Network Analysis Tools (NeAT)
Tutorial

Sylvain Brohée
sbrohee@ulb.ac.be

Karoline Faust
kfaust@ulb.ac.be

Jacques van Helden
jvhelden@ulb.ac.be

Laboratoire de Bioinformatique des Génomes et des Réseaux (BiGRe)
Laboratory of Genome and Network Biology
Université Libre de Bruxelles, Belgium
http://www.bigre.ulb.ac.be

January 9, 2008

Contents

0.1 Warning

This tutorial is in construction. The current version only covers a very small fraction of the *NeAT* tools. For the tools not covered yet by the tutorial, the DEMO buttons already give some hints about typical cases of utilization. We intend to develop further those tutorials very soon.

Chapter 1

Introduction

Since a few years, large scale biological studies produced huge amounts of data about networks of molecular interactions (protein interactions, gene regulation, metabolic reactions, signal transduction). The integration of these data sets can be combined to acquire a global view of the pieces that, altogether, contribute to the complexity of biological processes. High-throughput data is however notoriously noisy and incomplete, and it is important to evaluate the quality of the different pieces of information that are taken in consideration for building higher views of biological networks.

An important effort will be required to extract reliable information from the ever-increasing ocean of high-throughput data. This will require the utilization of powerful tools that enable us to apply statistical analysis on large graphs. For this purpose, we developed the **Network Analysis Tools** (*NeAT*), as set of tools performing basic operations on networks and clusters.

The tools can be used in three ways:

1. **Web server interface**

   http://rsat.scmbb.ulb.ac.be/neat/

   The Web interface gives a convenient and intuitive access to the tools, and allows you to bring your data sets through some typical analysis work flows in order to extract the best of it.

2. **Stand-alone application**

   http://rsat.scmbb.ulb.ac.be/rsat/distrib/

   Most of the tools are freely available to academic users, according to a licence for non-commercial and non-military usage.

   The license covers both the Regulatory Sequence Analysis Tools (*RSAT*) and the Network Analysis Tools (*NeAT*). It can be downloaded from the RSAT Web site.

3. **Web services**

In addition, people having computer skills can also use be same tools via a Web services interface, in order to integrate them in automatic work-flows. To obtain information on the Web services, connect the *NeAT* web server, and in the left menu, select **Information - Web services**.

Chapter 2

Comparisons between networks

2.1 Introduction

Protein interaction networks have deserved a special attention for molecular biologists, and several high-throughput methods have been developed during the last years, to reveal either pairwise interactions between proteins (two-hybrid technology) or protein complexes (methods relying on mass-spectrometry). The term *interactome* has been defined to denote the complete set of interactions between proteins of a given organism.

Interactome data is typically represented by an un-directed graph, where each node represents a polypeptide, and each edge an interaction between two polypeptides.

The yeast interactome was characterized by the two-hybrid method by two independent groups, Uetz and co-workers [6], and Ito and co-workers [3], respectively. Surprisingly, the two graphs resulting from these experiments showed a very small intersection.

In this tutorial, we will use the program ***compare-graphs*** to analyze the interactome graphs published by from Uetz and Ito, respectively.

We will first perform a detailed comparison, by merging the two graphs, and labelling each node according to the fact that it was found in Ito's network, in Uetz' network, or in both. We will then compute some statistics to estimate the significance of the intersection between the two interactome graphs.

2.2 Computing the intersection, union and differences between two graphs

2.2.1 Study case

In this demonstration, we will compare the networks resulting from the two first publications reporting a complete characterization of the yeast interactome, obtained using the two-hybrid method.The first network [6] contains 865 interactions between 926 proteins.The second network [3] contains 786 interactions between 779 proteins. We

will merge the two networks (i.e. compute their union), and label each edge according to the fact that it is found in Ito's network, Uetz' network, or both. We will also compute the statistical significance of the intersection between the two networks.

2.2.2 Protocol for the web server

1. In the *NeAT* menu, select the command **network comparison**.

   In the right panel, you should now see a form entitled "compare-graphs".

2. Click on the button DEMO.

   The form is now filled with two graphs, and the parameters have been set up to their appropriate value for the demonstration. At the top of the form, you can read some information about the goal of the demo, and the source of the data.

