**Experimental gene sets**

Imagine an experiment has been performed and e.g. differentially expressed genes have

been detected. Obviously, these would be a subset of your yeast chromosome.

1. Download the ​ S.cerevisiae ​ S228C proteome from Uniprot

(http://www.uniprot.org/proteomes/) and use BLAST to match your predicted

genes against UniProt proteome.

1. We first downloaded the chromosome X from S.cerevisiae S228C proteomes from Uniprot.
2. In the same directory, make a protein blast database using the downloaded proteomes by typing the below command:

makeblastdb -in YeastChrX.fa -dbtype prot

1. Blast the query multi-fasta file generated from GENESCAN in practical 2 against the database in 2)

blastp -outfmt 5 -query protein30.fa -db YeastChrX.fa -out out.yeast.blastp.txt

2. Parse blast results to extract gene names (GN=) for genes present on chromosome.

The python blastResultParser.py was modified such that it displays the best-hit in YeastChrX.fa.

An output file “out.yeast.blastp.txt” was obtained.

python2 blastResultParser.py out.yeast.blastp.txt

The modified script is in attachment no.1.

3. Find two experimental gene sets (experiments.txt) that overlap most with genes

on your chromosome. The goal is to find out if any of these are present on the

chromosome.

The python script readID.py that extracts the overlap between experiments.txt and out.yeast.bastp.txt is in attachment no.2. It reads the two files and outputs a dictionary containing line as key and number of overlapping proteins as value, such as :

Line40: 0

Moreover, it sorts the dictionary so that keys are ordered according to ascending number of overlapping proteins.

4. Report how many genes overlap and save two experimental gene sets for further

analysis.

The output from readID.py is as below:

[('line40', 0), ('line34', 0), ('line17', 0), ('line39', 0), ('line51', 0), ('line5', 0), ('line4', 0), ('line19', 0), ('line55', 0), ('line16', 0), ('line30', 0), ('line1', 1), ('line22', 1), ('line53', 1), ('line43', 1), ('line8', 1), ('line42', 1), ('line31', 1), ('line49', 1), ('line20', 1), ('line28', 1), ('line7', 1), ('line6', 1), ('line50', 1), ('line33', 1), ('line52', 1), ('line25', 1), ('line2', 1), ('line27', 1), ('line38', 1), ('line12', 1), ('line23', 1), ('line37', 2), ('line24', 2), ('line48', 2), ('line47', 2), ('line45', 2), ('line44', 2), ('line11', 2), ('line21', 2), ('line3', 2), ('line10', 2), ('line14', 2), ('line41', 2), ('line36', 2), ('line46', 2), ('line18', 2), ('line35', 2), ('line15', 2), ('line29', 2), ('line13', 2), ('line9', 2), ('line32', 2), ('line26', 3), ('line54', 4)]

Therefore, line 54 and line 26 in experiments.txt should be selected for further use, as they have the most overlaps (4 proteins and 3 proteins respectively) with the out.yeast.blastp.txt.

**Comparative network analysis using FunCoup and STRING**

1. Using your two experimental gene sets query FunCoup and STRING for

sub-networks containing these genes.

a. FunCoup works with space delimited list of gene names, STRING expects

each gene name in new line for a query.

b. Use the same expansion depth or max number of interactors for searching

both databases, so the results are comparable.

In FunCoup, we set the expansion depth as 0, while in STRING, we set the max number of interactions as “none/query proteins only”, so that this parameter is comparable in the two methods.

2. Compare results

a. How do these networks differ in terms of nodes, links, and hubs (the three

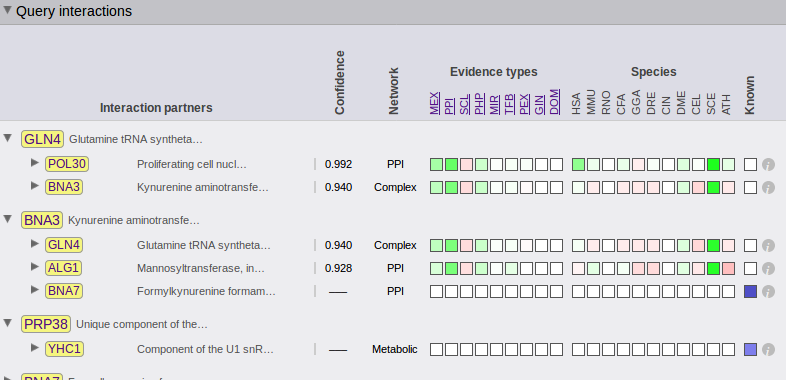
nodes with highest degree)?

* For set no. 26, the agreement between Funcoup and STRING is not as obvious as one would expect. Their only shared feature is the interaction between FAS1 and FAS2.
* For set no. 54, they have the below shared features:
* Nodes: MET3, MET16, PRP28, PRP31, PRP40, KIN28, TOA2, PHS1, TSC13
* Links: MET3-MET16, PRP40-PRP31-PRP28, KIN28-TOA2
* Hubs: by Funcoup (PRP28, PRP40, GAR1), by STRING (GSH1, FAS1, PRP28)

|  |  |
| --- | --- |
| **Set no. 26**  **Set no. 54**  **Funcoup** | **STRING** |
|  |  |

b. What is the most common evidence type with high confidence (>0.9)?

|  |  |  |
| --- | --- | --- |
|  | Funcoup | STRING |
| set26 | Protein-protein interaction | textmining |
| set54 | mRNA co-expression | textmining |

Take set 16 by Funcoup as an example, It is shown that PPI (protein-protein interaction) is the most common evidence type, as it is bright green in color, which symbolizes the highest supporting score.

c. Can you explain the differences in terms of underlying data sources in the

databases?

* Funcoup:
* Its data sources are mainly experimental, e.g. physical protein-protein interactions, mRNA/protein co-expression, co-regulation, which should be the most reliable
* Or can come from literature, e.g. genetic profile, subcellular localization, shared transcription factor
* Bioinformations: domain prediction or phylogenic relationship
* STRING:
* Experimental: co-expression or high-throughput experiment
* Bioinformatics: textmining, genomic context prediction
* From other primary databases[[1]](#endnote-1)

As a consequence, Funcoup should be more accurate, as it is heavily based on multiple types of experimental results, such as mRNA co-expression, protein co-expression, miRNA co-regulation....which are more reliable. It should be more sensitive than STRING as well, because it also takes into account the evolutionary relationship between proteins (i.e. using InParanoid as reference).[[2]](#endnote-2)

This can be reflected by the fact that in question 2b, the most common evidence type found by STRING is textmining, which is by no means comparable to Funcoup.

1. Szklarczyk D, Morris JH, Cook H, Kuhn M, Wyder S, Simonovic M, Santos A, Doncheva NT, Roth A, Bork P, Jensen LJ, von Mering C. 2017. The STRING database in 2017: quality-controlled protein-protein association networks, made broadly accessible. *Nucleic Acids Research* 45: D362-368. [↑](#endnote-ref-1)
2. Schmitt T, Ogris C, Sonnhammer EL. 2017.[FunCoup 3.0: database of genome-wide functional coupling networks.](http://www.ncbi.nlm.nih.gov/pubmed/24185702) *Nucleic Acids Research* 42 (Database issue): D380-388. [↑](#endnote-ref-2)