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

October 13, 2015

Contents

1 Introduction

DNA microarrays and other high-throughput omics technologies provide large datasets that often include 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 vignette we introduce a platform developed in John Quackenbush's lab, which enables inference of predictive gene interaction networks from prior biological knowledge, in the form of biomedical literature and structured databases, and from gene expression data. Using real data, we will show the benefit of using prior biological knowledge to infer networks and how to quantitatively assess the quality of such networks.

Getting started

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

```
> library(predictionet)
```

This will load *predictionet* as well as its dependencies. In this vignette we will use two example datasets included in the *predictionet* package, namely `exp0.colon.ras` and `jorissen.colon.ras`.

The vignette 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 list of genes of interest.
2. Extract priors from the biomedical literature and public structured databases using the *Predictive Networks* web application.
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 two colon cancer gene expression datasets as examples (see Section 2) we go step by step and provide all the R code necessary to perform the entire analysis.

2 Biology and Data

Let's focus on colon cancer and more specifically the RAS signaling pathway.

2.1 RAS signaling pathway

A series of alterations in the cellular genome affecting the expression or function of genes controlling cell growth and differentiation is considered to be the main cause of cancer. These mutational events include activation of oncogenes and inactivation of tumor suppressor genes. The elucidation of human cancer at the molecular level allows the design of rational, mechanism-based therapeutic agents that antagonize the specific activity of biochemical processes that are essential to the malignant phenotype of cancer cells. Because the frequency of RAS mutations is among the highest for any gene in human cancers, development of inhibitors of the Ras-mitogen-activated protein kinase (RAS/MAPK) pathway as potential anticancer agents is a very promising pharmacologic strategy [Reuter et al., 2000].

Bild et al. identified a list of genes being differentially expressed between colorectal cancer cell lines carrying the RAS mutation and those with the wild-type RAS gene [Bild et al., 2006]. This gene list is provided in `files/bild2006_ras_signature_348.csv` and will serve as the core set of genes involved in the RAS pathway.

2.2 Colon cancer gene expression data

We use two large gene expression datasets of primary colon tumors collected before any adjuvant therapy. The first dataset, provided in the *predictionet* package (see `?expO.colon.ras`), was published by the expO project[1] and consists of 292 colon tumors hybridized on the Affymetrix GeneChip HG-U133PLUS2, composed of 54,675 probesets. The second dataset, also provided in the *predictionet* package (see `?jorissen.colon.ras`) was published in [Jorissen et al., 2009] and consists of 290 colon tumors hybridized on the same Affymetrix GeneChip (HG-U133PLUS2). The raw data have been collected from GEO[2], series accession numbers GSE2109 and GSE14333 for the first and the second dataset respectively

Data preprocessing The raw files (`*.CEL`) have been normalized using `frma` [McCall et al., 2010].

Then only a subset of the gene expressions has been kept for further analysis of the RAS signaling pathway by selecting the probesets in Bild's RAS signature (see `files/bild2006_ras_signature_34` which represent a unique gene. When multiple probesets represented the same gene, the most variant has ben selected. The final datasets contain 259 genes and are stored in the *predictionet* package (see `?expO.colon.ras` and `?jorissen.colon.ras`).

2.3 Known gene interactions extracted 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

[1] `http://www.intgen.org/expo/`
[2] `http://www.ncbi.nlm.nih.gov/geo/`

at ENTAGEN[3]. 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 (referred to as gene list).

One can use the *Predictive Networks* web application (Figure 1) to retrieve a list of interactions involving at least one gene included in our list of RAS-related genes. To do so, one has to first create an account to login to the webapp. Then go to "My Page" and create a new gene list by cutting-and-pasting the gene symbols[4] included in `files/bild2006_ras_signature_348.csv`. Lastly, we have to export all the resulting triples (e.g., "$gene_a$ regulates $gene_b$ "represented by the interaction $gene_a \rightarrow gene_b$) in a CSV file, `priors_ras_bild2006_pnwebapp.csv`, by clicking on "View Triples" and then "Download w/ Sentences As". One can use R to read this file and count how many times a gene interaction has been observed in the biomedical literature and reported structured biological databases.

Figure 1: Screenshot of the *Predictive Networks* web application where one searches known interactions for HRAS.

Thanks to PN a directed graph can be efficiently constructed from known gene interactions which is represented by the matrix $P_{m \times m}$ where m is the number of genes in the network and P_{ij} contains the number of times the interaction $X_i \rightarrow X_j$ between the two genes X_i and X_j has been reported in the biomedical literature and structured biological databases. The prior

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

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

counts in P are then rescaled into $\{-1, 1\}$ where $P_{ij} = 0$ represents no prior information about interaction ij, $P_{ij} = 1$ represents strong evidence for interaction ij and $P_{ij} = -1$ represents strong evidence for the absence of interaction between genes i and j. Such a directed graph constructed from priors will be later combined with a directed graph inferred from genomic data (Figure 2; see Section 3).

Let's read the newly generated priors, `priors_ras_bild2006_pnwebapp.csv`, into a R session.

```
> ## RAS-related genes
> genes.ras <- colnames(data.ras)
> ## read priors generated by the Predictive Networks web application
> pn.priors <- read.csv(system.file(file.path("extdata", "priors_ras_bild2006_pnwebapp.csv
> ## the column names should be: subject, predicate, object, direction, evidence, sentence
>
> ## remove special characters in the gene symbols
> pn.priors[ ,"subject"] <- gsub(pattern="[-]|[+]|[*]|[%]|[$]|[#]|[{]|[}]|[[]|[]]|[|]|[\\^
> pn.priors[ ,"object"] <- gsub(pattern="[-]|[+]|[*]|[%]|[$]|[#]|[{]|[}]|[[]|[]]|[|]|[\\^]
> genes.ras <- gsub(pattern="[-]|[+]|[*]|[%]|[$]|[#]|[{]|[}]|[[]|[]]|[|]|[\\^]", replaceme
> ## missing values
> pn.priors[!is.na(pn.priors) & (pn.priors == "" | pn.priors == " " | pn.priors == "N/A")]
> ## select only the interactions in which the genes are comprised in our gene expression
> myx <- is.element(pn.priors[ , "subject"], genes.ras) & is.element(pn.priors[ , "object"
> 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(head(pn.priors))
```

```
         X subject       predicate object direction evidence
2709 10289   FOSL1  interacts with   JUNB     right positive
838  33995   STX1A  interacts with PLXNA2     right positive
2149 38411    SKP2 does not affect   SKP2     right negative
1938 24386   STX1A     reacts with  STX1A     right positive
1460 28822  TUBA4A  interacts with   MLL3     right positive
422  35640     LYN  interacts with TUBA4A     right positive

2709
838
2149 A conservative change from Asp to Glu at position 331 of Skp2 does not affect Skp2-Ck
1938
1460
422
             article           network
2709            <NA>   Pathway Commons
838             <NA>   Pathway Commons
2149 PubMed:12813041 Medline Abstracts
1938            <NA>   Pathway Commons
```

```
1460            <NA>   Pathway Commons
422             <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
> pn.priors.counts <- matrix(0, nrow=length(genes.ras), ncol=length(genes.ras), dimnames=l
> for(i in 1:nrow(pn.priors)) {
+         switch(tolower(pn.priors[i, "direction"]),
+         "right"={ pn.priors.counts[pn.priors[i, "subject"], pn.priors[i, "object"]] <- p
+         "left"={ pn.priors.counts[pn.priors[i, "object"], pn.priors[i, "subject"]] <- pn
+         { pn.priors.counts[pn.priors[i, "subject"], pn.priors[i, "object"]] <- pn.priors
+         if(pn.priors[i, "object"] != pn.priors[i, "subject"]) { pn.priors.counts[pn.prio
+ }
> ## negative count represent evidence for ABSENCE of an interaction, positive otherwise
```

In this example, few interactions have been previously reported.

```
> print(table(pn.priors.counts))

pn.priors.counts
   -1     0     1
    1 67042    38
```

2.4 Predictionet package

All these function implemented in *predictionet* are listed in the documentation of the package itself.

