Inference of regulatory networks from microarray data with R and the Bioconductor package qpgraph

Robert Castelo
Research Program on Biomedical Informatics, Department of Experimental and Health Sciences, Universitat Pompeu Fabra, and Institut Municipal d'Investigació Mèdica, Barcelona, Spain**.** Email: robert.castelo@upf.edu (corresponding author)

Alberto Roverato
Department of Statistical Science. Università di Bologna, Bologna, Italy
Email: alberto.roverato@unibo.it

(This is a preprint from a chapter of the following book currently in press: *Next Generation Microarray Bioinformatics*, J. Wang, A.C. Tan and T. Tian, Eds., Methods in Molecular Biology Series, Humana Press, USA, ISBN 978-1-61779-399-8.)

Abstract

Regulatory networks inferred from microarray data sets provide an estimated blueprint of the functional interactions taking place under the assayed experimental conditions. In each of these experiments, the gene expression pathway exerts a finely tuned control simultaneously over all genes relevant to the cellular state. This renders most pairs of those genes significantly correlated, and therefore, the challenge faced by every method that aims at inferring a molecular regulatory network from microarray data, lies in distinguishing direct from indirect interactions. A straightforward solution to this problem would be to move directly from bivariate to multivariate statistical approaches. However, the daunting dimension of typical microarray data sets, with a number of genes p several orders of magnitude larger than the number of samples n, precludes the application of standard multivariate techniques and confronts the biologist with sophisticated procedures that address this situation. We have introduced a new way to approach this problem in an intuitive manner, based on limited-order partial correlations, and in this chapter we illustrate this method through the R package `qpgraph`, which forms part of the Bioconductor project and is available at its website ***(1)***.

Key words: molecular regulatory network, microarray data, reverse engineering, network inference, non-rejection rate, qpgraph

1. Introduction

The genome-wide assay of gene expression by microarray instruments provides a high-throughput readout of the relative RNA concentration for a very large number of genes p across a typically much smaller number of experimental conditions n. This enables a fast systematic comparison of all expression profiles on a gene-by-gene basis by analysis techniques such as differential expression. However, the simultaneous assay of all genes embeds in the microarray data a pattern of correlations projected from the regulatory interactions forming part of the cellular state of the samples, and therefore, estimating this pattern from the data can aid in building a network model of the transcriptional regulatory interactions.

Many published solutions to this problem rely on pairwise measures of association based on bivariate statistics, such as Pearson correlation or mutual information ***(2)***. However, marginal pairwise associations cannot distinguish direct from indirect (that is, spurious) relationships and specific enhancements to this pairwise approach have been made to address this problem (see, for instance, ***(3)*** and ***(4)***).

A sensible approach is to try to apply multivariate statistical methods such as undirected Gaussian graphical modeling ***(5)*** and compute partial correlations which are a measure of association between two variables while controlling for the remaining ones. However, these methods require inverting the sample covariance matrix of the gene expression profiles and this is only possible when $n > p$ ***(6)***. This has led to the development of specific inferential procedures, which try to overcome the *small n and large p problem* by exploiting specific biological background knowledge on the structure of the network to be inferred. From this viewpoint, the most relevant feature of regulatory networks is that they are sparse, that is the direct regulatory interactions between genes represent a small proportion of the edges present in a fully connected network (see, for instance, ***(7)***). Statistical procedures for inference on sparse networks include, among others, a Bayesian approach with sparsity inducing prior ***(8)***, the lasso estimate of the inverse covariance matrix (see, among others, ***(9)*** and ***(10)***), the shrinkage estimate of the covariance matrix ***(11)*** and procedures based on limited-order partial correlations (see, for instance, ***(12)*** and ***(13)***).

In ***(14)*** a procedure is proposed for the statistical learning of sparse networks based on a quantity called the non-rejection rate. The computation of the non-rejection rate requires carrying out a large number of hypothesis tests involving limited-order partial correlations, nonetheless that procedure is not affected by the multiple testing problem. Furthermore, in ***(15)*** it is shown that averaging non-rejection rates obtained through different orders of the partial correlations is an effective strategy to release the user from making an educated guess on the most suitable order. In the same article, a method based on the concept of functional coherence is introduced, for the comparison of the functional relevance of different inferred networks and their regulatory modules. In the rest of this chapter we show how to apply this entire methodology by using the statistical software R and the Bioconductor package `qpgraph`.

2. Materials

2.1. The non-rejection rate

We represent the molecular regulatory network we want to infer by means of a mathematical object called a *graph*. A graph is a pair $G=(V, E)$, where $V=\{1,2, \ldots , p\}$ is a finite set of *vertices* and E is a subset of pairs of vertices, called the *edges* of G. In this context, vertices are genes and edges are direct regulatory interactions (see **Note 1**). Nevertheless, the graphs we consider here have no multiple edges and no loops;

furthermore, they are undirected so that both $(i,j) \in E$ and $(j, i) \in E$ are an equivalent way to write that the vertices i and j are linked by an edge. A basic feature of graphs is that they are visual objects. In the graphical representation, vertices may be depicted with circles while undirected edges are lines joining pairs of vertices. For example, the graph $G=(V, E)$ with $V=\{1, 2, 3\}$ and $E=\{(1, 2), (2, 3)\}$ can be represented as ①——②——③. A *path* in G from i to j is a sequence of vertices such that i and j are the first and last vertex of the sequence, respectively, and every vertex in the sequence is linked to the next vertex by an edge. The subset $Q \subseteq V$ is said to *separate* i from j if all paths from i to j have at least one vertex in Q. For instance, in the graph of the example above the sequence $(1, 2, 3)$ is a path between 1 and 3 whereas the sequence $(1, 3, 2)$ is not a path. Furthermore, the set $Q=\{2\}$ separates 1 from 3.

The random vector of gene expression profiles is indexed by the set V and denoted by $X_V=(X_1, X_2, \dots, X_p)^T$ and, furthermore, we denote by $\rho_{ij.V\setminus\{i,j\}}$ the *full-order partial correlation* between the genes i and j, that is the correlation coefficient between the two genes adjusted for all the remaining genes $V\setminus\{i, j\}$. We assume that X_V belongs to a Gaussian graphical model with graph $G=(V, E)$ and refer to **(5)** for a full account on these models. Here, we recall that in a Gaussian graphical model X_V is assumed to be multivariate normal and that the vertices i and j are not linked by an edge if and only if $\rho_{ij.V\setminus\{i,j\}}=0$. It follows that the sample version of full-order partial correlations plays a key role in statistical procedures for inferring the network structure from data. However, these quantities can be computed only if n is larger than p and this has precluded the application of standard techniques in the context of regulatory network inference from microarray data. On the other hand, if the edge between the genes i and j is missing from the graph then possibly a large number of limited-order partial correlations are equal to zero. More specifically, for a subset $Q \subset V\setminus\{i,j\}$ we denote by $\rho_{ij.Q}$ the *limited-order partial correlation*, that is the correlation coefficient between i and j adjusted for the genes in Q. It can be shown that if Q separates i and j in G, then $\rho_{ij.Q}$ is equal to zero. This is a useful result because the sample version of $\rho_{ij.Q}$ can be computed whenever $n > q+2$ and, if the distribution of X_V is *faithful* to G (see ***(14)*** and references therein), then $\rho_{ij.Q}=0$ also implies that the vertices i and j are not linked by an edge in G.