3. Click on the button GO.

   The computation should take a few seconds only. The result page shows you some statistics about the comparison (see interpretation below), and a link pointing to the full result file.

4. Click on the link to see the full result file.

2.2.3 Protocol for the command-line tools

If you have installed a stand-alone version of the NeAT distribution, you can use the program **compare-graphs** on the command-line. This requires to be familiar with the Unix shell interface. If you don't have the stand-alone tools, you can skip this section and read the next section (Interpretation of the results).

   We will now describe the use of **compare-graphs** as a command line tool. The two two-hybrid datasets described in the previous section may be downloaded at the following address *http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/*. These are the files *uetz_2001.tab* and *ito_2002.tab*.

1. Go in the directory where the files containing the graphs to compare are located.

2. Type the following command

   ```
   compare-graphs -v 1 -Q ito_2002.tab -R uetz_2001.tab -return union \
   -o uetz_2001_union_ito_2002.tab
   ```

Using these options, some comparaison statistics are displayed and the results are stored in the tab-delimited file *uetz_2001_union_ito_2002.tab*.

   In order to compute the difference or the intersection, you must change the -return option. For example, to compute the intersection, you shoud type.

```
compare-graphs -v 1 -Q ito_2002.tab -R uetz_2001.tab -return intersection \
-o uetz_2001_inter_ito_2002.tab
```

2.2.4 Interpretation of the results

The program **compare-graphs** uses symbols R and Q respectively, to denote the two graphs to be compared. Usually, R stands for reference, and Q for query.

In our case, R indicates Ito's network, whereas Q indicates Uetz' network. The two input graphs are considered equivalent, there is no reason to consider one of them as reference, but this does not really matter, because the statistics used for the comparison are symmetrical,as we will see below.

Union, intersection and differences

The result file contains the union graph, in tab-delimited format. This format is very convenient for inspecting the result, and for importing it into statistical packages (R, Excel, . . .).

The rows starting with a semicolon (;) are comment lines. They provide you with some information (e.g. statistics about the intersection), but they will be ignored by graph-reading programs. The description of the result graph comes immediately after these comment lines.

Each row corresponds to one arc, and each column specifies one attribute of the arc.

1. **source**: the ID of the source node

2. **target**: the ID of the target node

3. **label**: the label of the arc. As labels, we selected the option "Weights on the query and reference". Since the input graphs were un-weighted, edge labels will be used instead of weights. The label `<NULL>` indicates that an edge is absent from one input network.

4. **color** and **status**: the status of the arc indicates whether it is found at the intersection, or in one graph only. A color code reflects this status, as indicated below.

   - *R.and.Q*: arcs found at the intersection between graphs R and Q. Default color: green.
   - *R.not.Q*: arcs found in graph R but not in graph Q. Default color: violet.
   - *Q.not.R*: arcs found in graph Q but not in graph R. Default color: red.

   The result file contains several thousands of arcs, and we will of course not inspect them by reading each row of this file. Instead, we can generate a drawing in order to obtain an intuitive perception of the graph.

Sizes of the union, intersection and differences

The beginning of the result file gives us some information about the size of the two input files, their union, intersection, and differences.

```
; Counts of nodes and arcs
;    Graph    Nodes    Arcs      Description
;    R        779      786       Reference graph
;    Q        926      865       Query graph
;    QvR      1359     1529      Union
;    Q^R      346      122       Intersection
;    Q!R      580      743       Query not reference
;    R!Q      433      664       Reference not query
```

Statistical significance of the intersection between two graphs

The next lines of the result file give some statistics about the intersection between the two graphs. These statistics are computed in terms of arcs.

```
; Significance of the number of arcs at the intersection
;  Symbol   Value      Description                       Formula
;  N        1359       Nodes in the union
;  M        922761     Max number of arcs in the union   M = N*(N-1)/2
;  E(Q^R)   0.74       Expected arcs in the intersection E(Q^R) = Q*R/M
;  Q^R      122        Observed arcs in the intersection
;  perc_Q   14.10      Percentage of query arcs          perc_Q = 100*Q^R/Q
;  perc_R   15.52      Percentage of reference arcs      perc_R = 100*Q^R/R
;  Jac_sim  0.0798     Jaccard coefficient of similarity Jac_sim = Q^R/(QvR)
;  Pval     2.5e-228   P-value of the intersection       Pval=P(X >= Q^R)
```

