges is very stable, especially around the two "hubs" that are RPL14 and PIK3R3. However the inference of the interactions with SPET6 is unstable.

### 4.4 Gene-specific predictive ability

Since our regression-based network inference approach actually fits local regression models to assess the dependence between the parent genes and the target/child gene, we can actually quantify the strength of this dependence by assessing the predictive ability of the network model for each individual gene. Several performance criteria have been implemented so far:

- $R^2$: proportion of variance of the target/child gene explained by the regression model. The value lies in $\{0, 1\}$, the larger, the better.
- *NRMSE*: normalized root mean squared error of the regression model. The values are $> 0$, the lower, the better.
- *MCC*: Matthews correlation coefficient (also called multi-class correlation). This is a performance criterion widely used in the classification framework so it requires first a discretization of the gene expression data in classes (this is done implicitly by the functions `netinf.cv` and `pred.score`). The value lies in $\{0, 1\}$, the larger, the better.

Let's focus on $R^2$ and *MCC*. We can easily display the topology by coloring nodes according to the predictive ability of the network estimated by $R^2$ (Figure 5) and *MCC* (Figure 6). Here is the code for generating the plot reporting the $R^2$ estimations:

```
> myr2 <- apply(myres.cv$prediction.score.cv$r2, 2, mean, na.rm = TRUE)
> myr2 <- round(myr2 * 10)/10
> ## preparing colors
> ii <- 0:10
> mycols <- c("#BEBEBE", rev(rainbow(10, v = 0.8, alpha = 0.5)))
> names(mycols) <- as.character(ii/10)
> myvertexcols <- mycols[match(myr2, names(mycols))]
> names(myvertexcols) <- names(myr2)


> def.par <- par(no.readonly = TRUE)
> layout(rbind(1, 2), heights = rbind(8, 1))
> par(mar = c(1, 1, 1, 1))
> ## network topology
> xnet3 <- network(x = mytopoglobal, matrix.type = "adjacency",
+     directed = TRUE, loops = FALSE, vertex.attrnames = dimnames(mytopoglobal)[[1]])

> plot.network(x = xnet3, displayisolates = TRUE, displaylabels = TRUE,
+     boxed.labels = FALSE, label.pos = 0, arrowhead.cex = 1.5,
+     vertex.cex = 2, vertex.col = myvertexcols, jitter = FALSE,
+     pad = 0.5, edge.col = "black", 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 = "R^2", 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)
```

```
> par(def.par)
```

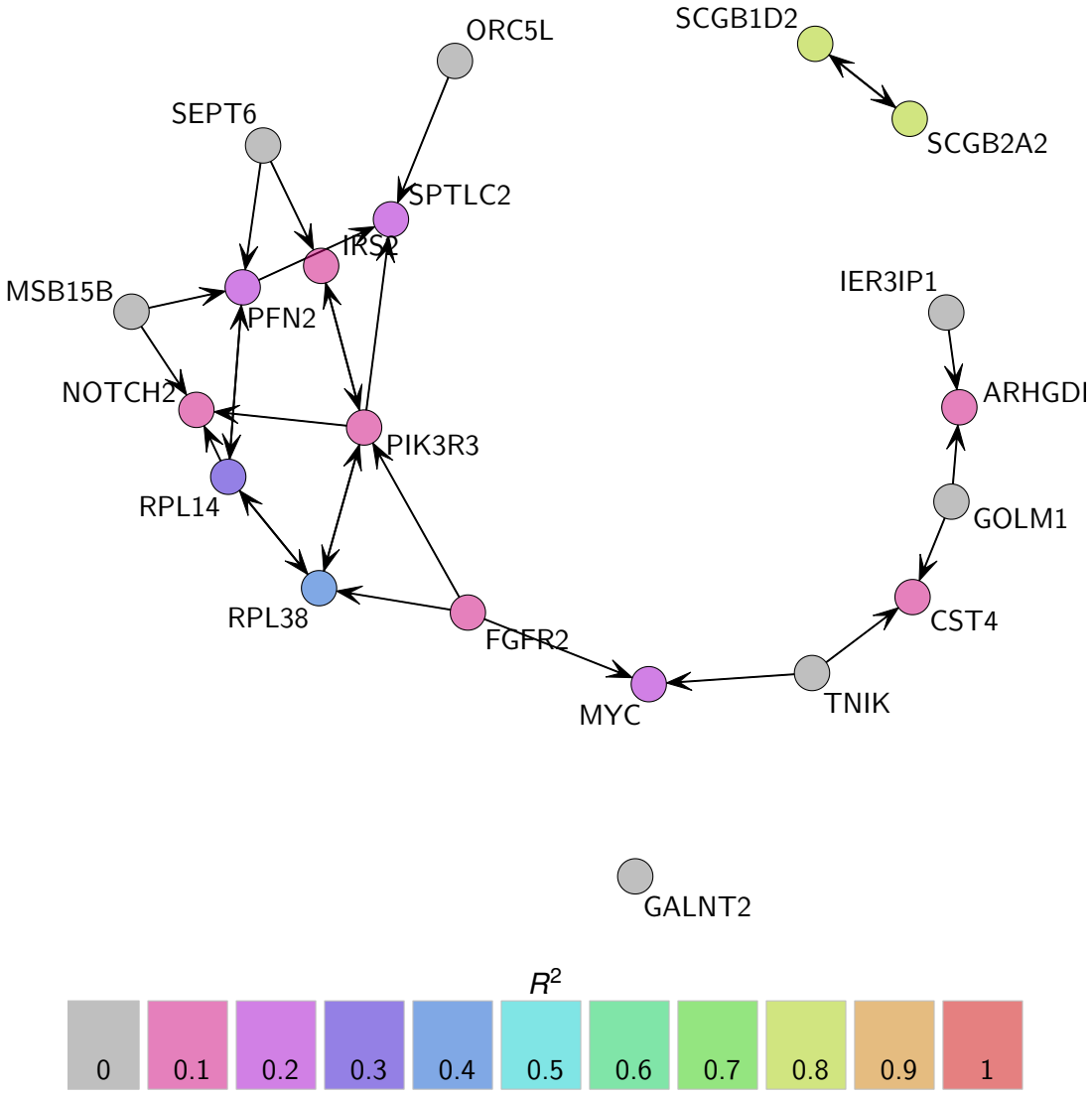


Figure 5: Network topology where the genes are colored according to the predictive performance of the network measure by $R^2$ in cross-validation.