In sparse graphs one should expect a high degree of separation between vertices and therefore limited-order partial correlations are useful tools for inferring sparse molecular regulatory networks from data. There are, however, several difficulties related to the use of limited-order partial correlations because for every pair of genes i and j there are a huge number of potential subsets Q, and this leads to computational problems as well as to multiple testing problems. In ***(14)*** the authors propose to use a quantity based on partial correlations of order q that they call the non-rejection rate. The *non-rejection rate* for vertices i and j is denoted by $NRR(i,j\,|\,q)$ and it is the probability of not rejecting, on the basis of a suitable statistical test, the hypothesis that $\rho_{ij.Q}=0$ where Q is a subset of q genes randomly selected from $V\setminus\{i,j\}$. Hence, the non-rejection rate is a probability associated to every pair of vertices, genes in the context of this chapter, and takes values between zero and one, with larger values providing stronger evidence that an edge is not present in G. The procedure introduced in ***(15)*** amounts to estimating the non-rejection rate for every pair of vertices, ranking all the possible edges of the graph according to these values and then removing those edges whose non-rejection rate values are above a given threshold. Different methods for the choice of the threshold are discussed in the forthcoming sections where the graph inferred with this method will be called the *qp-graph*; we refer to ***(14)*** and ***(15)*** for technical details. Here we recall that the computation of the non-rejection rate requires the specification of a value q corresponding to the dimension of the potential separator, with q ranging from the value *1* to the value *n-3*. Obviously, a key question when using the non-rejection rate with microarray data is what value of *q* should be employed. We know that a larger value of *q* increases the probability that a randomly chosen subset *Q* separates *i* and *j*, but this could compromise the statistical power of the tests which depends on *n-q*. In ***(15)*** a simple and effective solution to this question was introduced and consists of averaging (taking the arithmetic mean), for each pair of genes, the estimates of the non-rejection rates for different values of *q* spanning its entire range from 1 to somewhere close to *n-3*. These authors also showed that the average non-rejection rate is more stable than the non-rejection rate, avoids having to specify a particular value of *q* and it behaves similarly to the non-rejection rate for connected pairs of vertices in the true underlying graph *G* (i.e., for directly interacting genes in the underlying molecular regulatory network). They also pointed out that the drawback of averaging is that a disconnected pair of vertices *(i,j)* in a graph *G* whose indirect relationship is mediated by a large number of other vertices, will be easier to identify with the non-rejection rate using a sufficiently large value of *q* than with the average non-rejection rate. However, in networks showing high degrees of modularity and sparseness the number of genes mediating indirect interactions should not be very large, and therefore, the average non-rejection rate should be working well, just as they observed in the empirical results reported in ***(15)***.

2.2. Functional coherence

A critical question when estimating a molecular regulatory network from data is to know the extent to which the inferred regulatory relationships reflect the functional organization of the system under the experimental conditions employed to generate the microarray data. The authors in ***(15)*** addressed this question using the Gene Ontology (GO) database ***(16)*** which provides structured functional annotations on genes for a large number of organisms including Escherichia coli (E. coli). The approach followed consists of assessing the functional coherence of every regulatory module within a given network. Assume a *regulatory module* is defined as a transcription factor and its set of regulated genes. The functional coherence of a regulatory module is estimated by relying on the observation that, for many transcription factor genes, their biological function, beyond regulating transcription, is related to the genes they regulate. Note that different regulatory modules can form part of a common pathway and thus share some more general functional annotations, which can lead to some degree of functional coherence between target genes and transcription factors of different modules. However, in ***(15)*** it is shown that for the case of E. coli data, the degree of functional coherence within a regulatory module is higher than between highly correlated but distinct modules. This observation allowed them to conclude that functional coherence constitutes an appealing measure for assessing the discriminative power between direct and indirect interactions and therefore can be employed as an independent measure of accuracy.

The way in which the authors in ***(15)*** estimated functional coherence is as follows. Using GO annotations, concretely those that refer to the biological process (BP) ontology, two GO graphs are built such that vertices are GO terms and (directed) links are GO relationships. One GO graph is induced (i.e., grown toward vertices representing more generic GO terms) from GO terms annotated on the transcription factor gene discarding those terms related to transcriptional regulation. The other GO graph is induced from GO terms overrepresented among the regulated genes in the estimated regulatory module which, to try to avoid spuriously enriched GO terms, we take it only into consideration if it contains at least 5 genes. These overrepresented GO terms can be found, for instance, by using the conditional hypergeometric test implemented in the Bioconductor package `GOstats` ***(17)*** on the

E. coli GO annotations from the `org.EcK12.eg.db` Bioconductor package. Finally, the level of functional coherence of the regulatory module is estimated as the degree of similarity between the two GO graphs, which in this case amounts to a comparison of the two corresponding subsets of vertices. The level of functional coherence of the entire network is determined by the distribution of the functional coherence values of all the regulatory modules for which this measure was calculated (see **Note 2**).

2.3. Escherichia coli microarray data

In this chapter we describe our procedure through the analysis of an E. coli microarray data set from ***(18)*** and deposited at the NCBI Gene Expression Omnibus (GEO) with accession GDS680. It contains 43 microarray hybridizations that monitor the response from E. coli during an oxygen shift targeting the *a priori* most relevant part of the network by using six strains with knockouts of key transcriptional regulators in the oxygen response (Δ*arcA*, Δ*appY*, Δ*fnr*, Δ*oxyR*, Δ*soxS* and the double knockout Δ*arcA*Δ*fnr*). We will infer a network starting from the full gene set of E. coli with *p=4,205* genes (see the following subsection for details on filtering steps).

2.4. Escherichia coli functional and microarray data processing

We downloaded the Release 6.1 from RegulonDB ***(19)*** formed by an initial set of 3,472 transcriptional regulatory relationships. We translated the Blattner IDs into Entrez IDs, discarded those interactions for which an Entrez ID was missing in any of the two genes and did the rest of the filtering using Entrez IDs. We filtered out those interactions corresponding to self-regulation and among those conforming to feedback-loop interactions we discarded arbitrarily one of the two interactions. Some interactions were duplicated due to a multiple mapping of some Blattner IDs to Entrez IDs, in that case we removed the duplicated interactions arbitrarily. We finally discarded interactions that did not map to genes in the array and were left with 3,283 interactions involving a total of 1,428 genes.