A first interesting point is the maximal number of arcs (M) that can be traced between any two nodes of the union graph. In our study case, the graph obtained by merging Ito's and Uetz' data contains $N = 1359$ nodes. This graph is undirected, and there are no self-loops. The maximal number of arcs is thus $M = N * (N - 1)/2 = 922,761$. This number seems huge, compared to the number of arcs observed in either Uetz' ($A_Q = 865$) or Ito's ($A_R = 786$) graphs. This means that these two graphs are sparse: only a very small fraction of the node pairs are linked by an arc.

The next question is to evaluate the statistical significance of the intersection between the two graphs. For this, we can already compute the size that would be expected if we select two random sets of arcs of the same sizes as above ($A_Q = 865$, $A_R = 4,038$).

If the same numbers of arcs were picked up at random in the union graph, we could estimate the probability for an arc to be found in the network R as follows: $P(R) = A_R/M = 0.000852$. Similarly, the probability for an arc of the union graph to be part of the network Q is $P(Q) = A_Q/M = 0.000937$. The probability for an arc to be found independently in two random networks of the same sizes as R and Q is the product of these probabilities.

$$P(QR) = P(Q) * P(R) = A_R/M \cdot A_Q/M = 7.98e - 07$$

The number of arcs expected by chance in the intersection is the probability multiplied by the maximal number of arcs.

$$
\begin{aligned}
E(QR) &= P(QR) \cdot M \\
&= (A_Q \cdot A_R)/M \\
&= 7.98e-07 \cdot 922761 = 0.74
\end{aligned}
$$

Thus, at the intersection between two random sets of interaction, we would expect on the average a bit less than one interaction. It seems thus clear that the 122 interactions found at the intersection between he two published experiments is much higher than the random expectation.

We can even go one step further, and compute the *P-value* of this intersection, i.e. the probability to select at least that many interactions by chance.

The probability to observe *exactly* x arcs at the intersection is given by the hypergeometrical distribution.

$$
P(QR = x) = \frac{C_R^x C_{M-R}^{Q-x}}{C_M^Q} \tag{2.1}
$$

where

   R is the number of arcs in the reference graph;

   Q i the number of arcs in the query graph;

   M is the maximal number of arcs;

   x is the number of arcs at the intersection between the two graphs.

By summing this formula, we obtain the P-value of the intersection, i.e. the probability to observe *at least* x arcs at the intersection.

$$
Pval = P(QR >= x) = \sum_{i=x}^{min(Q,R)} P(X = i) = \sum_{i=x}^{min(Q,R)} \frac{C_R^i C_{M-R}^{Q-i}}{C_M^Q}
$$

We can replace the symbols by the numbers of our study case.

$$
\begin{aligned}
Pval \quad &= \quad P(QR >= 122) \\
&= \quad \sum_{i=x}^{min(865,786)} \frac{C_{786}^i C_{922761-786}^{865-i}}{C_{922761}^{865}} \\
&= 2.5e-228
\end{aligned}
$$

This probabilty is so small that it comes close to the limit of precision of our program ($\approx 10^{-321}$).

Summary

In summary, the comparison revealed that the number of arcs found in common between the two datasets (Ito and Uetz) is highly significant, despite the apparently small percentage of the respective graphs it represents (14.10% of Ito, and 15.52% of Uetz).

2.3   Strengths and weaknesses of the approach

2.4   Exercises

1. Using the tool the tool **network randomization**, generate two random graphs of 1000 nodes and 1000 arcs each (you will need to store these random networks on your hard drive). Use the tool **network comparison** to compare the two random graphs. Discuss the result, including the following questions:

    (a) What is the size of the intersection ? Does it correspond to the expected value ?

    (b) Which P-value do you obtain ? How do you interpret this P-value ?

2. Randomize Ito's network with the tool **network randomization**, and compare this randomized graph with Uetz' network. Discuss the result in the same way as for the previous exercise.