A similar piece of code could be used to generate the plot reporting the *MCC* estimations.

As can be seen in Figures 5 and 6, the genes SCGB2A2 and SCGB1D2 are surprisingly well predicted by the network; in fact these two gene are highly correlated (Pearson's correlation coefficient of 0.9). If we look at the rest of the genes we can see that the assessment of the network predictive ability is concordant with its stability analysis: the genes around RPL14 and PIK3R3 are quite well predicted while the genes connected with unstable edges are poorly predicted.

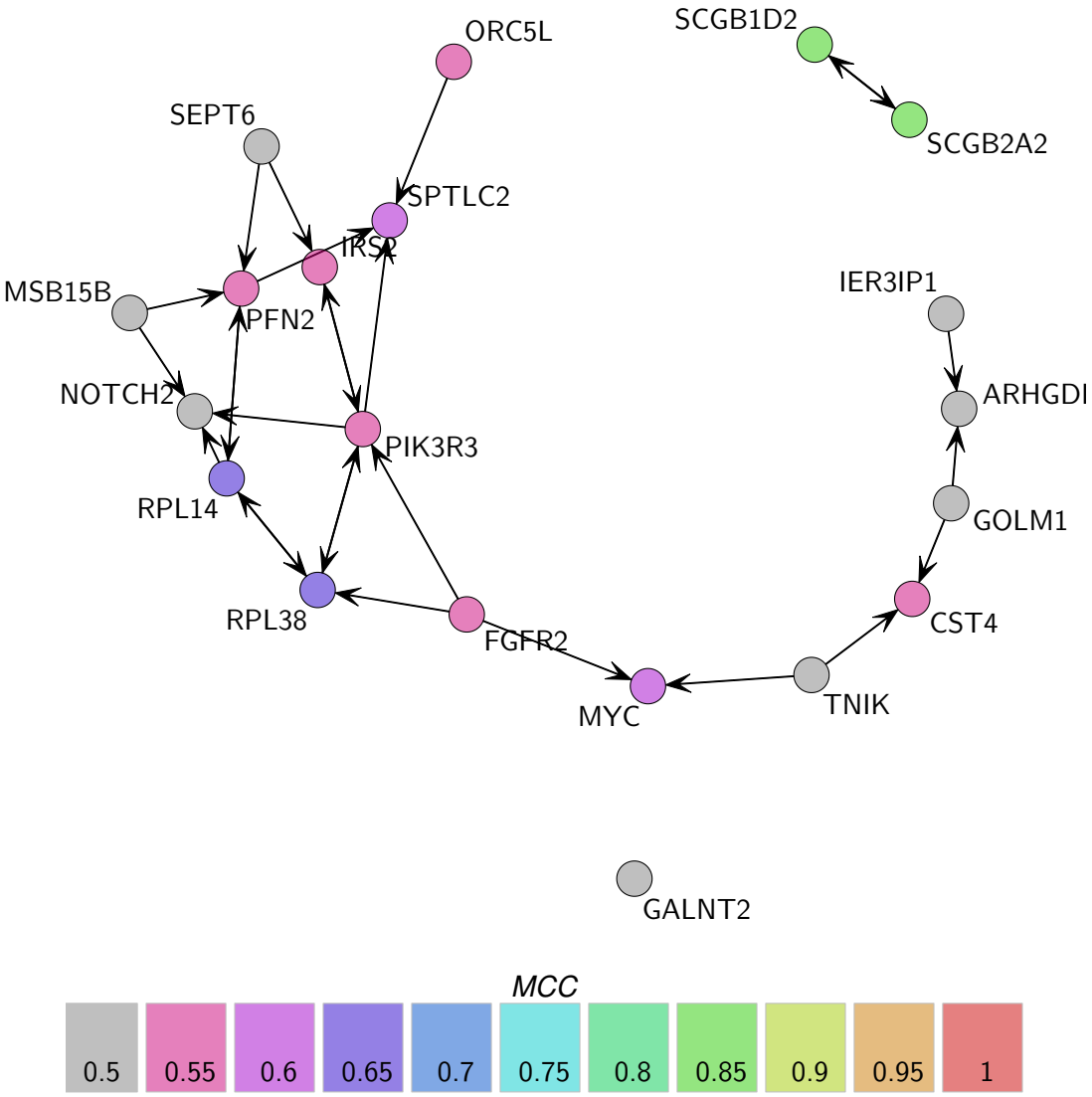


Figure 6: Network topology where the genes are colored according to the predictive performance of the network measured by *MCC* in cross-validation.

**Validation** Even though we used a proper cross-validation procedure to get an unbiased estimate of the predictive ability of the model, it is of interest to validate our network model in a fully independent dataset. The task is challenging because of the various biases that are inevitably present in a dataset generated from a different population of breast cancer patients and in a different lab. Although the resulting hidden batch effects might dramatically drive down the performance of a model, this additional validation step is necessary to assess the quality of the model in a real word situation where we cannot control for all the batch effects.