We have obtained RMA expression values for the data in ***(18)*** using the `rma()` function from the `affy` package in Bioconductor. We filtered out those genes, for which there was no Entrez ID and when two or more probesets were annotated under the same Entrez ID we kept the probeset with highest median expression level. These filtering steps left a total number of *p=4,205* probesets mapped one-to-one with E. coli Entrez genes.

3. Methods

3.1. Running the Bioconductor package qpgraph

The methodology briefly described in this chapter is implemented in the software called `qpgraph`, which is an add-on package for the statistical software R ***(20)***. However, unlike most other available software packages for R, which are deposited at the Comprehensive R Archive Network -CRAN- ***(21)***, the package `qpgraph` forms part of the Bioconductor project (see ***(22)*** and ***(1)***) and it is deposited in the Bioconductor website instead. The version of the software employed to illustrate this chapter runs over R 2.12 and thus forms part of Bioconductor package bundle version 2.7 (see **Note 3**). Among the packages that get installed by default with R and Bioconductor, `qpgraph` will automatically load some of them when calling certain functions but one of these, `Biobase`, should be explicitly loaded to manipulate microarray expression data through the `ExpressionSet` class of objects. Therefore, the initial sequence of commands to successfully start working with `qpgraph` through the example illustrated in this chapter is as folows:

```
> library(Biobase)
> library(qpgraph)
```

Additionally, we may consider the fact that most modern desktop computers come with four or more core processors and that it is relatively common to have access to a cluster facility with dozens, hundreds or perhaps thousands of processors scattered through an interconnected network of computer nodes. The `qpgraph` package can take advantage of such a multiprocessor hardware by performing some of the calculations in parallel. In order to enable this feature it is necessary to install the R packages `snow` and `rlecuyer` from the CRAN repository and load them prior to using the `qpgraph` package. The specific type of cluster configuration that will be employed will depend on whether additional packages providing such a specific support are installed. For example if the package `Rmpi` is installed, then the cluster configuration will be that of an MPI cluster (see ***(23)*** and **Note 4** for details on this subject). Thus, if we want to take advantage of an available multiprocessor infrastructure we should additionally write the following commands:

```
> library(snow)
> library(rlecuyer)
```

Once these packages have been successfully loaded, to perform calculations in parallel it is necessary to provide an argument, called `clusterSize`, to the corresponding function indicating the number of processors that we wish to use. In this chapter we assume we can use 8 processors, which should allow the longest calculation illustrated in this chapter to finish in less than 15 minutes. During long calculations it is convenient to monitor their progress and this is possible in most of the functions from the `qpgraph` package if we set the argument `verbose=TRUE,` which by default is set to `FALSE`.

3.2. A quick tour through the qpgraph package

In this section we illustrate the minimal function calls in the `qpgraph` package that allow one to infer a molecular regulatory network from microarray data. We need first to load the data described in the previous section and which is included as an example data set in the `qpgraph` package.

```
> data(EcoliOxygen)
```

The previous command will load on our current R default environment two objects, one of them called `gds680.eset`, which is an object of the class `ExpressionSet` and contains the E. coli microarray data described in the previous section. We can see these objects in the workspace with the function `ls()` and figure out the dimension of this particular microarray data set with `dim()`, as follows:

```
> ls()
[1] "filtered.regulon6.1" "gds680.eset"
> dim(gds680.eset)
Features  Samples
    4205       43
```

When we have a microarray data set, either as an `ExpressionSet` object or simply as a matrix of numeric values, we can immediately proceed to estimate non-rejection rates with a *q*-order of, for instance, *q=3* with the function `qpNrr()`:

```
> nrr <- qpNrr(gds680.eset, q=3, clusterSize=8)
```

This function returns a symmetric matrix of non-rejection rate values with its diagonal entries set to `NA`. Using this matrix as input to the function `qpGraph()` we can directly infer a molecular regulatory network by setting a non-rejection rate cutoff value above which edges are removed from an initial fully connected graph. The selection of this cutoff could be done, for instance, on the basis of targeting a graph of specific density which can be examined by calling first the function `qpGraphDensity()`, whose result is displayed in Figure 1a and from which we consider retrieving a graph of 7% density by using a 0.1 cutoff value:

```
> qpGraphDensity(nrr, title="", breaks=10)
> g <- qpGraph(nrr, threshold=0.1, return.type="graphNEL")
> g
A graphNEL graph with undirected edges
Number of Nodes = 4205
Number of Edges = 644036
```

By default, the `qpGraph()` function returns an adjacency matrix but, by setting `return.type="graphNEL"` we obtain a `graphNEL-class` object as a result, which, as we shall see later, is amenable for processing with functions from the Bioconductor packages `graph` and `Rgraphviz`. We can conclude this quick tour through the main cycle of the task of inferring a network from microarray data by showing how we can extract a ranking of the strongest edges in the network with the function `qpTopPairs()`:

```
> qpTopPairs(nrr)
       i      j x
1 947758 947761 0
2 948517 948512 0
3 944834 944794 0
4 948517 944797 0
5 948512 944797 0
6 945112 945108 0
```

where the first two columns, called `i` and `j`, correspond to the identifiers of the pair of variables and the third column `x` corresponds, in this case, to non-rejection rate values. An immediate question is whether the value of *q=3* was appropriate for this data set and while we may try to find an answer by exploring the estimated non-rejection rate values in a number of ways described in ***(14)***, an easy solution introduced in ***(15)*** consists of estimating the so-called average non-rejection rates whose corresponding function, `qpAvgNrr()`, is called in an analogous way to `qpNrr()` but without the need to specify a value for *q*.