2.5   Troubleshooting

1. The P-value of the intersection between two graphs is 0. Does it mean that it is impossible to have such an intersection by chance alone ?

   No. Any intersection that you observe in practice might occur by chance, but the limit of precision for the hypergeometric P-value is $\approx 10^{-321}$. Thus, a value of 0 can be interpreted as $Pval < 10^{-321}$.

2. The web server indicates that the result will appear, and after a few minutes my browser displays a message "No response the server".

   How big are the two graphs that you are comparing ? In principle, compare-graphs can treat large graphs in a short time, but if your graphs are very large (e.g. several hundreds of thousands of arcs), the processing time may exceed the patience of your browser. In such case, you should consider either to install the stand-alone version of *NeAT* on your computer, or write a script that uses *NeAT* via their Web services interface.

Chapter 3

Node degree statistics

3.1   Introduction

In a graph, the degree k of a node is the number of edges connected to this node. If the graph is directed, we can make a distinction between the in-degree (the number input arcs) and the out-degree (number of output arcs). In this case, the degree of the node consists in the sum of the in-degree and of the out-degree of this node.

Different nodes having different degrees, this variability is characterized by the degree distribution function $P(k)$, which gives the probability that a node has exactly k edges, or, in other words gives the observed frequency of a node of degree k.

Scale-free graphs were first described by Barabasi based on the study of the web connectivity, followed by several different biological networks [4].

A graph is scale-free if the distribution of the vertex degree (k) follows a power-law distribution of the form $P(k)\,k^{-\gamma}$.

The main property of such graphs is that it should have on one hand some highly connected nodes, called hubs, which are central to the network topology, and *keep the network together* and on the other hand a lot of poorly connected nodes linked to the hubs.

In the following, we will check if this scale free property also applies to the two-hybrid network described by Uetz *et al* [6] by computing the degree of each node and plotting the node degree distribution of the graph.

3.2   Analysis of the node degree distribution of a biological network

3.2.1   Study case

In this demonstration, we will analyze the node degree distribution of the first published yeast protein interaction network. This network yeast interactome, obtained using the two-hybrid method [6]. This network contains 865 interactions between 926 proteins.

3.2.2   Protocol for the web server

1. In the *NeAT* menu, select the command ***stats on node degrees***.

   In the right panel, you should now see a form entitled "graph-node-degree".

2. Click on the button DEMO.

   The form is now filled with a graph in the tab-delimited format, and the parameters have been set up to their appropriate value for the demonstration, i.e., the degree of all nodes will be computed. At the top of the form, you can read some information about the goal of the demo, and the source of the data.

3. Click on the button GO.

   The computation should take less than one minute. On one haned, the result page displays a link to the result file and on the other had the graphics and raw data of the node degree distribution are also available. These will be discussed in the *Interpretation of the results* section.

3.2.3   Protocol for the command-line tools

If you have installed a stand-alone version of the NeAT distribution, you can use the program ***graph-node-degree*** on the command-line. This requires to be familiar with the Unix shell interface. If you don't have the stand-alone tools, you can skip this section and read the next section (Interpretation of the results).

We will now describe the use of ***graph-node-degree*** as a command line tool. The two two-hybrid dataset described in the previous section may be downloaded at the following address *http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/*. This is the file *uetz_2001.tab*.

1. The first step consist in applying ***graph-node-degree*** on the two-hybrid dataset. To this, go into the directory where you downloaded the file *uetz_2001.tab* and use this command.

   ```
   graph-node-degree -v 1 -i uetz_2001.tab -all -o uetz_2001_degrees.tab
   ```

   The file *uetz_2001_degrees.tab* is created and contains the in-, out- and global degree of each node of the Uetz *et al* data set.

2. In the second step, we will study the degree distribution of the nodes. To this, we use the program ***classfreq*** from the RSAT suite that compute the distribution of a set of number. As the graph we are working with is undirected, we will only compute this degree distribution for the global degree of the nodes which is the fourth column of the file *uetz_2001_degrees.tab* obtained at the previous step.