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

Looking at the help page of `expO.colon.ras` and `jorissen.colon.ras` will give the details about the data that we will use during this course.

```
> help(expO.colon.ras)
> help(jorissen.colon.ras)
```

`expO.colon.ras` and `jorissen.colon.ras` contain three R objects:

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

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

`demo*.ras` data frame of clinical information of the colon cancer patients; patients in rows, clinical variables in columns.

`priors*.ras` matrix of prior information about the gene interactions; parents/sources in rows, children:targets in columns.

Although the objects `data*.ras`, and `demo*.ras` are specific to each dataset, the objects `data*.ras` and `data*.ras` are common in the two datasets under study because the same microarray technology has been used to generate them and we extracted the same genes, and subsequently the same known gene interactions (priors). In this tutorial, we will not used the object `priors.ras` but we will use `pn.priors.counts` instead.

3 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 ≥ 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.

3.1 Methodology

The design of our regression-based approach for network inference is illustrated in Figure 2. The extraction of prior knowledge about gene interactions has been explained in Section 2.3. 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 X_T. The algorithm then selects the genes that exhibit the highest difference between mutual information with the target gene X_T and redundancy with the previously selected genes (referred to as $\mathbf{X}_\mathcal{S}$). This difference acts as a score for the possible interactions.

In the first step of the feature selection procedure, the gene X_j which has the highest mutual information with the target gene X_T is selected, so at this stage $\mathbf{X}_\mathcal{S} = \{X_j\}$. In the

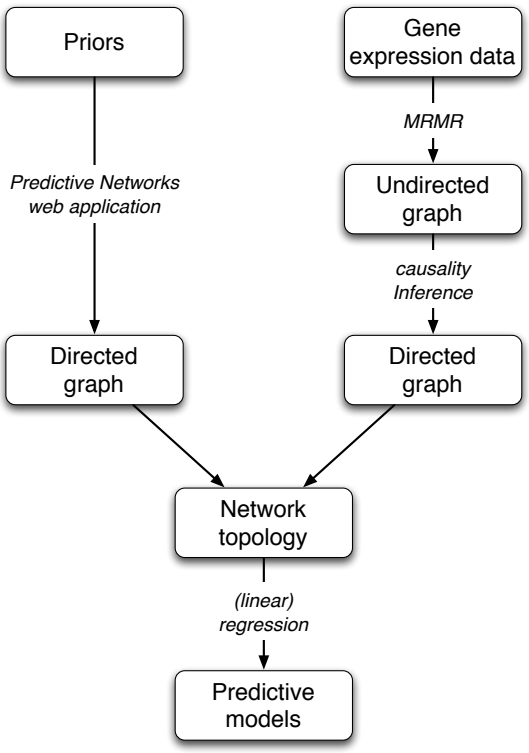

Figure 2: Design of the regression-based network inference approach.

next steps, given a set $\mathbf{X}_{\mathcal{S}}$ of selected variables, the criterion updates $\mathbf{X}_{\mathcal{S}}$ by adding the gene X_k that maximizes the score

$$s_k = I(X_k, X_T) - \frac{1}{|\mathbf{X}_{\mathcal{S}}|} \sum_{X_i \in \mathbf{X}_{\mathcal{S}}} I(X_k, X_i)$$

where $|\mathbf{X}_{\mathcal{S}}|$ is the number of variables previously selected in set $\mathbf{X}_{\mathcal{S}}$. The feature selection procedure stops when the size of $\mathbf{X}_{\mathcal{S}}$, the number of genes connected to the target X_T, is equal to the maximum number of parents allowed by the user.

Note that the resulting undirected graph is locally acyclic since the target gene cannot be selected.

Causality inference

Once the undirected graph is built with the genes included in $\mathbf{X}_{\mathcal{S}}$, 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 common effect creates a dependency between two variables (this can be graphically represented by the v-structure, where the value of the collider is conditioned upon). In [Ding and Peng, 2005, Meyer, 2008] this effect has been related to the property

$$I(X_i, X_k|X_j) > I(X_i, X_k) \Leftrightarrow C(X_i, X_j, X_k) < 0 \tag{1}$$

where X_i and X_k are potential causes of X_j, $I(X_i, X_k)$ is the mutual information of X_i and X_k, $I(X_i, X_k|X_j)$ is the mutual information of X_i and X_k given X_j, and $C(X_i, X_j, X_k)$ is the interaction information of X_i, X_j and X_k such that

$$C(X_i, X_j, X_k) \quad = \quad I(X_i, X_k) - I(X_i, X_k|X_j)$$

The interaction information has been introduced in [McGill, 1954] as an extension of mutual information taking into account sets of variables instead of only pairwise relations.

If the undirected (acyclic) graph is given, then for every triple of variables $X_i - X_j - X_k$ the property (1) corresponds to a v-structure $X_i \rightarrow X_j \leftarrow X_k$. On the contrary, knowing only that the interaction information is less than zero does not help when inferring the network because the three possible colliders exhibit all the same interaction information, that is the difference between mutual information and conditional mutual information. Thus, it is not evident which variable should be the collider.

In this work we use property (1) to rank the genes selected in $\mathbf{X}_\mathcal{S}$ based on their degree of causality for the target gene X_T. More specifically the genes selected in $\mathbf{X}_\mathcal{S}$ are ranked by a causality score

$$q_k = \max_{X_i \in \mathbf{X}_\mathcal{S}} \{-C(X_i, X_T, X_k)\},$$

where for each gene X_k, the score q_k is determined by the maximal negative interaction information between the target gene X_T, X_k and any other variable $X_i \in \mathbf{X}_\mathcal{S}$ forming a possible v-structure $X_i \rightarrow X_T \leftarrow X_k$. The causality score q_k for $X_k \in \mathbf{X}_\mathcal{S}$ lays in $\{-1, 1\}$.

For each target gene X_T, the selected genes $X_k \in \mathbf{X}_\mathcal{S}$ are ranked by their causality score q_k while the genes which are not selected, $X_j \notin \mathbf{X}_\mathcal{S}$, are assigned the minimum causality score of -1. The matrix $Q_{m \times m}$ is then populated such that Q_{ij} is the causality score of the interaction between genes $X_i \rightarrow X_j$. 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 X_i and gene X_j, the score based on priors (P_{ij}) and MRMR+causality inference (Q_{ij}). We let users control the weight put on the priors, which represents their confidence on their prior knowledge (either gathered with the PN web application or another source), using the following combination schema

$$g_{ij} = wP_{ij} + (1 - w)Q_{ij} \tag{2}$$

where g_{ij} is the combined topological score for the interaction $X_i \rightarrow X_j$ and $w \in [0, 1]$. To ensure sparsity of the inferred network only the n interactions with the largest $g_{ij} > 0$ are considered in the network topology. In this case n is less or equal to the maximum number of parents allowed by the user. The choice of 0 as cutoff for the interaction score enables to focus on the genes for which causality is supported by the data and/or the priors.

Note that our schema for combining scores computed from priors and genomic data (Equation (2)) allows users to easily infer networks using prior knowledge only ($w = 1$) or using genomic data only ($w = 0$), or a combination of priors and genomic data.

4 Predictive ability of the network model

In the last step of our network inference approach we use the network topology to build a predictive model. Such a predictive network model should be able to make one- and multi-step predictions. One-step prediction refers (i) to prediction the expression of a gene of interest from the expression of its parents only or (ii) to prediction of the effect of a gene perturbation on its direct children. Multi-step prediction refers to prediction of the state of some missing genes in the network or the direct and indirect effect of a perturbation.

With the inferred topology and genomic data in hands we can use different methods to make our network model predictive. If the data can be discretized in a biological meaningful way, we can compute conditional probability tables for all genes and use then to make one-step prediction, and subsequently compute the marginal probabilities to make multi-step predictions. If the data are continuous, we can fit local (linear) regression models for all genes, using parent genes as predictors, and then use these models to make one- and multi-step predictions.

