simulatorAPMS

Tony Chiang
Denise Scholtens
Robert Gentleman

simulatorAPMS is divided into three distinct components. The first component contains functions for simulating the AP-MS technology. The second employs a protein complex membership algorithm using data drawn from the simulated AP-MS experiment. The last component of the package contains the statistical tools used to test the accuracy of the complex membership algorithm(s)' estimates.

```
> library(apComplex)

Loading required package: graph
Loading required package: cluster
Loading required package: Ruuid

> library(simulatorAPMS)
```

AP-MS Data

AP-MS technology is designed to detect complex comembership among proteins. A set of proteins are used as baits, and in each purification, the bait protein finds the set of hit proteins with which it shares membership in at least one complex. Suppose proteins P_1, P_2, P_4, and P_6 compose one complex and proteins P_3, P_4 and P_5 compose a separate complex (See Figure 1). If proteins P_1, P_2, and P_3 are used as baits, then with perfectly sensitive and specific AP-MS technology, the following data should be observed.

		Hits					
		P_1	P_2	P_3	P_4	P_5	P_6
Baits	P_1	1	1	0	1	0	1
	P_2	1	1	0	1	0	1
	P_3	0	0	1	1	1	0

The rows of the matrix are baits, the columns are hits, an entry of 1 in the ith row and jth column indicates that bait protein i finds protein j as a hit, and an entry of 0 in the ith row and jth column indicates that bait protein i does not find protein j as a hit. The diagonal entries are 1 since a protein is always a complex comember with itself. Note that bait proteins can be found as hits by other bait proteins. Also note that some proteins are never used as baits.

A graph of the data is useful for understanding which comembership relationships are tested in AP-MS experiments and which are not. In the graph in Figure 1, nodes represent proteins and directed edges from baits to hits represent complex comembership. Bait proteins are yellow and hit-only proteins (i.e. proteins that are found as hits but never used as baits) are white. Directed edges always originate at yellow bait proteins and point to the set of hits detected by that bait. The red reciprocated edge connecting P_1 and P_2 represents a bait-bait relationship that is tested twice, once in each purification. The gray unreciprocated edges represent bait-hit-only edges that are only tested once. Missing edges between baits and other baits or hit-only proteins represent comemberships that are tested, but not observed. Edges between hit-only proteins are always missing, not because the comemberships are known not to exist, but because they are never tested.

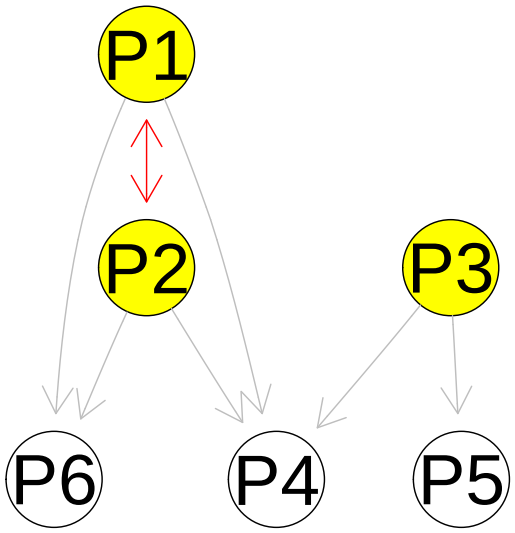

Figure 1: True complex comemberships that would be detected with perfectly sensitive and specific AP-MS technology using P_1, P_2, and P_3 as baits.

In reality, AP-MS technology is neither perfectly sensitive nor specific, resulting in false positive (FP) and false negative (FN) observations of the complex comemberships under investigation. Suppose in this experiment, we make a FN observation between P_2 and P_4, i.e. P_4 is not found as a hit when we use P_2 as a bait. Also suppose we make two FP observations: 1) when we use P_3 as a bait, we find an extraneous hit-only protein P_7, and 2) when performing a purification using P_8 as a bait, we find P_3 as a hit. Data for this example are contained in the matrix `apEX`. In this matrix, rows again represent baits and columns represent hits.

```
> data(apEX)
> apEX
```

```
   P1 P2 P3 P8 P4 P5 P6 P7
P1  1  1  0  0  1  0  1  0
```

```
P2  1  1  0  0  0  0  1  0
P3  0  0  1  0  1  1  0  1
P8  0  0  1  1  0  0  0  0
```

The graph of the data in Figure 2 demonstrates the observations recorded in `apEX`. Note the missing edge from P_2 to P_4 and the new edge from P_3 to P_7. Also note the blue unreciprocated edge between P_3 and P_8 – this is a complex comembership that was tested twice when P_3 and P_8 were used as baits, but only detected once in the purification using P_8 as a bait.