In this course, we use our second dataset of breast cancer patients to validate the performance of the network model inferred from the first dataset. we can actually compute the predictions and the corresponding $R^2$ and *MCC* estimates using the following pice of code and then display the

network using the same code than before (see Figures 7 and 8 for the $R^2$ and *MCC* respectively).

```
> ## compute predictions
> mynet.test.pred <- netinf.predict(net = mynet, data = data2)
> ## performance estimation: R2
> mynet.test.r2 <- pred.score(data = data2, pred = mynet.test.pred,
+     method = "r2")
> ## performance estimation: MCC
> mynet.test.mcc <- pred.score(data = data2, categories = 3, pred = mynet.test.pred,
+     method = "mcc")
```

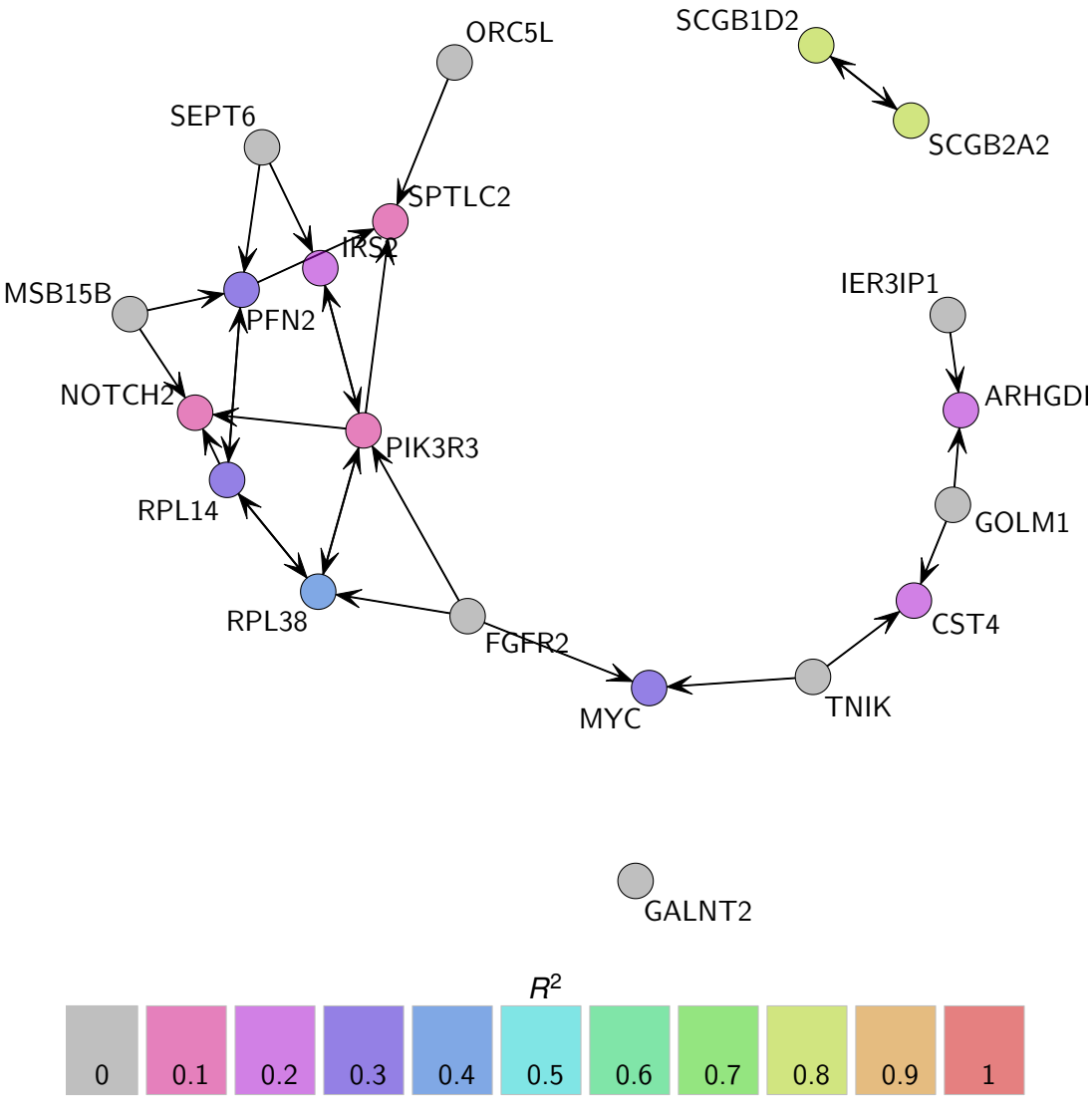


Figure 7: Network topology where the genes are colored according to the predictive performance of the network measured by $R^2$ in a fully independent dataset.