In this work we fitted traditional linear regression model for each gene (target gene X_T) in the network by using only the parent genes ($\mathbf{X}_\mathcal{P}$) as predictors

$$X_T = \beta_0 + \sum_{X_i \in \mathbf{X}_\mathcal{P}} \beta_i X_i \tag{3}$$

where β are the coefficients estimated via the ordinary least squares method.

This novel approach is implemented in the *predictionet* package and is also integrated in the *Predictive Networks* web application (see "Analysis" panel). Even if the package is not yet available from Bioconductor[5], a public Git repository is accessible from `https://github.com/bhaibeka/predictionet`; any non-violent and constructive feedback is welcome. ☺

4.1 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 key parameters: .

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

categories If this parameter missing, `data` should be already discretized; otherwise 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 to discretize each of the variables in `data` matrix. If method='bayesnet' and `categories` is missing, `data` should contain categorical values and the number of categories will determine from the data.

perturbations Matrix of {0, 1} specifying whether a gene has been perturbed (e.g., knock-down, 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

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

instance). if priors counts are used the parameter `priors.count` should be TRUE so the priors are scaled accordingly.

predn Indices or names of variables to fit during network inference. If missing, all the variables will be used for network inference. Note that for bayesian network inference (method='bayesnet') this parameter is ignored and a network will be generated using all the variables.

priors.count `TRUE` if priors specified by the user are number of citations (count) for each interaction, `FALSE` if probabilities or any other weight 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)
```

The `netinf` function returns a list containing the names of the method used for network inference, the network topology and a list of local regression models.
You can infer a gene interaction network using the expO dataset `data.ras`, by equally balancing the importance of priors and data in the network inference process (`priors.weight` = 0.5).

To reduce the computational time, we will focus on the top 15 genes which are the most differentially expressed between RAS mutated and wild type cell lines:

```
> ## number of genes to select for the analysis
> genen <- 30
> ## select only the top genes
> goi <- dimnames(annot.ras)[[1]][order(abs(log2(annot.ras[ ,"fold.change"])), decreasing=
```

Now run the network inference on the reduced `expO.colon.ras` dataset:

```
> mynet <- netinf(data=data.ras[ ,goi], priors=pn.priors.counts[goi,goi], priors.count=TRU
```

Now let's display the topology of the network we just inferred (see Figure 3) using the `plot` function of the *network* package.
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*,... To enable interactive manipulation of the network

```
> ## network topology
> mytopoglobal <- mynet$topology
> library(network)
> xnet <- network(x=mytopoglobal, matrix.type="adjacency", directed=TRUE, loops=FALSE, ver
> mynetlayout <- plot.network(x=xnet, displayisolates=TRUE, displaylabels=TRUE, boxed.labe
```

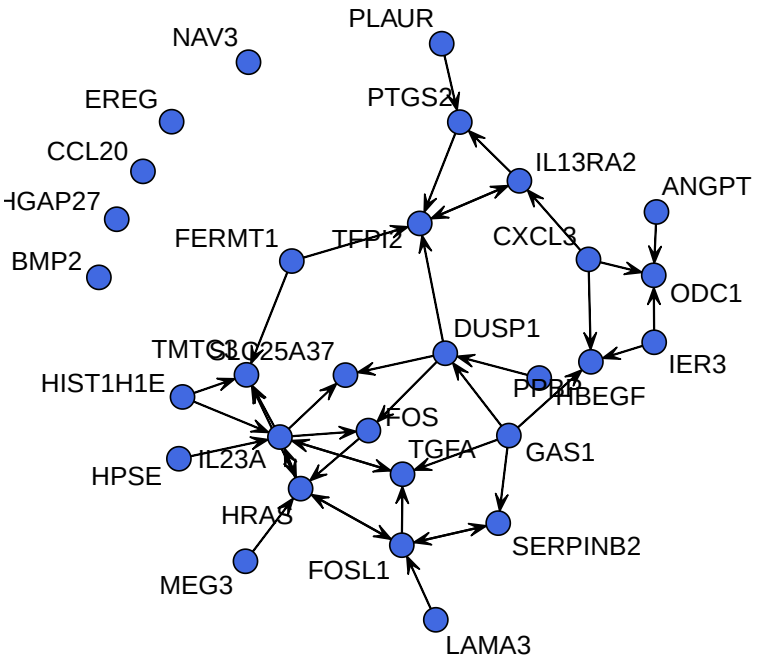

Figure 3: Directed graph representing the topology of the network inferred from priors and gene expression data.

topology we choose another approach that is to export the network object in `.gml` file which could be imported in external software such as Cytoscape[6] [Smoot et al., 2011]. We will see later how to use the function `netinf2gml` from the *predictionet* package to export a `.gml` file and import it to Cytoscape (see Section 4.4).

A quick look at the topology in Figure 3 allows us to identify IL13RA2 as a highly connected gene. 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 IL13RA2, in addition to be included in the RAS signature published in [Bild et al., 2006], is also part

[6] `http://www.cytoscape.org/`
[7] `http://compbio.dfci.harvard.edu/genesigdb/`

of other signatures published in colon, leukemia, ovarian and stomach cancers. Receptors for interleukin-13 (IL-13R) are over-expressed on several types of solid cancers; it is not only associated with colon cancer, but also with many other cancers including colon cancer [Niranjan et al., 2008].

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 4.

```
> ## preparing colors
> mycols <- c("blue", "green", "red")
> names(mycols) <- c("prior", "data", "both")
> myedgecols <- matrix("white", nrow=nrow(mytopoglobal), ncol=ncol(mytopoglobal), dimnames
> ## prior only
> myedgecols[mytopoglobal == 0 & pn.priors.counts[goi,goi] >= 1] <- mycols["prior"]
> ## data only
> myedgecols[mytopoglobal == 1 & pn.priors.counts[goi,goi] < 1] <- mycols["data"]
> ## both in priors and data
> myedgecols[mytopoglobal == 1 & pn.priors.counts[goi,goi] >= 1] <- mycols["both"]
```

In Figure 4 we can see that all 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 prediction score.

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=data.ras[ ,goi], categories=3, priors=pn.priors.counts[goi,go
```

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

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

```
List of 8
 $ method             : chr "regrnet"
 $ topology           : num [1:30, 1:30] 0 0 0 0 1 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
```

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

```
> mytopoglobal2 <- (myedgecols != "white") + 0
> ## network topology
> xnet2 <- network(x=mytopoglobal2, matrix.type="adjacency", directed=TRUE, loops=TRUE, ve
> plot.network(x=xnet2, displayisolates=TRUE, displaylabels=TRUE, boxed.labels=FALSE, labe
```

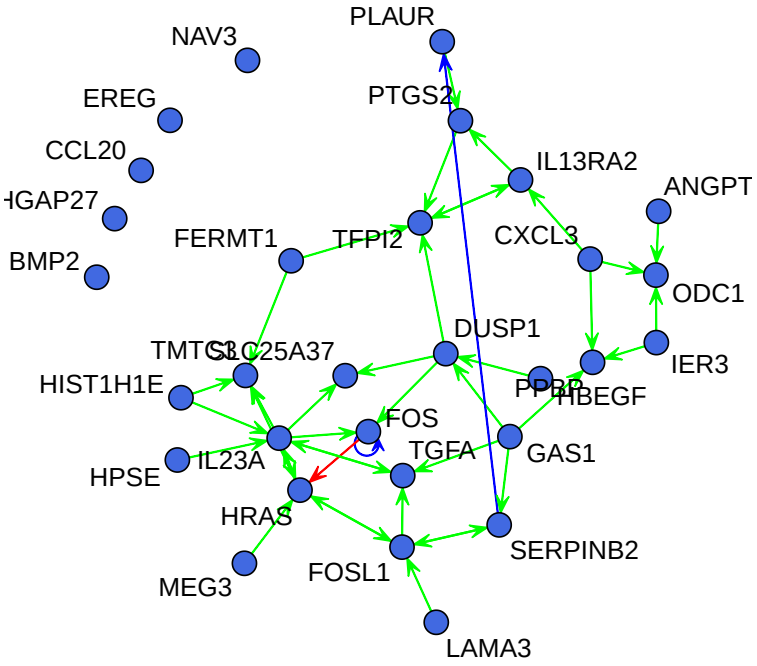

Figure 4: Network topology where the edges are colored according to their source of evidence: **blue** for edges supported by the priors only, **green** for edges supported by the data only, and **red** for edges supported by both the priors and the data.

```
 $ topology.cv        :List of 10
 $ prediction.score.cv:List of 3
 $ edge.stability     : num [1:30, 1:30] 0 0 0 0 1 0 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
 $ edge.stability.cv  : num [1:30, 1:30] 0 0 0 0 1 0 0.2 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
 $ edge.relevance     : num [1:30, 1:30] 0 0 0 0 0.542 ...
  ..- attr(*, "dimnames")=List of 2
 $ edge.relevance.cv  :List of 10
NULL
```

where

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.

topology.coeff coefficients of each local linear regression model fitted using the entire dataset.