In ***(15)*** a comparison of this procedure with other widely used techniques is carried out. Here, we restrict the comparison to a simple procedure based on sample Pearson correlation coefficients and, furthermore, to the worst performing strategy which consists of setting association values uniformly at random to every pair of genes (which we shall informally call the *random association method*) leading to a completely random ranking of the edges of the graph. All these quantities can be computed using two functions available also through the `qpgraph` package:

```
> allGenes <-  featureNames(gds680.eset)
> pcc <- qpPCC(gds680.eset)
> rndcor <- qpUnifRndAssociation(length(allGenes), allGenes)
```

3.3. Avoiding unnecessary calculations

We saw before that as part of the `EcoliOxygen` example data set included in the `qpgraph` package, there was an object called `filtered.regulon6.1`. This object is a `data.frame` and contains pairs of genes corresponding to curated transcriptional regulatory relationships from E. coli retrieved from the 6.1 version of the RegulonDB database. Each of these relationships indicates that one transcription factor gene activates or represses the transcription of the other target gene. If we are interested in just this kind of transcriptional regulatory interactions, i.e., associations involving at least one transcription factor gene, we can substantially speed up calculations by restricting them to those pairs of genes suitable to form such an association. In order to illustrate this feature, we start here by extracting from the RegulonDB data what genes form the subset of transcription factors:

```
> regulonTFgenes <- unique(filtered.regulon6.1[, "EgID_TF"])
```

In general, this kind of functional information about genes is available for many organisms through different on-line databases ***(24)***. Once we have a list of transcription factor genes, restricting the pairs that include at least one of them can be done through the arguments `pairup.i` and `pairup.j` in both functions, `qpNrr()` and `qpAvgNrr()`. We use here the latter to estimate average non-rejection rates that will help us to infer a transcriptional regulatory network without having to specify a particular *q*-order value. Since the estimation of non-rejection rates is carried out by means of a Monte Carlo sampling procedure, in order to allow the reader to reproduce the exact numbers shown here we will set a specific seed to the random number generator before estimating average non-rejection rates.

```
> set.seed(123)
> avgnrr <- qpAvgNrr(gds680.eset, pairup.i=regulonTFgenes,
                     pairup.j=allGenes, clusterSize=8)
```

The default settings for the function `qpAvgNrr()` employ 4 *q*-values uniformly distributed along the available range of *q* values. In this example, these correspond to *q={1, 11, 21, 31}*. However, we can change this default setting by using the argument `qOrders`.

3.4. Network accuracy with respect to a gold-standard

E. coli is the free-living organism with the largest fraction of its transcriptional regulatory network supported by some sort of experimental evidence. As a result of an effort in combining all this evidence the database RegulonDB ***(19)*** provides a curated set of transcription factor and target gene relationships that we can use as a gold-standard to, as we shall see later, calibrate a nominal precision or recall at which we want to infer the network or compare the performance of different parameters and network inference methods. This performance is assessed in terms of *precision-recall curves*.

Every network inference method that we consider here provides a ranking of the edges of the fully connected graph, that is, of all possible interactions. Then a threshold is chosen and this leads to a partition of the set of all edges into a set of *predicted edges* and a set of *missing edges*. On the other hand, the set of RegulonDB interactions are a subset of the set of all possible interactions and a predicted edge that belongs to the set of RegulonDB interactions is called a *true positive*. Following the conventions from ***(25)***, when using RegulonDB interactions for comparison the *recall* (also known as sensitivity) is defined as the fraction of true positives in the set of RegulonDB interactions and the *precision* (also known as positive predictive value) is defined as the number of true positives over the number of predicted edges whose genes belong to at least one transcription factor and target gene relationship in RegulonDB. For a given network inference method, the precision-recall curve is constructed by plotting the precision against the recall for a wide range of different threshold values. In the E. coli dataset we analyze, precision-recall curves should be calculated on the subset of 1,428 genes forming the 3,283 RegulonDB interactions and this can be achieved with the `qpgraph` package through the function `qpPrecisionRecall()` as follows:

```
> regulonGenes <- unique(c(filtered.regulon6.1$EgID_TF,
                           filtered.regulon6.1$EgID_TG))
> avgnrr.pr <- qpPrecisionRecall(avgnrr,
                 refGraph=filtered.regulon6.1[, c("EgID_TF", "EgID_TG")],
                 decreasing=FALSE,
                 pairup.i=regulonTFgenes,
                 pairup.j=regulonGenes,
                 recallSteps=c(seq(0,0.1,0.005),
                               seq(0.1,1.0,0.1)))
```

The previous lines calculate the precision-recall curve for the ranking derived from the average non-rejection rate values. The calculation of these curves for the other two rankings derived from Pearson coefficients and uniformly random association values would require replacing the first argument by the corresponding matrix of measurements in absolute value since these two methods provide values ranging from -1 to +1. We can plot the resulting precision-recall curve for the average non-rejection rate stored in `avgnrr.pr` as follows:

```
> plot(100*avgnrr.pl[, 1:2], type="b", pch=19, xlim=c(0, 6),
       xlab="Recall (% RegulonDB interactions)", ylab="Precision (%)")
```

In Figure 1b this plot is shown jointly with the other calculated curves, where the comparison of the average non-rejection rate (labeled qp-graph) with the other methods yields up to 40% improvement in precision with respect to using absolute Pearson correlation coefficients and observe that for precision levels between 50% and 80% the qp-graph method doubles the recall. We shall see later that this has an important impact when targeting a network of a reasonable nominal precision in such a data set with *p=4,205* and *n=43*.

[FIGURE 1 NEAR HERE]

3.5. Inference of molecular regulatory networks of specific size

Given a measure of association for every pair of genes of interest, the most straightforward way to infer a network is to select a number of top-scoring interactions that conform a resulting network of a specific size that we choose. We showed before such a strategy by looking at the graph density as a function of threshold, however, we can also extract a network of specific size by using the argument `topPairs` in the call to the `qpGraph()` and `qpAnyGraph()` functions where the call for the random association values would be analogous to the one of Pearson correlations.

```
> qpg1000sze <- qpGraph(avgnrr, threshold=NULL, topPairs=1000)
> pcc1000sze <- qpAnyGraph(abs(pcc$R), threshold=NULL,
                  topPairs=1000, decreasing=TRUE,
                  pairup.i=regulonTFgenes,
                  pairup.j=allGenes)
```

In the example above we are extracting networks formed by the top-scoring 1,000 interactions.

3.6. Inference of molecular regulatory networks at nominal precision and recall levels

When a gold-standard network is available we can infer a specific molecular regulatory network using a nominal precision and/or using a nominal recall. This is implemented in the `qpgraph` package by calling first the function `qpPRscoreThreshold()` which, given a precision-recall curve calculated with `qpPrecisionRecall()`, will calculate for us the score that attains the desired nominal level. In this particular example, and considering the precision-recall curve of Figure 1b, we will employ nominal values of 50% precision and 3% recall:

```
> avgnrr.50p.thr <- qpPRscoreThreshold(avgnrr.pr, level=0.50,
                      recall.level=FALSE, max.score=0)
> avgnrr.3r.thr <- qpPRscoreThreshold(avgnrr.pr, level=0.03,
                     recall.level=TRUE, max.score=0)
```

where the thresholds for the other methods would be analogously calculated replacing the first argument by the object storing the corresponding curve returned by `qpPrecisionRecall()`.