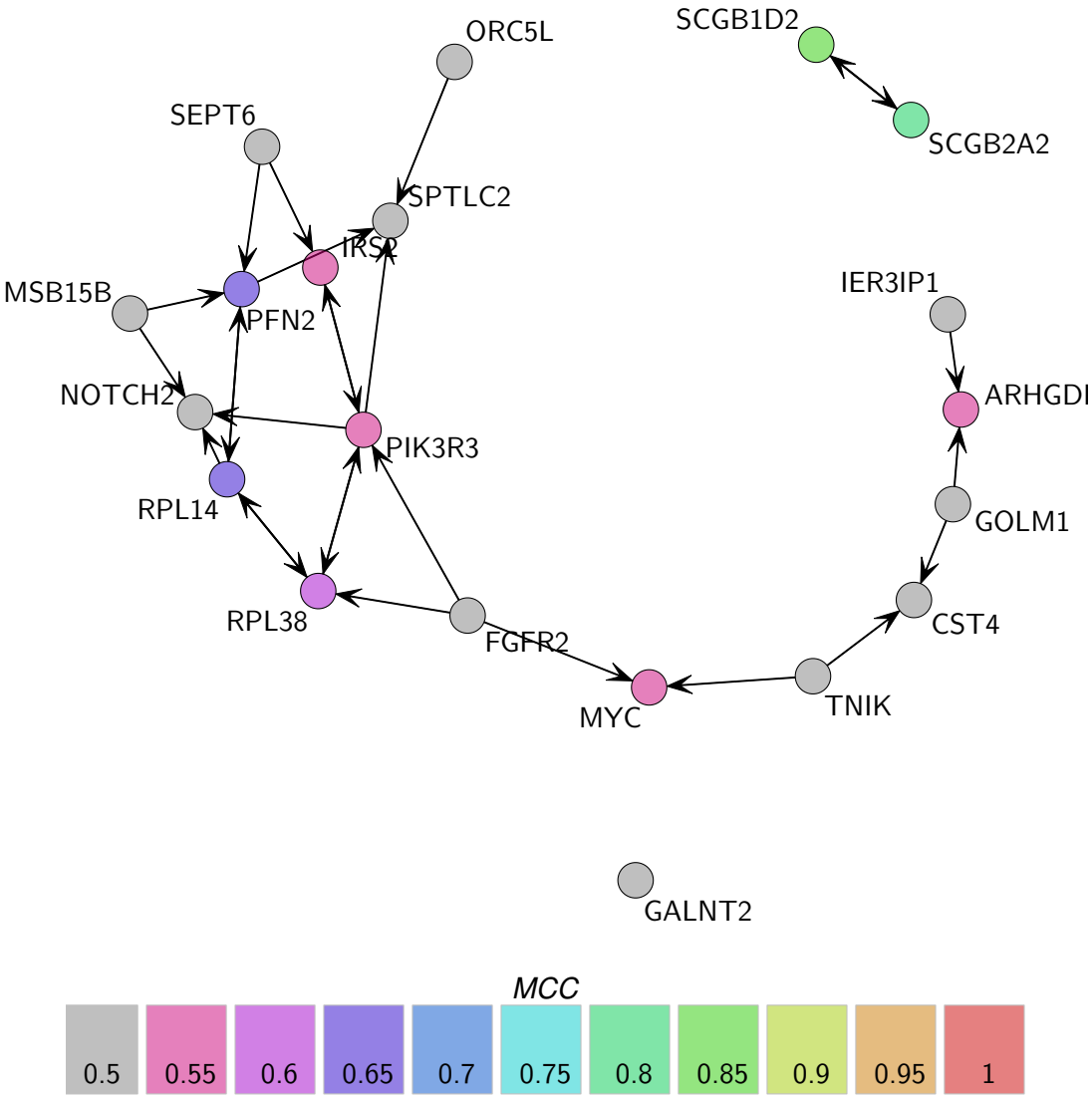


Figure 8: Network topology where the genes are colored according to the predictive performance of the network measured by *MCC* in a fully independent dataset.

As expected, the performance of our model slightly decreased but the vast majority of our observations remain true.

# 5 Statistical comparison of network models

Based on our approach to quantify the stability and predictive ability of a gene interaction network, we are now able to statistically compare the performance of two or more networks. This network model selection could help us optimize a parameter such as the weight of the priors during the network inference (`priors.weight`) or identify the best method of network inference (Bayesian vs regression-based network inference for instance).

Let's infer and compare the predictive ability (estimated in cross-validation) of five networks with `priors.weight` = 0, 0.25, 0.5, 0.75 and 1.

```
> ## priors.weight 0
> myres.cv.pw0 <- netinf.cv(data = data1, categories = 3, priors = priors.count,
+     maxparents = 3, priors.weight = 0, method = "regrnet", nfold = 10,
+     seed = 54321)
> ## priors.weight 0.25
> myres.cv.pw025 <- netinf.cv(data = data1, categories = 3, priors = priors.count,
+     maxparents = 3, priors.weight = 0.25, method = "regrnet",
+     nfold = 10, seed = 54321)
> ## priors.weight 0.5
> myres.cv.pw050 <- myres.cv
> ## priors.weight 0.75
> myres.cv.pw075 <- netinf.cv(data = data1, categories = 3, priors = priors.count,
+     maxparents = 3, priors.weight = 0.75, method = "regrnet",
+     nfold = 10, seed = 54321)
> ## priors.weight 0
> myres.cv.pw1 <- netinf.cv(data = data1, categories = 3, priors = priors.count,
+     maxparents = 3, priors.weight = 1, method = "regrnet", nfold = 10,
+     seed = 54321)
```

Now let's display the topology of the networks we just inferred by varying the weight we put on the priors (see Figure 9).

```
> def.par <- par(no.readonly = TRUE)
> layout(mat = matrix(1:4, nrow = 2, ncol = 2, byrow = TRUE))
> ## priors.weight 0
> mytopot <- net2topo(net = myres.cv.pw0$net, coefficients = FALSE)
> xnett <- network(x = mytopot, matrix.type = "adjacency", directed = TRUE,
+     loops = FALSE, vertex.attrnames = dimnames(mytopot)[[1]])
> plot.network(x = xnett, displayisolates = TRUE, displaylabels = FALSE,
+     boxed.labels = FALSE, label.pos = 0, arrowhead.cex = 1.5,
+     vertex.cex = 2, vertex.col = "royalblue", jitter = FALSE,
+     pad = 0.5, edge.col = "black", coord = mynetlayout, main = "Priors weight
= 0 (data only)")
> ## priors.weight 0.25
> mytopot <- net2topo(net = myres.cv.pw025$net, coefficients = FALSE)
> xnett <- network(x = mytopot, matrix.type = "adjacency", directed = TRUE,
```

```
+     loops = FALSE, vertex.attrnames = dimnames(mytopot)[[1]])
> plot.network(x = xnett, displayisolates = TRUE, displaylabels = FALSE,
+     boxed.labels = FALSE, label.pos = 0, arrowhead.cex = 1.5,
+     vertex.cex = 2, vertex.col = "royalblue", jitter = FALSE,
+     pad = 0.5, edge.col = "black", coord = mynetlayout, main = "Priors weight
= 0.25")
> ## priors.weight 0.75
> mytopot <- net2topo(net = myres.cv.pw075$net, coefficients = FALSE)
> xnett <- network(x = mytopot, matrix.type = "adjacency", directed = TRUE,
+     loops = FALSE, vertex.attrnames = dimnames(mytopot)[[1]])
> plot.network(x = xnett, displayisolates = TRUE, displaylabels = FALSE,
+     boxed.labels = FALSE, label.pos = 0, arrowhead.cex = 1.5,
+     vertex.cex = 2, vertex.col = "royalblue", jitter = FALSE,
+     pad = 0.5, edge.col = "black", coord = mynetlayout, main = "Priors weight
= 0.75")
> ## priors.weight 1
> mytopot <- net2topo(net = myres.cv.pw1$net, coefficients = FALSE)
> xnett <- network(x = mytopot, matrix.type = "adjacency", directed = TRUE,
+     loops = FALSE, vertex.attrnames = dimnames(mytopot)[[1]])
> plot.network(x = xnett, displayisolates = TRUE, displaylabels = FALSE,
+     boxed.labels = FALSE, label.pos = 0, arrowhead.cex = 1.5,
+     vertex.cex = 2, vertex.col = "royalblue", jitter = FALSE,
+     pad = 0.5, edge.col = "black", coord = mynetlayout, main = "Priors weight
= 1 (priors only)")
> par(def.par)
```