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

topology.coeff coefficients of each local linear regression model fitted 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.

4.2 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 the frequency of its presence 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 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), dimn
> for(k in 1:length(mycols)) { myedgecols[myres.cv$edge.stability == names(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, ve
```

```
> plot.network(x=xnet3, displayisolates=TRUE, displaylabels=TRUE, boxed.labels=FALSE, labe
> 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="St
> 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)
```

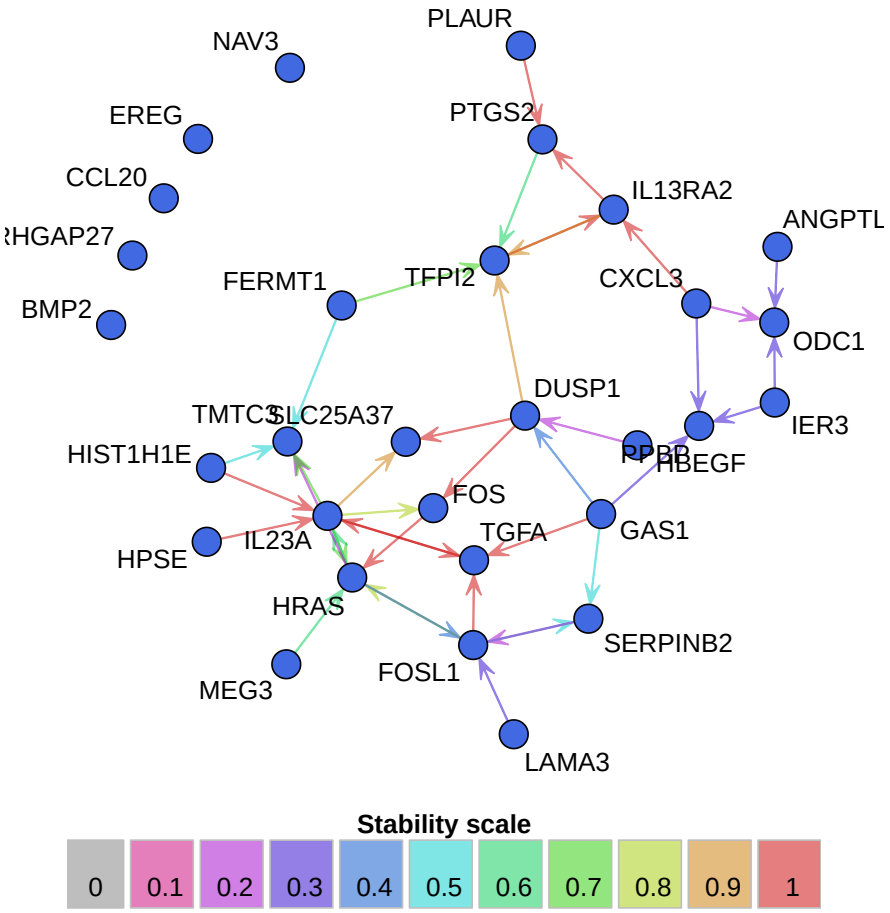

Figure 5: Network topology where the edges are colored according to their stability.

As can be seen, the inference of more than half of the edges is very stable, especially around the highly connected genes, that are IL13RA2, PLAUR, PTGS2, FOS. However the inference of the interactions with PPBP is unstable.

4.3 Gene-specific prediction score

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 6) and MCC (Figure 7). 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, ve
> plot.network(x=xnet3, displayisolates=TRUE, displaylabels=TRUE, boxed.labels=FALSE, labe
> 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
> 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)
```

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

```
> 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, ve
> plot.network(x=xnet3, displayisolates=TRUE, displaylabels=TRUE, boxed.labels=FALSE, labe
> 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="$M
> 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)
```

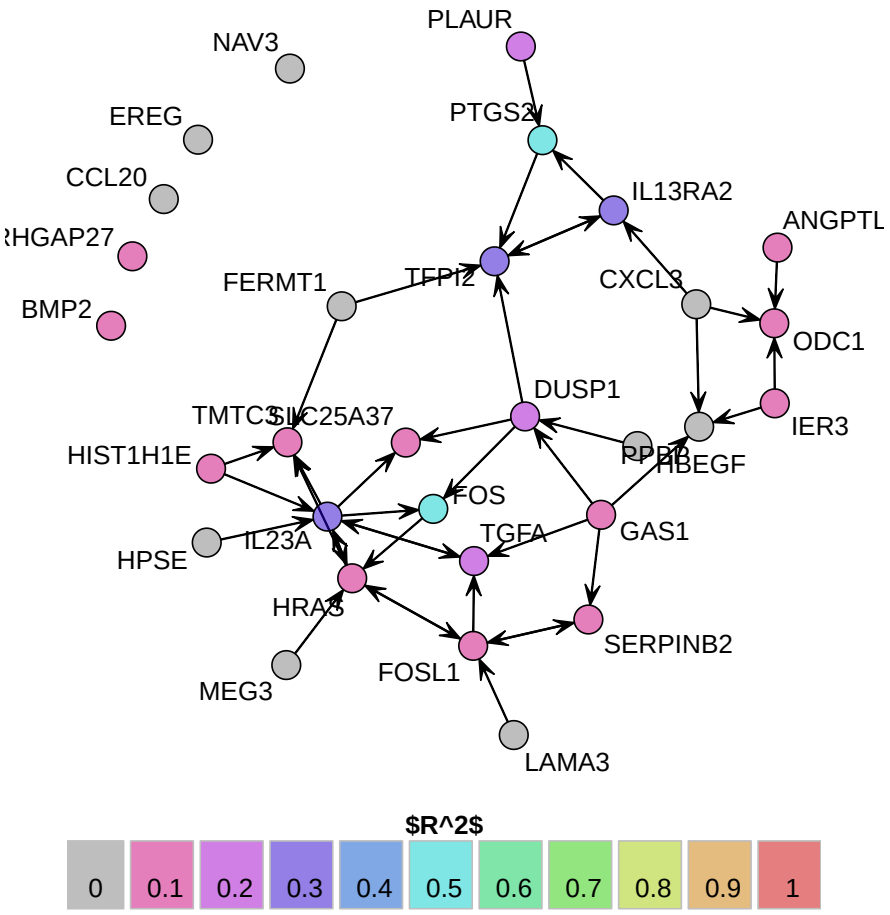

Figure 6: Network topology where the genes are colored according to the predictive performance of the network measure by R^2 in cross-validation.

As can be seen in Figures 6 and 7, the local predictive ability we get with the inferred network is quite low. This may be due to the small number of genes we consider in the RAS pathway what provide us with little information about causality of many genes (a majority of them are unconnected or have no parents). If we look at the genes with the best prediction we can see that the assessment of the network predictive ability is concordant with its stability analysis: the genes around PLAUR,, IL13RA and PTGS2 are quite well predicted while the genes connected with unstable edges are poorly predicted.