   ```
   classfreq -i uetz_2001_degrees.tab -v 1 -col 4 -ci 1 -o uetz_2001_degrees_freq.tab
   ```

3. Finally, we will display the distribution graph in the PNG format in order to visualize the degree distribution and determine if it has a scale free behaviour. The program XYgraph from RSA-tools will be used for this purpose. Note that we could use other tools like **Microsoft Excel** or **R**. The results will be stored in the file *uetz_2001_degrees_freq.png* that you can open with any visualization tool.

```
XYgraph -i uetz_2001_degrees_freq.tab \
-title 'Global node degree distribution for Uetz et al (2001) interaction graph' \
-xcol 2 -ycol 4,6 -xleg1 Degree -lines \
-yleg1 'Number of nodes' -legend -header -format png \
-o uetz_2001_degrees_freq.png
```

3.2.4 Interpretation of the results

graph-node-degree result file

Open the resulting file produced by **graph-node-degree**. According to the requested level of verbosity (`-v #` option), the file begins with some lines starting with the '#' or ';' symbols that contains some information about the graph and the description of the columns.

The results consists in a five column data set.

1. Node name

2. Number of ingoing edges

3. Number of outgoing edges

4. Global degree (sum of the second and third columns).

5. Indicate whether the node only contains outgoing edges (source node) or ingoing edges (target node) or both (intermediate).

Node degree distribution

Let us first have a look at the node degree distribution data file produced by the **class-freq** program (raw data). This file is a tab-delimited file containing 9 columns. Each line consists in a value interval. In our case, the value is the degree of the nodes.

1. Minimal value of the interval

2. Maximal value of the interval

3. Central value of the interval

4. Frequency : Number of elements in this class interval (number of nodes having a degree comprised betwee the minimal and the maximal values.

5. Cumulative frequency.

6. Inverse cumulative frequency

7. Relative frequency : number of elements in this class over the total number of elements

8. Relative cumulative frequency

9. Inverse relative cumulative frequency

The first result line contains the distribution results for the nodes having only one neighbour (i.e. degree comprised between 1 and 2), from it we can see that 577 over 926, i.e., 62% of the nodes have a degree of one. Moreover, about 90% of the nodes have a degree lower than 4. This is indicative of the scale-free nature of the interaction network.

The figure best illustrates the scale-freeness of the graph. When looking at the graphical representation of this distribution, we can see two curves. The blue curve represents the absolute frequency and the green curve the inverse cumulative frequency. The exponential decrease of both curves shows that there are a lot more nodes poorly connected than highly connected (hubs). The Uetz graph thus presents a scale free behaviour.

Chapter 4

Study of the neighborhood of the nodes

4.1 Introduction

In a graph, the neighbours of a node consist in the set of nodes that are connected to this node up to a certain distance, i.e., the number of steps between the source node and its neighbours. In weighted graphs, one can also consider the neighbours up to a certain maximal weight.

In the following, we will call the node for which we search the neighbours the seed node.

According to the type of graph, it might be interesting to retrieve the neighbours of the nodes in a graph.

For example, in protein-protein interaction network, the function of the neighbours of an protein whose biological role is unknown might give insight in the function of the protein. Moreover, in interaction graphs, if a group of neighbours have similar biological functions, they are likely to form a structural complex.

In co-regulation networks, where each node is a gene and an edge between two genes means that those genes are co-regulated (i.e. co-repressed and co-expressed), exploring the neighbours of the nodes may help in the discovery of new regulons.

In the following, we will illustrate the study of nodes neighborhood by looking for neighbours of some orphan proteins (i.e. protein of unknown function) in a protein protein interaction network. We will then look if the neighbours of these proteins present similar functions.

4.2   Analysis of the node degree distribution of a biological network

4.2.1   Study case

In this demonstration, we will analyze the neighbours of the orphan nodes of the Gavin *et al* (2006) interaction data set. These interaction data were obtained by co-immunoprecipitation followed by a mass spectrometry experiment in order to discover structural protein complexes. [2]. This network contains 6531 interactions between 1430 proteins.

   We will then compare these groups of neighbours with functionnal classes of proteins annotated in the MIPS [5] in order to detect if the groups of neighbours present a significatively high number of co-regulated proteins.

4.2.2   Protocol for the web server

1. In the **NeAT** menu, select the command **get node neighborhood**.

   In the right panel, you should now see a form entitled "graph-neighbours".