As can be seen in Figure 9, the networks generated from data only (`priors.weight` $= 0$) and from priors only (`priors.weight` $= 1$) are very sparse, and the combination of priors and data enables the identification of more gene interactions. But a denser topology does not imply that all the inferred interactions are well supported by the data. In order to select the best network(s) we can statistically compare their stability and predictive ability.

## 5.1 Comparison of edge-specific stability

In order to compare the stability of the network inference, we can draw a boxplot representing the edge-specific stability of the network inference with respect to the weight put on the priors (Figure 10). As expected, the network inferred from priors only is perfectly stable (the inference does not depend on the data anymore) but what is interesting is the important gain in stability when the priors are used as compared to networks inferred from data alone.

```
> gg <- c("0", "0.25", "0.50", "0.75", "1")
> mystab.cv.pw <- list(myres.cv.pw0$edge.stability[myres.cv.pw0$topology ==
+     1], myres.cv.pw025$edge.stability[myres.cv.pw025$topology ==
+     1], myres.cv.pw050$edge.stability[myres.cv.pw050$topology ==
+     1], myres.cv.pw075$edge.stability[myres.cv.pw075$topology ==
+     1], myres.cv.pw1$edge.stability[myres.cv.pw1$topology ==
+     1])
```

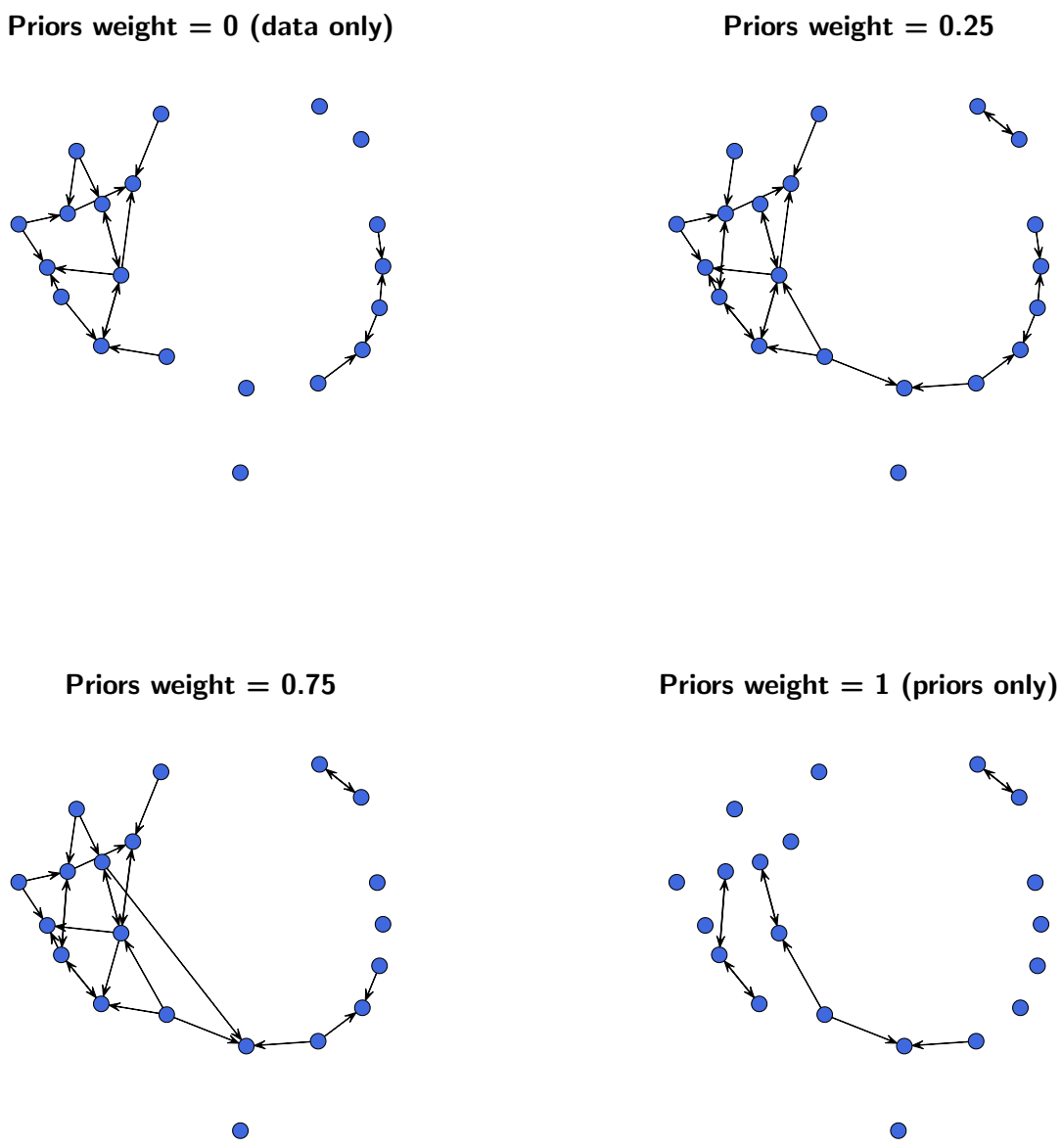


Figure 9: Directed graph representing the topology of the networks inferred from priors and gene expression data with varying prior weight.

```
> names(mystab.cv.pw) <- gg
> boxplot(mystab.cv.pw, xlab = "priors.weight", ylim = c(0, 1),
+     ylab = "Edge stability", border = "grey", col = "lightgrey")
> points(x = jitter(x = rep(1:length(mystab.cv.pw), times = sapply(mystab.cv.pw,
```

```
+      length)), amount = 0.15), y = unlist(mystab.cv.pw), cex = 0.55,
+      pch = 16, col = "royalblue")
```