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 colon 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 colon cancer patients to validate the perfor-

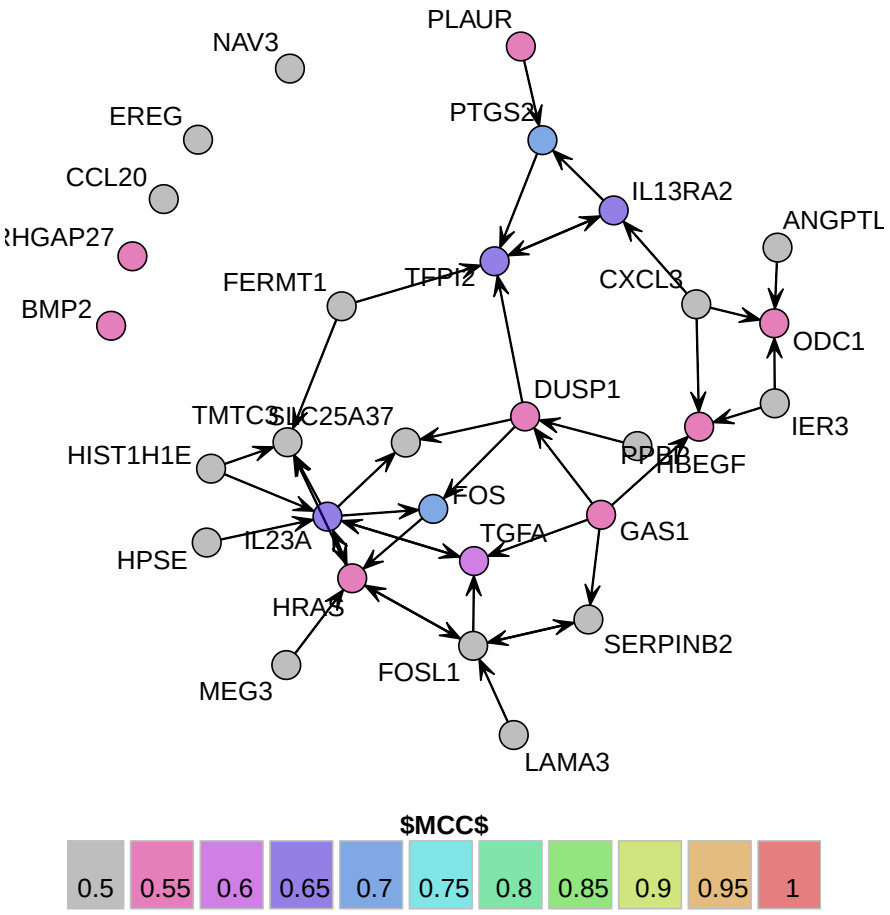

Figure 7: Network topology where the genes are colored according to the predictive performance of the network measured by MCC in cross-validation.

mance 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 piece of code and then display the network using the same code than before (see Figures 8 and 9 for the R^2 and MCC respectively).

```
> ## make the network model predictive
> mynet <- net2pred(net=mynet, data=data.ras[ ,goi], method="linear")
> ## compute predictions
> mynet.test.pred <- netinf.predict(net=mynet, data=data2.ras[ ,goi])
> ## performance estimation: R2
> mynet.test.r2 <- pred.score(data=data2.ras[ ,goi], pred=mynet.test.pred, method="r2")
> ## performance estimation: MCC
> mynet.test.mcc <- pred.score(data=data2.ras[ ,goi], categories=3, pred=mynet.test.pred,

> 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, ve
> plot.network(x=xnet3, displayisolates=TRUE, displaylabels=TRUE, boxed.labels=FALSE, labe
> 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
> 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)
```

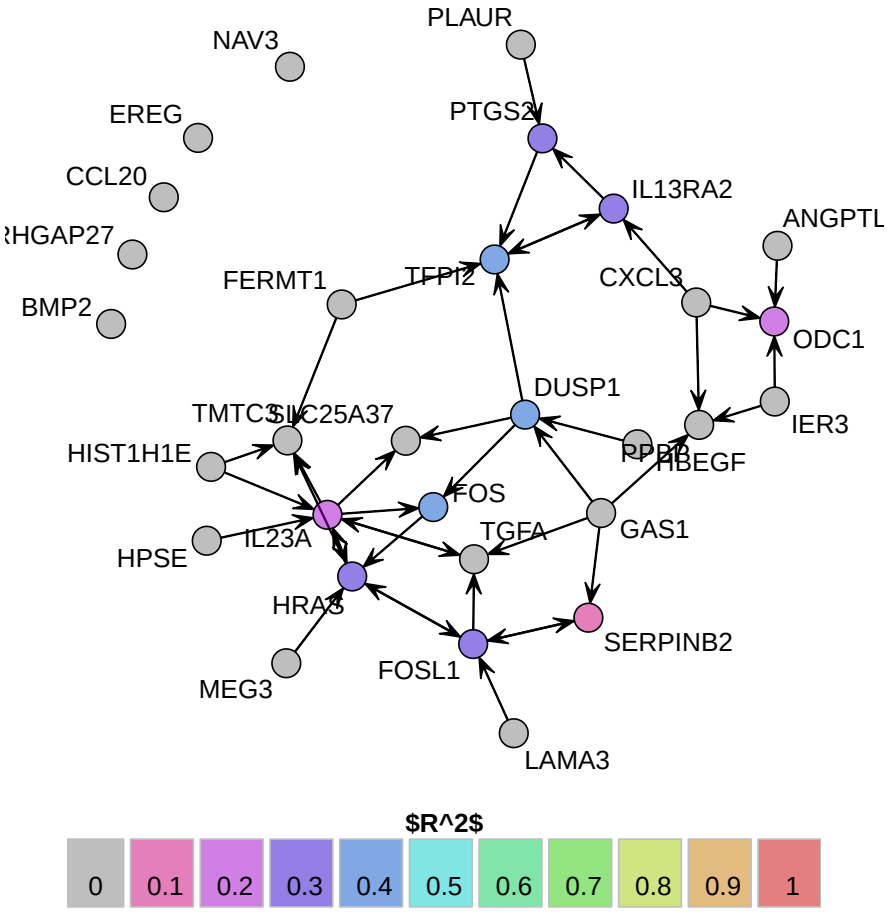

Figure 8: Network topology where the genes are colored according to the predictive performance of the network measured by R^2 in a fully independent dataset.

```
> 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, ve
> plot.network(x=xnet3, displayisolates=TRUE, displaylabels=TRUE, boxed.labels=FALSE, labe
> 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="$M
> 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)
```