Next, we apply these nominal precision and recall thresholds to obtain the networks by using the functions `qpGraph()` for the average non-rejection rate and `qpAnyGraph()` for any other type of association measure, here illustrated only with Pearson correlation coefficients:

```
> qpg50pre <- qpGraph(avgnrr, avgnrr.50p.thr)
> pcc50pre <- qpAnyGraph(abs(pcc$R), abspcc.50p.thr,
                 remove="below", pairup.i=regulonTFgenes,
                 pairup.j=allGenes)
> qpg3rec <- qpGraph(avgnrr, avgnrr.3r.thr)
> pcc3rec <- qpAnyGraph(abs(pcc$R), abspcc.3r.thr,
                remove="below", pairup.i=regulonTFgenes,
                pairup.j=allGenes)
```

3.7. Estimation of functional coherence

In order to estimate functional coherence we need to install a Bioconductor package with GO functional annotations associated to the feature names (genes, probes, etc.) of the microarray data. For this example, we require the E. coli GO annotations stored in the package `org.EcK12.eg.db`. It will be also necessary to have installed the `GOstats` package to enable the GO enrichment analysis. The function `qpFunctionalCoherence()` will allow us to estimate functional coherence values as we illustrate here below for the case of the nominal 50%-precision network obtained with the qp-graph method. The estimation for the other networks would require replacing only the first argument by the object storing the corresponding network:

```
> library(GOstats)
> library(org.EcK12.eg.db)
> qpg50preFC <- qpFunctionalCoherence(qpg50pre,
                  TFgenes=regulonTFgenes,
                  chip="org.EcK12.eg.db",
                  minRMsize=5, clusterSize=8)
```

This function returns a `list` object storing the transcriptional network and the values of functional coherence for each regulatory module. These values can be examined by means of a boxplot as follows:

```
> boxplot(qpg50preFC$functionalCoherenceValues,
          col=grey(0.50), ylim=c(0,1), ylab="Functional coherence")
```

In Figure 2 we see the boxplots for the functional coherence values of all networks obtained from each method and selection strategy. Through the three different strategies, the networks obtained with the qp-graph method provide distributions of functional coherence with mean and median values larger than those obtained from networks built with Pearson correlation or simply at random.

[FIGURE 2 NEAR HERE]

3.8. The 50%-precision qp-graph regulatory network

We are going to examine in detail the 50%-precision qp-graph transcriptional regulatory network. A quick glance at the pairs with strongest average non-rejection rates including the functional coherence values of their regulatory modules within this 50%-precision network can be obtained with the function `qpTopPairs()` as follows:

```
> qpTopPairs(measurementsMatrix=avgnrr, refGraph=qpg50pre,
             pairup.i=regulonTFgenes, pairup.j=allGenes,
             annotation="org.EcK12.eg.db", fcOutput=qpg50preFC,
             fcOutput.na.rm=TRUE)
       i      j iSymbol jSymbol    x funCoherence
1 948797 945585    appY    appC 0.01         0.33
2 947466 945938    glcC    mhpR 0.01         0.14
3 945938 946255    mhpR    astC 0.01         0.71
4 947466 948336    glcC    fadB 0.02         0.14
5 948797 947547    appY    appB 0.02         0.33
6 948797 946206    appY    appA 0.02         0.33
```

The previous function call admits also a `file` argument that would allow us to store these information as a tab-separated column text file, thus more amenable for automatic processing when combined with the argument `n=Inf` since by default this is set to a limited number (`n=6`) of pairs being reported.

For many other types of analysis, it is useful to store the network as an object of the `graphNEL` class, which is defined in the `graph` package. This is obtained by calling the `qpGraph()` function setting properly the argument `return.type` as follows:

```
> g <- qpGraph(avgnrr, threshold=avgnrr.50p.thr, return.type="graphNEL")
> g
A graphNEL graph with undirected edges
Number of Nodes = 147
Number of Edges = 120
```

As we see from the object's description, the 50%-precision qp-graph network consists of 120 transcriptional regulatory relationships involving 147 different genes. A GO enrichment analysis on this subset of genes can give us insights into the main molecular processes related to the assayed conditions. Such an analysis can be performed by means of a conditional hypergeometric test using the Bioconductor package `GOstats` as follows:

```
> goHypGparams <- new("GOHyperGParams",
                      geneIds=nodes(g),
                      universeGeneIds=allGenes,
                      annotation="org.EcK12.eg.db", ontology="BP",
                      pvalueCutoff=0.05, conditional=TRUE,
                      testDirection="over")
> goHypGcond <- hyperGTest(goHypGparams)
```

where the object `goHypGcond` stores the result of the analysis which can be examined in R through the `summary()` function whose output is displayed in Table 1. The GO terms enriched by the subset of 147 genes reflect three broad functional categories one being transcription, which is the most enriched but it is also probably a byproduct of the network models themselves that are anchored on transcription factor genes. The other two are metabolism and response to an external stimulus, which are central among the biological processes that are triggered by an oxygen shift. Particularly related to this, is the fatty acid oxidation process since fatty acid metabolism is crucial to allow the cell to adapt quickly to environmental changes and allows E. coli to grow under anaerobic conditions ***(26)***.

[TABLE 1 NEAR HERE]

Finally, using the `graphNEL` representation of our network stored in the variable g and the function `connComp()`from the `graph` package we can easily look up the distribution of sizes of the connected components:

```
> table(sapply(connComp(g), length))

 2  3  4  6  8  9 17 19
24  6  4  2  1  1  1  1
```

and observe that two of them, formed by 17 and 19 genes, are distinctively larger than the rest, thus corresponding to the more complex part of the network. In order to examine in more detail these two subnetworks we can plot them using the Bioconductor package `Rgraphviz` (see **Note 5**) and calling the function `qpPlotNetwork()` which will output Figure 3a:

```
> library(Rgraphviz)
> qpPlotNetwork(g, minimumSizeConnComp=17, pairup.i=regulonTFgenes,
               pairup.j=allGenes, annotation="org.EcK12.eg.db")
```

Often the visualization of many interacting genes is dificult to interpret as, for instance in this case, the module regulated by *mhpR*. We can also visualize the part of the network connected to *mhpR* by using the arguments `vertexSubset` and `boundary` as follows and obtain the result shown in Figure 3b:

```
> g_mhpR <- qpPlotNetwork(g, vertexSubset="mhpR", boundary=TRUE,
                          pairup.i=regulonTFgenes, pairup.j=allGenes,
                          annotation="org.EcK12.eg.db")
```

Note that we have assigned the result of this function to a variable named `g_mhpR`. This will store the graph we have just visualized into this variable as a `graphNEL` object and can be useful to extract the list of edges forming this subnetwork again through the function `qpTopPairs()`:

```
> qpTopPairs(measurementsMatrix=avgnrr[nodes(g_mhpR), nodes(g_mhpR)],
             refGraph=g_mhpR, pairup.i=regulonTFgenes, pairup.j=allGenes,
             annotation="org.EcK12.eg.db", fcOutput=qpg50preFC)
       i      j iSymbol jSymbol    x funCoherence
1 945938 947466    mhpR    glcC 0.01         0.71
2 945938 946255    mhpR    astC 0.01         0.71
3 947466 948336    glcC    fadB 0.02         0.14
4 945938 944954    mhpR    mhpC 0.02         0.71
5 947466 946255    glcC    astC 0.02         0.14
6 945938 948823    mhpR    fadI 0.02         0.71
```

This last step allows us to see that the two strongest associations occur within the *mhpR* regulatory module, which also has a very high value of functional coherence, thus constituting two promising candidates for a follow up study.