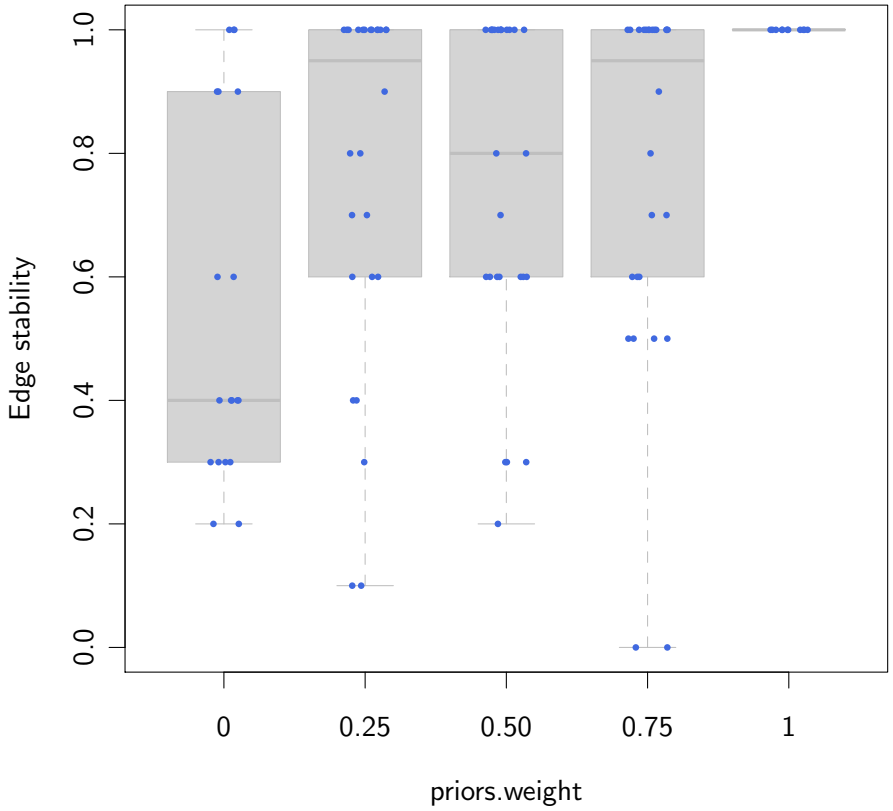


Figure 10: Boxplot of the edge-specific stability computed in cross-validation, for network models with different prior weight.

## 5.2 Comparison of gene-specific predictive ability

In terms of predictive ability, the boxplot representing the gene-specific $R^2$ scores computed in cross-validation, for network models with different prior weights (Figure 11), suggests that a combination of both data and priors yield better predictive ability.

```
> gg <- c("0", "0.25", "0.50", "0.75", "1")
> myr2.cv.pw <- cbind(apply(myres.cv.pw0$prediction.score.cv$r2,
+      2, mean, na.rm = TRUE), apply(myres.cv.pw025$prediction.score.cv$r2,
+      2, mean, na.rm = TRUE), apply(myres.cv.pw050$prediction.score.cv$r2,
+      2, mean, na.rm = TRUE), apply(myres.cv.pw075$prediction.score.cv$r2,
+      2, mean, na.rm = TRUE), apply(myres.cv.pw1$prediction.score.cv$r2,
```

```
+     2, mean, na.rm = TRUE))
> colnames(myr2.cv.pw) <- gg
> gg <- factor(rep(gg, each = genen), levels = gg)
> boxplot(myr2.cv.pw, xlab = "priors.weight", ylim = c(0, 1), ylab = "$R^2$",
+     border = "grey", col = "lightgrey", outline = FALSE)
> points(x = jitter(x = rep(1:ncol(myr2.cv.pw), times = nrow(myr2.cv.pw)),
+     amount = 0.15), y = as.vector(t(myr2.cv.pw)), cex = 0.55,
+     pch = 16, col = "royalblue")
```

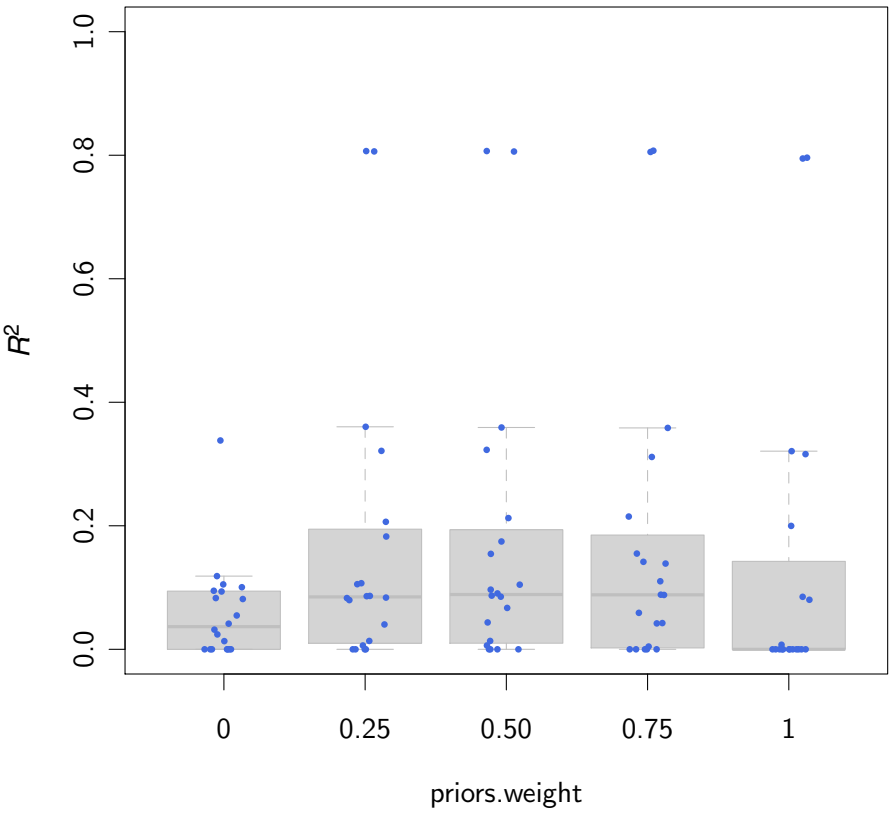


Figure 11: Boxplot of the gene-specific $R^2$ scores computed in cross-validation, for network models with different prior weight.