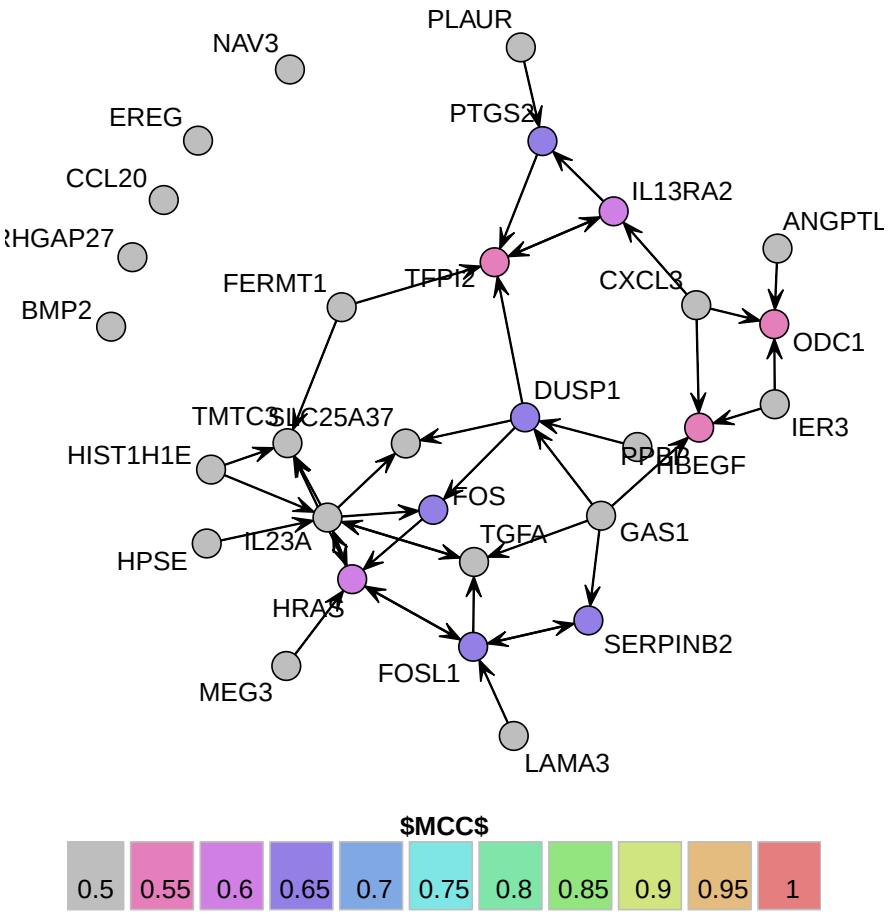

Figure 9: Network topology where the genes are colored according to the predictive performance of the network measured by MCC in a fully independent dataset.

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

4.4 Predictionet and Cytoscape

When large network are inferred (> 50 variables), any plot will be extremely busy and complicated to interpret, This is the reason why the function `netinf2gml` has been implemented in the *predictionet* package in order to allow to export any networks inferred from the `netinf` or `netinf.cv` functions in a file that is readable in many third-party software. Let's export a `gml` file containing all the information about the network model we just inferred from the `netinf.cv` function.

```
> netinf2gml(object=myres.cv, file="regrnet_exp0_cv")
```

A GML file called `regrnet_exp0_cv.gml` and a Vizmap Property file `egrnet_exp0_cv.props` are then created in the working directory, You can import the GML into Cytoscape and load the corresponding Vizmap property file. You can move vertices, filter them based on prediction scores, color the edges based on stability, and so on.

Figure 10: Screenshot of a network inferred by *predictionet* and imported into Cytoscape with the Vizmap Property file *predictionet_vizmap2*. Size of each vertex (gene) is proportional to their prediction score; color of the edges report the edge-specific stability where gray → green → orange → red colors represent edges with low to high stability.

5 Comparison of network inference with respect to the priors

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 prediction scores (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=data.ras[ ,goi], categories=3, priors=pn.priors.counts[go
> ## priors.weight 0.25
> myres.cv.pw025 <- netinf.cv(data=data.ras[ ,goi], categories=3, priors=pn.priors.counts[
> ## priors.weight 0.5
> myres.cv.pw050 <- myres.cv
> ## priors.weight 0.75
> myres.cv.pw075 <- netinf.cv(data=data.ras[ ,goi], categories=3, priors=pn.priors.counts[
> ## priors.weight 0
> myres.cv.pw1 <- netinf.cv(data=data.ras[ ,goi], categories=3, priors=pn.priors.counts[go
```

Now let's display the topology of the networks we just inferred by varying the weight we put on the priors (see Figure 11).

```
> def.par <- par(no.readonly=TRUE)
> layout(mat=matrix(1:4, nrow=2, ncol=2, byrow=TRUE))
> ## priors.weight 0
> mytopot <- myres.cv.pw0$topology
> xnett <- network(x=mytopot, matrix.type="adjacency", directed=TRUE, loops=FALSE, vertex.
> plot.network(x=xnett, displayisolates=TRUE, displaylabels=FALSE, boxed.labels=FALSE, lab
> ## priors.weight 0.25
> mytopot <- myres.cv.pw025$topology
> xnett <- network(x=mytopot, matrix.type="adjacency", directed=TRUE, loops=FALSE, vertex.
> plot.network(x=xnett, displayisolates=TRUE, displaylabels=FALSE, boxed.labels=FALSE, lab
> ## priors.weight 0.75
> mytopot <- myres.cv.pw075$topology
> xnett <- network(x=mytopot, matrix.type="adjacency", directed=TRUE, loops=FALSE, vertex.
> plot.network(x=xnett, displayisolates=TRUE, displaylabels=FALSE, boxed.labels=FALSE, lab
> ## priors.weight 1
> mytopot <- myres.cv.pw1$topology
> xnett <- network(x=mytopot, matrix.type="adjacency", directed=TRUE, loops=FALSE, vertex.
> plot.network(x=xnett, displayisolates=TRUE, displaylabels=FALSE, boxed.labels=FALSE, lab
> par(def.par)
```

As can be seen in Figure 11, 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 prediction scores

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 12). 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 gain in stability when the priors are used as compared to networks inferred from data alone.

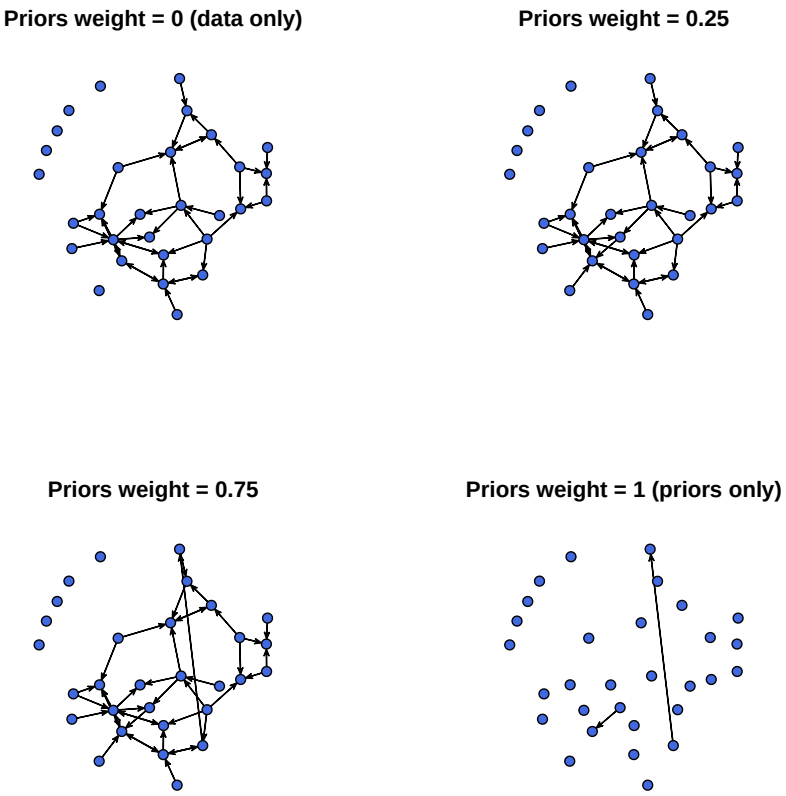

Figure 11: Directed graph representing the topology of the networks inferred from priors and gene expression data with varying prior weight.

5.2 Comparison of gene-specific prediction score

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 13), suggests that a combination of both data and priors yield better predictive ability.
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 statstically differe
> print(friedman.test(y=myr2.cv.pw))
```

[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.