[FIGURE 3 NEAR HERE]

Notes

1. The underlying method assumes that it is estimating an undirected Gaussian graphical model, which is a well-defined statistical model. However, our biological interpretation of this model as a transcriptional regulatory network will lead us to discard interactions between genes where none of them is a transcription factor, and to put directions in the resulting graph from transcription factor genes to their putative targets. This provides us with a network model of the underlying transcriptional regulation, which does not have a statistical interpretation anymore in terms of, for instance, conditional independence, but which allows one to formulate educated guesses on plausible biological hypotheses.
2. The limited availability of GO functional annotations for genes outside well-studied model organisms can compromise a reliable estimation of functional coherence values.
3. Bioconductor release versions are synchronized with R software release versions and thus updated twice a year. It is always recommended to work with the latest versions. For a detailed explanation on how to install and update the R and Bioconductor software please visit the website ***(27)***.
4. The installation of the package `Rmpi` requires a prior installation and configuration of an MPI library. For further details on this issue please visit the website ***(28)***.
5. The installation of the package `Rgraphviz` requires a prior installation of the software `graphviz` available at the website ***(29)***.

Acknowledgments

This work is supported by the Spanish Ministerio de Ciencia e Innovación (MICINN) [TIN2008-00556 / TIN] and the ISCIII COMBIOMED Network [RD07/0067/0001]. R.C. is a research fellow of the "Ramon y Cajal" program from the Spanish MICINN [RYC-2006-000932]. A.R. acknowledges support from the Ministero dell'Università e della Ricerca [PRIN-2007AYHZWC].

References

1. http://www.bioconductor.org
2. Butte, A. J., Tamayo, P., Slonim, D., et al. (2000) Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks., *Proc Natl Acad Sci U S A 97*, 12182-12186.
3. Basso, K., Margolin, A. A., Stolovitzky, G., et al. (2005) Reverse engineering of regulatory networks in human B cells., *Nat Genet 37*, 382-390.
4. Faith, J. J., Hayete, B., Thaden, J. T., et al. (2007) Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles., *PLoS Biol 5*, e8.
5. Edwards, D. (2000) *Introduction to graphical modelling*, Springer New York.
6. Dykstra, R. L. (1970) Establishing Positive Definiteness of Sample Covariance Matrix, *Ann Math Statist 41*, 2153-2154.
7. Barabasi, A.-L., and Oltvai, Z. N. (2004) Network biology: understanding the cell's functional organization., *Nat Rev Genet 5*, 101-113.
8. Dobra, A., Hans, C., Jones, B., et al. (2004) Sparse graphical models for exploring gene expression data, *J. Multivariate. Anal. 90*, 196-212.
9. Friedman, J., Hastie, T., and Tibshirani, R. (2008) Sparse inverse covariance estimation with the graphical lasso, *Biostatistics 9*, 432-441.

10. Yuan, M., and Lin, Y. (2007) Model selection and estimation in the Gaussian graphical model, *Biometrika 94*, 19-35.

11. Schäfer, J., and Strimmer, K. (2005) A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics, *Stat. Appl. Genet. Mol. Biol. 4*, 1-32.

12. de la Fuente, A., Bing, N., Hoeschele, I., et al. (2004) Discovery of meaningful associations in genomic data using partial correlation coefficients *Bioinformatics 20*, 3565-3574.

13. Wille, A., and Bühlmann, P. (2006) Low-order conditional independence graphs for inferring genetic networks, *Stat. Appl. Genet. Mol. Biol. 5*, 1.

14. Castelo, R., and Roverato, A. (2006) A robust procedure for Gaussian graphical model search from microarray data with p larger than n, *J Mach Learn Res 7*, 2621-2650.

15. Castelo, R., and Roverato, A. (2009) Reverse engineering molecular regulatory networks from microarray data with qp-graphs, *J Comput Biol 16*, 213-227.

16. http://www.geneontology.org

17. Falcon, S., and Gentleman, R. (2007) Using GOstats to test gene lists for GO term association., *Bioinformatics 23*, 257-258.

18. Covert, M. W., Knight, E. M., Reed, J. L., et al. (2004) Integrating high-throughput and computational data elucidates bacterial networks., *Nature 429*, 92-96.

19. Gama-Castro, S., Jimenez-Jacinto, V., Peralta-Gil, M., et al. (2008) RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation., *Nucleic Acids Res 36*, D120-124.

20. http://www.r-project.org

21. http://cran.r-project.org

22. Gentleman, R. C., Carey, V. J., Bates, D. M., et al. (2004) Bioconductor: open software development for computational biology and bioinformatics., *Genome Biol 5*, R80.

23. Schmidberger, M., Morgan, M., Eddelbuettel, D., et al. (2009) State-of-the-art in Parallel Computing with R, *Journal of Statistical Software 31*.

24. Wasserman, W. W., and Sandelin, A. (2004) Applied bioinformatics for the identification of regulatory elements, *Nat Rev Genet 5*, 276-287.

25. Fawcett, T. (2006) An introduction to ROC analysis, *Pattern Recogn Lett 27*, 861-874.

26. Cho, B.-K., Knight, E. M., and Palsson, B. O. (2006) Transcriptional regulation of the fad regulon genes of Escherichia coli by arcA., *Microbiology 152*, 2207-2219.

27. http://www.bioconductor.org/install

28. http://www.stats.uwo.ca/faculty/yu/Rmpi

29. http://www.graphviz.org

Tables

Table 1. Gene Ontology biological process terms enriched (*P-value ≤ 0.05*) among the 147 genes forming the 50%-precision qp-graph network inferred from the oxygen deprivation data in ***(18)***.