2. Go on the demo dataset download web page.*http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/* and download the files *gavin_2006_names.tab*, *orphan_gavin.tab* and *mips_name_class_description.tab* on your computer.

3. In the *Upload graph from file* text area, load the file *gavin_2006_names.tab* you just downloaded.

4. Uncheck `Include each node in its neighborhood (with a distance of zero)`

5. Check the *List of nodes* in the seed node part of the form

6. In the *Upload seed nodes from file* text area, load the file *orphan_gavin.tab.tab* you just downloaded.

7. Click on the button `GO`.

   The computation should take less than one minute.

   The result page should display the results in the tab-delimited or HTML format. These files will be described in the section *Interpretation of the results*

8. We will now see if the different groups of neighbours contain a significantly high number of proteins of similar function. To this, we will compare the groups of neighbours we just obtained with annotated groups of proteins, e.g., the genes annotated according to the gene ontology [1] or, in this example, to the functionnal classes of the MIPS [5]. Click on the button `Compare the groups of neighbours`.

   You are redirected to the form of another program **compare-classes** that allows to compare two class files (the query file and the reference file). Each class of the

query file is compared to each class of the reference file. The number of common elements is reported, as well as the probability to observe at least this number of common elements by chance alone. The query classes are already loaded and consist in the different groups of neighbours we discovered previously with **graph-neighbours**.

9. In the *Upload reference classes from file* text area, load the file *mips_name_class_description.tab* downloaded in the first part of this tutorial. The classes files are two column files, the first column contains the element and the second column the class to which the element belong. Elements may belong to more than one class.

10. The default paramaters are sufficient. We will only keep the comparison presenting a significance higher than 0.

11. Click on the button GO.

12. You obtain the links to the result file in the tab-delimited format or in the HTML format. The obtained results will be described in the next section.

4.2.3 Protocol for the command-line tools

If you have installed a stand-alone version of the NeAT distribution, you can use the program **graph-neighbours** on the command-line. This requires to be familiar with the Unix shell interface. If you don't have the stand-alone tools, you can skip this section and read the next section (Interpretation of the results).

We will now describe the use of **graph-neighbours** as a command line tool. The Gavin *et al* (2006) [2] co-immunoprecipitation dataset described in the previous section and the other files necessary for this tutorial may be downloaded at the following address *http://rsat.scmbb.ulb.ac.be/rsat/data/neat_tuto_data/* (*gavin_2006_names.tab*, *orphan_gavin.tab* and *mips_name_class_description.tab*).

1. The first step consist in applying **graph-neighbours** on the two-hybrid dataset. To this, go into the directory where you downloaded the files *gavin_2006_names.tab*, *orphan_gavin.tab* and use this command.

```
graph-neighbours -v 1 -i gavin_2006_names.tab \
 -seedf orphan_gavin.tab \
 -o gavin_2006_orphan_neighbours_1.tab
```

The file *gavin_2006_orphan_neighbours_1.tab* is created and contains for each node of the seed file the list of its direct neighbours, i.e., for each protein, the list of protein for which an interaction was detected with it.

2. In the second step, we will compare these groups of neighbours to different groups of annotated proteins in order to discover if the groups of neighbours do contain a significatively high number of proteins of similar functions. This will give insights to the function of the orphans proteins used as seed node when looking for their neighbours. To this, we will use the RSAT **compare-classes**

program that allows to compare two class files (the query file and the reference file) (see previous section or the RSAT tutorial for a more complete description of **compare-classes**). Use the following command to compare the two files.

```
compare-classes -v 1 \
-q gavin_2006_orphan_neighbours_1.tab -r mips_name_class_description.tab \
-sort sig -return proba,occ,jac_sim \
-o gavin_2006_orphan_neighbours_1_cc_mips_functionnal_classes.tab -lth sig 0
```

We obtain a file *gavin 2006 orphan neighbours 1 cc mips functionnal classes.tab* containing the significant comparaison results. We will discuss it in the following section (interpretation of the results).

4.2.4   Interpretation of the results

graph-neighbours result file

According to the requested level of verbosity, the result file may first contain several lines (starting with "#" or ";". These deliver some information about the analysed graph (number of nodes, edges, seed nodes, ...). The results are then displayed in four columns.