However a rigorous statistical comparison does not allow us to claim that a network model inferred from data+priors yields always significantly better predictive ability than networks inferred from data or priors only. The Friedman test tells us that at least one of the network models yielded significantly different predictive ability; the pairwise Wilcoxon Rank Sum tests suggest that the model using the priors only (`priors.weight` $= 1$) is significantly less predictive than all the other network models; however the model inferred from data only does not yield statistically different

predictive ability (although the p-value is close to significance)[9].

```
> ## Friedman test to test whether at least one of the
> #   networks gives statically different predictive ability
> print(friedman.test(y = myr2.cv.pw))


        Friedman rank sum test

data:  myr2.cv.pw
Friedman chi-squared = 32.1019, df = 4, p-value = 1.824e-06

> ## Pairwise Wilcoxon Rank Sum tests
> print(pairwise.wilcox.test(x = as.vector(myr2.cv.pw), g = gg,
+     paired = TRUE, exact = FALSE, alternative = "two.sided",
+     p.adjust.method = "bonferroni"))


        Pairwise comparisons using Wilcoxon signed rank test

data:  as.vector(myr2.cv.pw) and gg

     0      0.25   0.50   0.75
0.25 0.2139 -      -      -
0.50 0.1048 1.0000 -      -
0.75 0.1149 1.0000 1.0000 -
1    1.0000 0.0048 0.0048 0.0109

P value adjustment method: bonferroni
```

Similar conclusions can be drawn by computing the *MCC*.

**Validation** The same analysis using the fully independent dataset to estimate the predictive ability of the different models supports the conclusions we drew from the cross-validation study in the training set.

```
> gg <- c("0", "0.25", "0.50", "0.75", "1")
> myr2.test.pw <- cbind(pred.score(data = data2, pred = netinf.predict(net = myres.cv.pw0$
+     data = data2), method = "r2"), pred.score(data = data2, pred = netinf.predict(net
= myres.cv.pw025$net,
+     data = data2), method = "r2"), pred.score(data = data2, pred = netinf.predict(net
= myres.cv.pw050$net,
+     data = data2), method = "r2"), pred.score(data = data2, pred = netinf.predict(net
= myres.cv.pw075$net,
+     data = data2), method = "r2"), pred.score(data = data2, pred = netinf.predict(net
= myres.cv.pw1$net,
```

[9] An analysis involving more genes and better priors could show that network inferred from a combination of priors and data always lead to significantly more predictive models.

```
+     data = data2), method = "r2"))
> colnames(myr2.test.pw) <- gg
> gg <- factor(rep(gg, each = genen), levels = gg)
> boxplot(myr2.test.pw, xlab = "priors.weight", ylim = c(0, 1),
+     ylab = "$R^2$", border = "grey", col = "lightgrey", outline = FALSE)
> points(x = jitter(x = rep(1:ncol(myr2.test.pw), times = nrow(myr2.test.pw)),
+     amount = 0.15), y = as.vector(t(myr2.test.pw)), cex = 0.55,
+     pch = 16, col = "royalblue")
```