GO Term Identifier	P-value	Odds Ratio	Exp. Count	Count	Size	Term
GO:0006350	< 0.0001	4.76	13.79	39	292	transcription
GO:0009059	0.0004	2.14	27.81	43	589	macromolecule biosynthetic process
GO:0019395	0.0022	5.34	1.42	6	30	fatty acid oxidation
GO:0030258	0.0022	5.34	1.42	6	30	lipid modification
GO:0044260	0.0035	1.84	38.15	51	808	cellular macromolecule metabolic process
GO:0044238	0.0073	2.08	66.10	76	1400	primary metabolic process
GO:0006542	0.0096	8.92	0.47	3	10	glutamine biosynthetic process
GO:0006578	0.0124	20.62	0.19	2	4	betaine biosynthetic process
GO:0009268	0.0124	20.62	0.19	2	4	response to pH
GO:0006807	0.0398	1.50	43.44	52	920	nitrogen compound metabolic process
GO:0042594	0.0428	4.44	0.80	3	17	response to starvation

Figure legends

Figure 1. Performance comparison on the oxygen deprivation E. coli data with respect to RegulonDB. **(A)** Graph density as function of the non-rejection rate estimated with $q=3$. **(B)** Precision-recall curves comparing a random ranking of the putative interactions, a ranking made by absolute Pearson correlation (Pairwise PCC) and a ranking derived from the average non-rejection rate (qp-graph).

Figure 2. Functional coherence estimated from networks derived with different strategies and methods. **(A)** A nominal RegulonDB-precision of 50%, **(B)** a nominal RegulonDB-recall of 3%, and **(C)** using the top ranked 1 000 interactions. On the x-axis and between square brackets, under each method, are indicated, respectively, the total number of regulatory modules of the network, the number of them with at least 5 genes and the number of them with at least 5 genes with GO-BP annotations. Among this latter number of modules, the number of them where the transcription factor had GO annotations beyond transcription regulation is noted above between parentheses by *n* and corresponds to the number of modules on which functional coherence could be calculated.

Figure 3. The 50%-precision qp-graph transcriptional network. **(A)** The two largest connected components. **(B)** The *mhpR* regulatory module in detail.