1. Name of the neighbour.

2. Name of the seed node (for which the neighbours are seeked in the graph).

3. Distance between the seed node and its neighbour (number of steps).

4. The last column, only relevant for directed graph, indicate whether the arc between the seed node and its neighbour is an out- or an in-going arc.

This file can be considered as a class file (see above for a more complete description) with the number of the neighbour being the member (first column) and the number of the seed node, the name of the class (second column).

compare-classes result file

The result of the comparaison between the groups of neighbours and the MIPS annotated classes are displayed in a multi-column file sorted by decreasing order of significance. When looking at the HTML version of the file, you may click on the header on the column to sort the table according to this field.

Each line displays the comparaison between a MIPS annotated class (reference class) and a group of neighbours (query class). What we want to know is if the number there is a significant high number of members of the same MIPS class in a group of neighours.

- ref : Name of the MIPS functionnal class.

- query : Name of the group of neighbours (seed node).

- R : Size of the reference class (number of members in this MIPS class).

- Q : Size of the query class (number of neighbour for this seed node).

- QR : Intersection size between the group of neighbours and the functionnal class.

- QvR : Union size between the group of neighbours and the functionnal class.

- R!Q : Elements that are in the functionnal class but not in the groups of neighbours.

- Q!R : Elements that are not in the functionnal class but are in the groups of neighbours.

- !Q!R : Elements that are not in the functionnal class nor in the groups of neighbours.

- P-val : P-value of the comparaison, propability (according to the hypergeometric law) to be wrong when claimin that there is a significatively high number of proteins of the same class in the group of neighbours.

- E-val : E-value of the comparaison. P-value multiplied by the total number of comparaisons. This value corresponds to the estimated number of false positives for that P-value threshold.

- sig : Significance of the comparaison. This correpsonds to $-log_{10}(E - val)$. This index gives an intuitive perception of the exceptionality of the common elements : a negative significance indicates that the common matches are likely to come by chance alone, a ositive value that they are significant. Higher sig values indicate a higher significance.

Considering the file, we can observe that 7 seed nodes (on the 46) have a group of neighbours presenting a similar function. For example, 9 out of the 10 neighbours of the Yil161w protein (interacting with this protein) have their function related to ribosome biogenesis and 8 out of 10 neigbours are located in the cytoplasm. This may indicate that this protein may also be implied in ribosome biogenesis

Chapter 5

Recapitulative exercises

Bibliography

[1] The Gene Ontology Consortium. The gene ontology project in 2008. *Nucleic Acids Res*, Nov 2007.

[2] Anne-Claude Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars Juhl Jensen, Sonja Bastuck, Birgit Dmpelfeld, Angela Edelmann, Marie-Anne Heurtier, Verena Hoffman, Christian Hoefert, Karin Klein, Manuela Hudak, Anne-Marie Michon, Malgorzata Schelder, Markus Schirle, Marita Remor, Tatjana Rudi, Sean Hooper, Andreas Bauer, Tewis Bouwmeester, Georg Casari, Gerard Drewes, Gitte Neubauer, Jens M Rick, Bernhard Kuster, Peer Bork, Robert B Russell, and Giulio Superti-Furga. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631–636, Mar 2006.

[3] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proc Natl Acad Sci U S A*, 98(8):4569–74., 2001.

[4] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabsi. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, Oct 2000.

[5] H. W. Mewes, S. Dietmann, D. Frishman, R. Gregory, G. Mannhaupt, K. F X Mayer, M. Mnsterktter, A. Ruepp, M. Spannagl, V. Stmpflen, and T. Rattei. Mips: analysis and annotation of genome information in 2007. *Nucleic Acids Res*, Dec 2007.

[6] P. Uetz, L. Giot, G. Cagney, T. A. Mansfield, R. S. Judson, J. R. Knight, D. Lockshon, V. Narayan, M. Srinivasan, P. Pochart, A. Qureshi-Emili, Y. Li, B. Godwin, D. Conover, T. Kalbfleisch, G. Vijayadamodar, M. Yang, M. Johnston, S. Fields, and J. M. Rothberg. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. *Nature*, 403(6770):623–7., 2000.