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

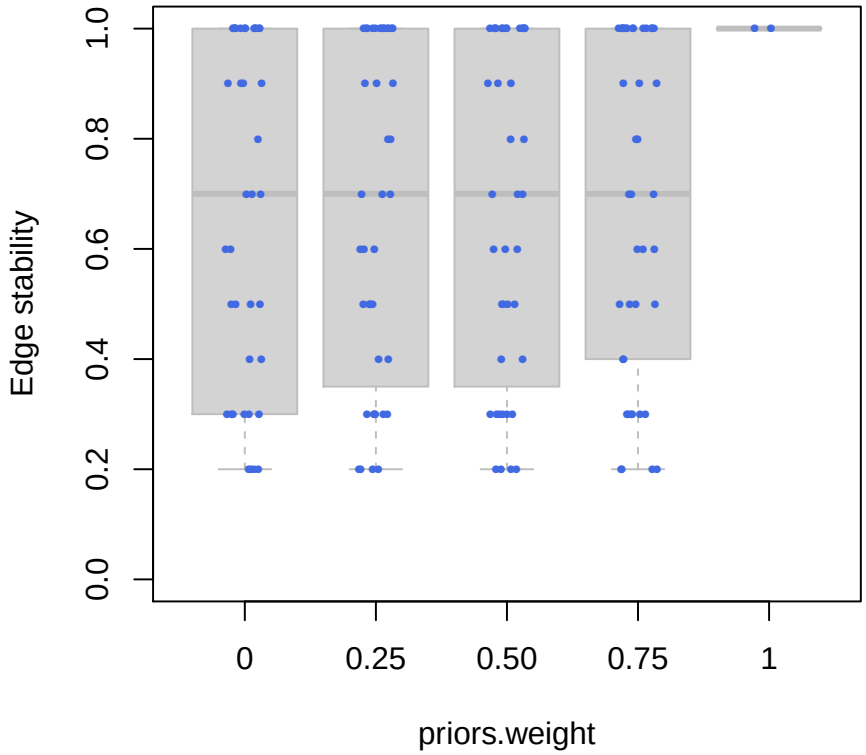

Figure 12: Boxplot of the edge-specific stability computed in cross-validation, for network models with different prior weight.

```
        Friedman rank sum test

data:  myr2.cv.pw
Friedman chi-squared = 90.725, df = 4, p-value < 2.2e-16

> ## Pairwise Wilcoxon Rank Sum tests
> print(pairwise.wilcox.test(x=as.vector(myr2.cv.pw), g=gg, paired=TRUE, exact=FALSE, alte

        Pairwise comparisons using Wilcoxon signed rank test

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

```
> 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), app
> 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
> points(x=jitter(x=rep(1:ncol(myr2.cv.pw), times=nrow(myr2.cv.pw)), amount=0.15), y=as.ve
```

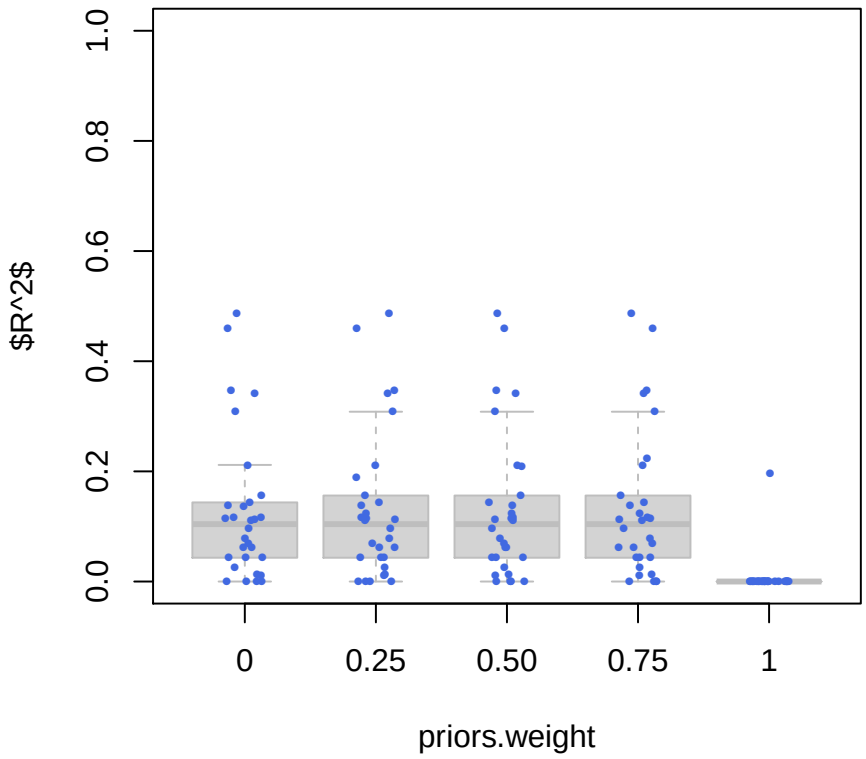

Figure 13: Boxplot of the gene-specific R^2 scores computed in cross-validation, for network models with different prior weight.

```
     0       0.25    0.50    0.75
0.25 1.00000 -       -       -
0.50 1.00000 1.00000 -       -
0.75 1.00000 1.00000 1.00000 -
1    0.00022 8.8e-05 8.8e-05 8.8e-05

P value adjustment method: holm
```

Similar conclusions can be drawn by computing the MCC.

6 Comparison of network inference with respect to the training data

To identify the parts of the networks where interactions are well supported by the data we estimated in cross-validation edge-specific stabilities and gene-specific prediction scores of a given network model. Ultimately these statistics should help us identify the parts of the network that are generalizable, whose inference does not strongly depend on the dataset used as training set.

Inference of generalizable network models is a challenging task because of the various biases that are inevitably present in datasets generated from different populations of cancer patients and in different labs. Therefore the resulting hidden batch effects might dramatically affect the inference of a network model.

Let's use `exp0.colon.ras` dataset as training set to infer our network model (use data only for inference, `priors.weight`=0). Now we will use our second dataset, `jorissen.colon.ras`, to infer another network model (use data only for inference, `priors.weight`=0) and compare them to test whether the network inference strongly depend on the training set or not.

First infer a network from `exp0.colon.ras` and `jerissen.colon.ras` datasets separately.

```
> ## exp0
> myres21.cv <- netinf.cv(data=data.ras[ ,goi], categories=3, priors=priors2.ras[goi,goi],
> ## jorissen
> myres22.cv <- netinf.cv(data=data2.ras[ ,goi], categories=3, priors=priors2.ras[goi,goi]
```

Let's display the topologies of the networks inferred from these two datasets. As can be seen in Figure 14, approximately half of the interactions are inferred in both datasets.

We could now ask the question whether the interactions inferred from both datasets are also the ones with high stability and/or involve genes with high prediction scores. We first compared the stability of interactions which are inferred only in one dataset and those which are inferred in both datasets (Figure 15). As can be seen, the interactions common to both datasets tend to have a higher stability.