Figure 1

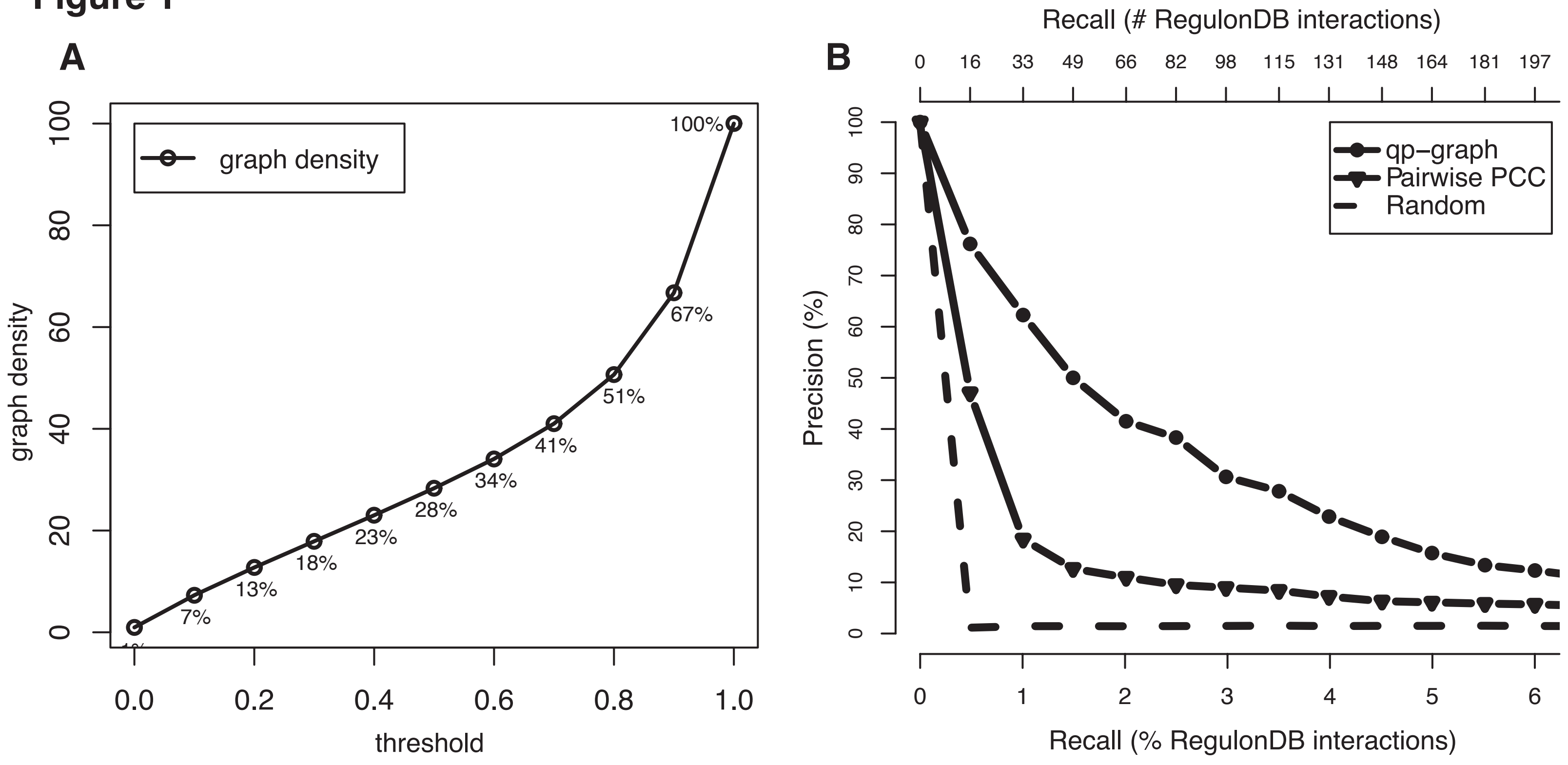


Figure 2

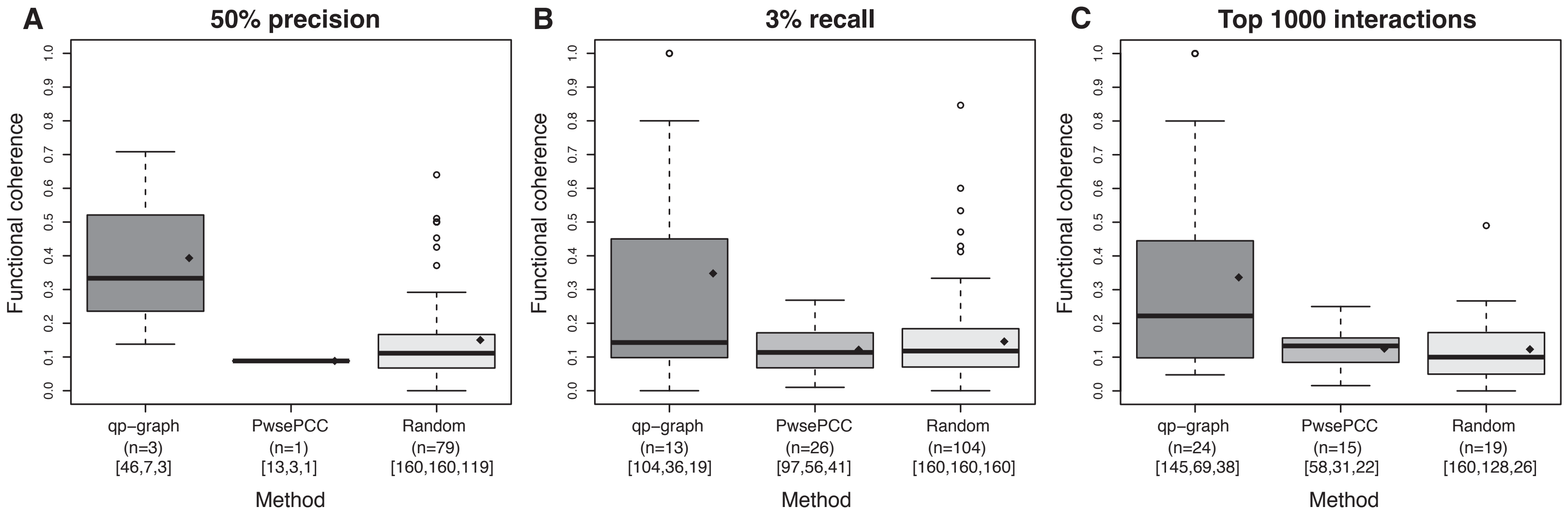
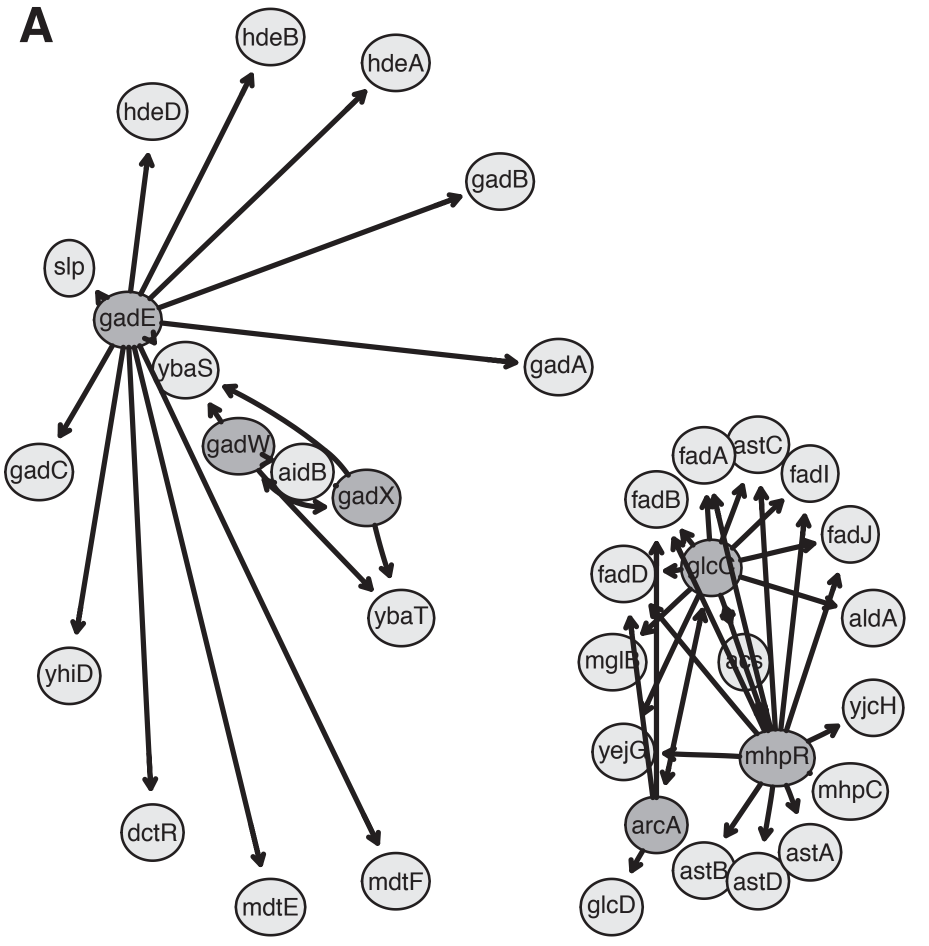
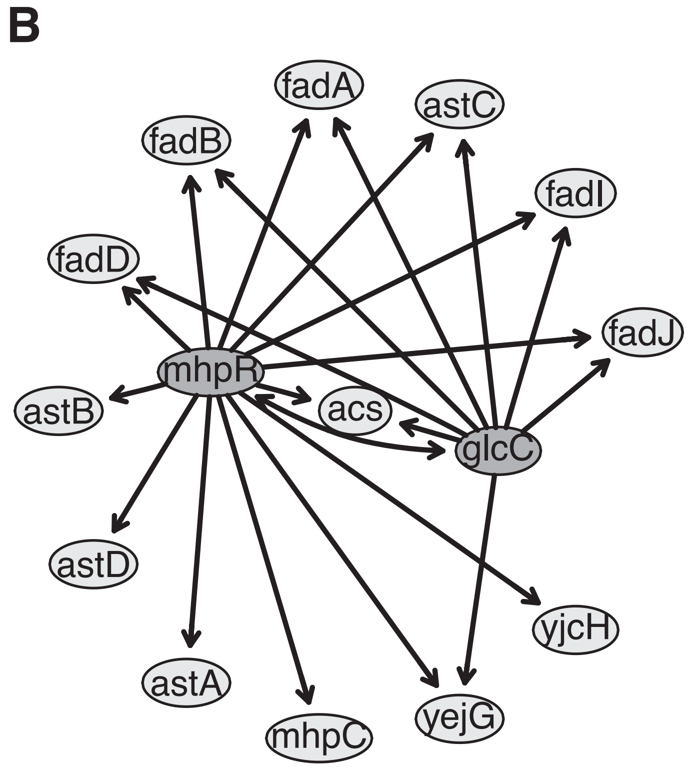


Figure 3


A
hdeB
hdeA
hdeD
gadB
slp
gadE
gadA
ybaS
gadW
aidB
gadX
gadC
ybaT
yhiD
dctR
mdtE
mdtF
fadA
astC
fadB
fadI
fadJ
fadD
glcC
aldA
mglB
acs
yjcH
yejG
mhpR
mhpC
arcA
astA
astB
astD
glcD


B
fadA
astC
fadB
fadI
fadD
fadJ
mhpR
astB
acs
glcC
astD
yjcH
astA
mhpC
yejG