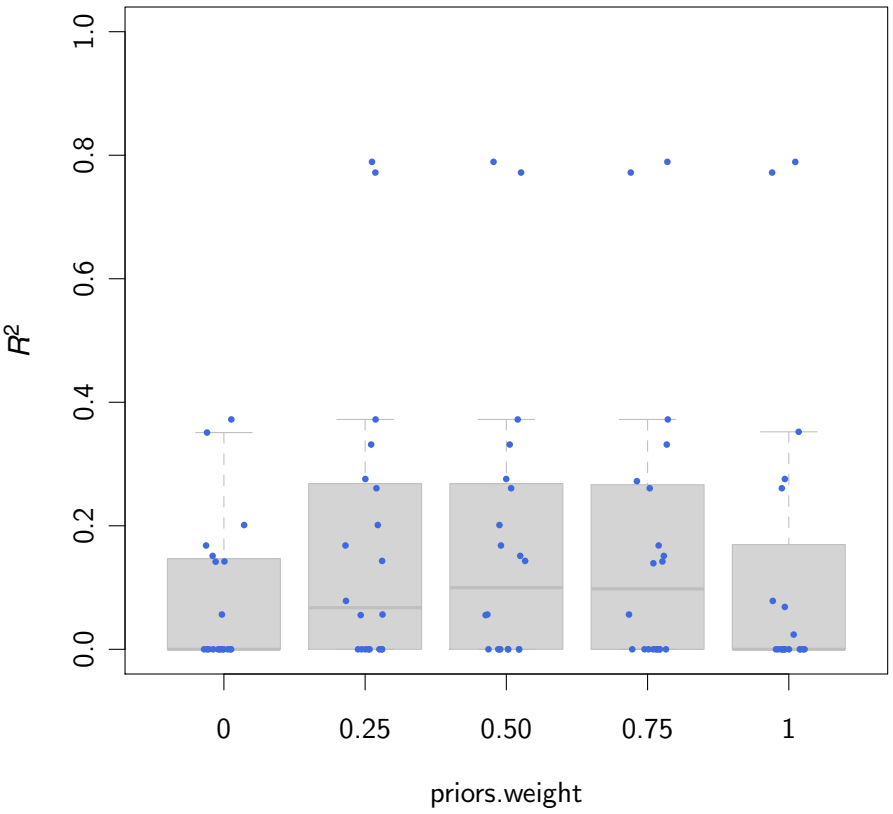


Figure 12: Boxplot of the gene-specific $R^2$ scores computed in the fully independent dataset, for network models with different prior weight.

```
> ## Friedman test to test whether at least one of the
> #   networks gives statically different predictive ability
> print(friedman.test(y = myr2.test.pw))

        Friedman rank sum test
```

```
data:  myr2.test.pw
Friedman chi-squared = 13.0292, df = 4, p-value = 0.01113
```

```
> ## Pairwise Wilcoxon Rank Sum tests
> print(pairwise.wilcox.test(x = as.vector(myr2.test.pw), g = gg,
+     paired = TRUE, exact = FALSE, alternative = "two.sided",
+     p.adjust.method = "bonferroni"))
```

```
        Pairwise comparisons using Wilcoxon signed rank test

data:  as.vector(myr2.test.pw) and gg

     0     0.25  0.50  0.75
0.25 1.000 -     -     -
0.50 1.000 1.000 -     -
0.75 1.000 1.000 1.000 -
1    1.000 0.143 0.092 0.191

P value adjustment method: bonferroni
```

# 6 Future works

The *Predictive Networks* web application and the `predictionet` R package that we are using today are currently under active development and new features will be introduced progressively. Here is a list of the major features that will included soon:

- *Predictive Networks* web application:
  - Option to filter gene interactions according to their domain (e.g., breast cancer).
  - Implementation of an interactive visualization tool to allow users play with the inferred networks and identify the parts of the networks that are novel and well supported by the data.
- *predictionet* R package:
  - Implementation of a bayesian network inference method using the *catnet* R package.
  - Development of a novel approach to improve the completeness of the inferred network (and so get a more biological meaningful network) and its predictive ability. The idea behind this *ensemble network inference* method is to generate many different networks that are equally well supported by the data and combine them in a consensus network instead of a single (local) optimum.
  - Allow users to export a Cytoscape[10] file to visualize complex network in an efficient way.

# 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

[10] `http://www.cytoscape.org/`

## References


*Learning Bayesian Networks*. Prentica Hall, 2003.

Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, and Weiru Liu. Learning bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1-2):43 – 90, 2002. ISSN 0004-3702. doi: DOI:10.1016/S0004-3702(02)00191-1. URL http://www.sciencedirect.com/science/article/B6TYF-45BCRWV-2/2/390d83eb89efc5e0bc8de9e43e0681d9.

Christine Desmedt, Fanny Piette, Sherene Loi, Yixin Wang, Francoise Lallemand, Benjamin Haibe-Kains, Giuseppe Viale, Mauro Delorenzi, Yi Zhang, Mahasti Saghatchian d'Assignies, Jonas Bergh, Rosette Lidereau, Paul Ellis, Adrian L. Harris, Jan G.M. Klijn, John A. Foekens, Fatima Cardoso, Martine J. Piccart, Marc Buyse, and Christos Sotiriou. Strong Time Dependence of the 76-Gene Prognostic Signature for Node-Negative Breast Cancer Patients in the TRANSBIG Multicenter Independent Validation Series. *Clin Cancer Res*, 13(11):3207–3214, 2007. doi: 10.1158/1078-0432.CCR-06-2765. URL http://clincancerres.aacrjournals.org/cgi/content/abstract/13/11/3207.

Chris Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *J Bioinform Comput Biol*, 3(2):185–205, Apr 2005.

Sherene Loi, Benjamin Haibe-Kains, Samira Majjaj, Francoise Lallemand, Virginie Durbecq, Denis Larsimont, Ana M. Gonzalez-Angulo, Lajos Pusztai, W. Fraser Symmans, Alberto Bardelli, Paul Ellis, Andrew N. J. Tutt, Cheryl E. Gillett, Bryan T. Hennessy, Gordon B. Mills, Wayne A. Phillips, Martine J. Piccart, Terence P. Speed, Grant A. McArthur, and Christos Sotiriou. PIK3CA mutations associated with gene signature of low mTORC1 signaling and better outcomes in estrogen receptor–positive breast cancer. *Proceedings of the National Academy of Sciences*, 107(22):10208–10213, 2010. doi: 10.1073/pnas.0907011107. URL http://www.pnas.org/content/107/22/10208.abstract.

Patrick E Meyer, Kevin Kontos, Frederic Lafitte, and Gianluca Bontempi. Information-theoretic inference of large transcriptional regulatory networks. *EURASIP J Bioinform Syst Biol*, page 79879, 2007. doi: 10.1155/2007/79879.

Patrick Emmanual Meyer. *Information-Theoretic Variable Selection and Network Inference from Microarray Data*. PhD thesis, Université Libre de Bruxelles, December 2008.

Andy J. Minn, Gaorav P. Gupta, David Padua, Paula Bos, Don X. Nguyen, Dimitry Nuyten, Bas Kreike, Yi Zhang, Yixin Wang, Hemant Ishwaran, John A. Foekens, Marc van de Vijver, and Joan Massague. Lung metastasis genes couple breast tumor size and metastatic spread. *Proceedings of the National Academy of Sciences*, 104(16):6740–6745, 2007. doi: 10.1073/pnas.0701138104. URL http://www.pnas.org/cgi/content/abstract/104/16/6740.

C. Olsen, P. E. Meyer, and G. Bontempi. Inferring causal relationships using information-theoretic measures. In *Proceedings of the 5th Benelux Bioinformatics Conference (BBC09)*, 2009.

M. Y. Park and T. Hastie. L1 regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society*, 69:659–677, 2007.

Y. Wang, J. G. Klijn, Y. Zhang, A. M. Sieuwerts, M. P. Look, F. Yang, D. Talantov, M. Timmermans, M. E. Meijer van Gelder, J. Yu, T. Jatkoe, E. M. Berns, D. Atkins, and J. A. Foekens. Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *Lancet*, 365(9460):671–679, 2005.