```
> def.par <- par(no.readonly=TRUE)
> layout(mat=matrix(1:2, nrow=1, ncol=2, byrow=TRUE))
> ## exp0
> stab2 <- list("specific"=myres21.cv$edge.stability[(topo1 == 1 & topo2 == 0)], "common"=
> wt <- wilcox.test(x=stab2$specific, y=stab2$common)
> boxplot(stab2, ylab="Stability", ylim=c(0, 1), xlab="", border="grey", col="lightgrey",
> points(x=jitter(x=rep(1:length(stab2), times=sapply(stab2, length)), amount=0.15), y=unl
> ## jorissen
> stab2 <- list("specific"=myres22.cv$edge.stability[(topo1 == 0 & topo2 == 1)], "common"=
> wt <- wilcox.test(x=stab2$specific, y=stab2$common)
> boxplot(stab2, ylab="Stability", ylim=c(0, 1), xlab="", border="grey", col="lightgrey",
> points(x=jitter(x=rep(1:length(stab2), times=sapply(stab2, length)), amount=0.15), y=unl
> par(def.par)
```

We can also compare the prediction scores in the networks inferred from the two different datasets. As illustrated in Figure 16, there is a weak correlation between the prediction

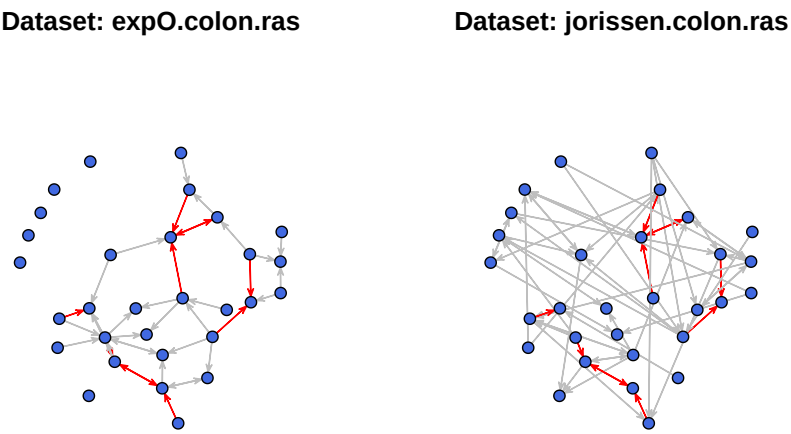

Figure 14: Directed graph representing the topology of the networks inferred from two different datasets, `expO.colon.ras` and `jerissen.colon.ras`. The orange edges represent interactions that are both supported by the data and the priors during network inference; the **red** edges represent the interactions that are supported only by the gene expression data and inferred in both networks; the gray edges represent interactions identified only in one dataset but not the other.

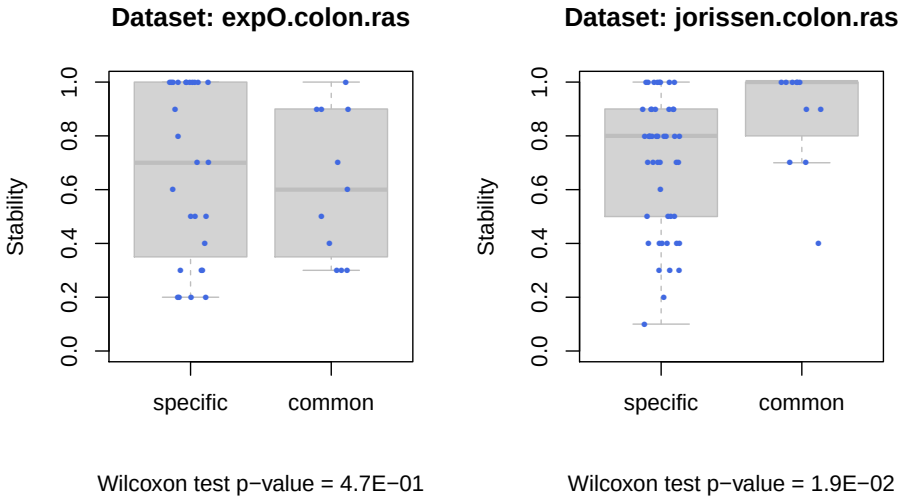

Figure 15: Boxplots reporting the stability of edges which are inferred only in one dataset and those which are inferred in both datasets.

scores (R^2) computed in the two datasets. So a gene with high prediction score in one dataset

may have a low score in another dataset, what makes this statistic not a good surrogate for generalization in network inference. This observation holds true for MCC.

```
> pred2 <- list("exp0"=apply(myres21.cv$prediction.score.cv$r2, 2, mean, na.rm=TRUE), "jor
> plot(x=pred2$exp0, y=pred2$jorissen, xlim=c(0, 0.6), ylim=c(0, 0.6), pch=16, col="royalb
> legend(x="bottomright", legend=sprintf("cor = %.2g", cor(pred2$exp0, pred2$jorissen, met
```

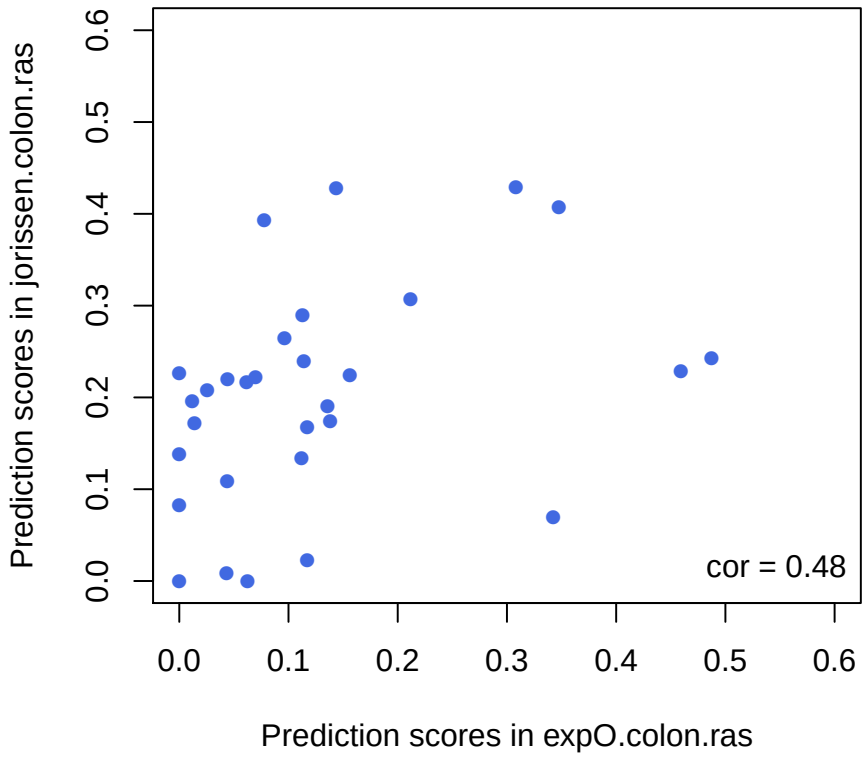

Figure 16: Prediction scores estimated in `expO.colon.ras` and `jorissen.colon.ras` and their corresponding Spearman correlation coefficient.

Session Info

- R version 3.2.2 (2015-08-14), `x86_64-pc-linux-gnu`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: catnet 1.14.8, igraph 1.0.1, network 1.13.0, predictionet 1.16.0
- Loaded via a namespace (and not attached): BiocGenerics 0.16.0, MASS 7.3-44, RBGL 1.46.0, graph 1.48.0, magrittr 1.5, parallel 3.2.2, penalized 0.9-45, splines 3.2.2, stats4 3.2.2, survival 2.38-3, tools 3.2.2

References

Learning Bayesian Networks. Prentica Hall, 2003.

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.

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. URL `http://www.sciencedirect.com/science/article/B6TYF-45BCRWV-2/2/390d83eb89efc5e0bc8de9e43e0681d9`.

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

Robert N Jorissen, Peter Gibbs, Michael Christie, Saurabh Prakash, Lara Lipton, Jayesh Desai, David Kerr, Lauri A Aaltonen, Diego Arango, Mogens Kruhøffer, Torben F Orntoft, Claus Lindbjerg Andersen, Mike Gruidl, Vidya P Kamath, Steven Eschrich, Timothy J Yeatman, and Oliver M Sieber. Metastasis-associated gene expression changes predict poor outcomes in patients with dukes stage b and c colorectal cancer. *Clin Cancer Res*, 15(24): 7642–7651, Dec 2009. doi: 10.1158/1078-0432.CCR-09-1431.

Matthew N McCall, Benjamin M Bolstad, and Rafael A Irizarry. Frozen robust multiarray analysis (frma). *Biostatistics*, 11(2):242–53, Apr 2010. doi: 10.1093/biostatistics/kxp059.

W J McGill. Multivariate information transmission. *Psychometrika*, 1954.

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.

V Niranjan, R Mahmood, A kalaivan, and Sudha S. Study of cancer gene in x-chromosome. *Journal of Theoretical and Applied Information Technology*, 4(1):31–55, 2008.

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.

C W Reuter, M A Morgan, and L Bergmann. Targeting the ras signaling pathway: a rational, mechanism-based treatment for hematologic malignancies? *Blood*, 96(5):1655–69, Sep 2000.

Michael E Smoot, Keiichiro Ono, Johannes Ruscheinski, Peng-Liang Wang, and Trey Ideker. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–2, Feb 2011. doi: 10.1093/bioinformatics/btq675.