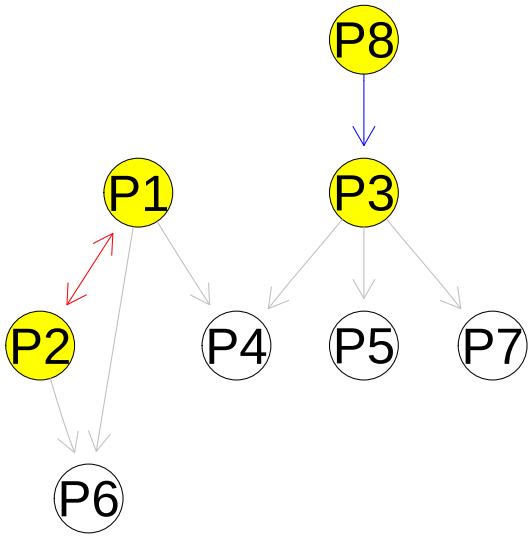

Figure 2: Hypothetical data from an AP-MS experiment with a FN observation between P_2 and P_6 and FP observations between P_3 and P_7 and P_8 and P_3.

1 Preliminaries

Before we proceed onto the package itself, it is necessary to acquaint ourselves with the language we will use. We can consider any experimental technology as a black box; the input to the black box is some true state of nature (TSN) which is basically an in silico interactome (ISI), the collection of protein complexes for some cell under some conditions. The output of the black box is an output from the simulated experiments. It is clear that the experiment begins with some representation of the truth. For complex co-membership, we begin with an ISI described by its bipartite graph: one set of nodes represents proteins; the other, protein complexes. The input is the incidence matrix representation of the bipartite graph: the rows index the proteins; the columns, protein complexes. The matrix has a one in row i and column j if protein i is part of complex j; otherwise that entry is zero. Here is an example of such a matrix:

```
> data(simEX)
> simEX
```

	MBME1	MBME2	MBME3	MBME4	MBME5	MBME6	MBME7	MBME8	MBME9	MBME10
YAL015C	1	1	1	0	0	0	0	0	0	0
YAL017W	0	0	0	1	0	0	0	0	0	0
YAL021C	0	0	0	0	0	0	0	0	0	0
YAL036C	0	0	0	0	0	0	0	0	0	0
YAR003W	0	0	0	0	0	0	0	0	0	0
YAR007C	0	0	0	0	1	0	0	0	0	0
YAR019C	0	0	0	0	0	0	0	0	0	0
YBL021C	0	0	0	0	0	0	0	0	0	0
YBL026W	0	0	0	0	0	1	1	1	0	0
YBL036C	0	0	0	0	0	0	0	0	0	0
YBL049W	0	0	0	0	0	0	0	0	1	0
YBL056W	0	0	0	0	0	0	0	0	0	0
YBL088C	0	0	0	0	0	0	0	0	0	0
YBR017C	0	0	0	0	0	0	0	0	0	1
YBR055C	0	0	0	0	0	0	0	0	0	0
YBR059C	0	0	0	0	0	0	0	0	0	0
YBR082C	0	0	0	0	0	0	0	0	0	0
YBR083W	0	0	0	0	0	0	0	0	0	0
YBR088C	0	0	0	0	0	0	0	0	0	0
YBR094W	0	0	0	0	0	0	0	0	0	0
YBR103W	0	0	0	0	0	0	0	0	0	0
YBR109C	0	0	0	0	0	0	0	0	0	0
YBR114W	0	0	0	0	0	0	0	0	0	0
YBR125C	0	0	0	0	0	0	0	0	0	0
YBR130C	0	0	0	0	0	0	0	0	0	0

Each organism or cell has an unique interactome for some specified conditions and time. For our in silico interactome, we have taken the estimation of apComplex on the HMSPCI yeast data set by Ho, et al. We note that any estimate can serve as the ISI. Once we have the bipartite graph representation of the ISI, we may begin the simulation of the APMS technology.

2 Simulator

In this section we discuss how to use the simulator in the *simulatorAPMS* package. In describing how to use the main function, `runSimulators()`, we will work with an example:

>runSimulators(TSNMatEX, vBaitsEX, vDeformed, vSticky, rateFP, rateFN, rateD, rateS, missedProtHMSPCI, Seed)

First we describe each of the inputs. Each input parameter has a wet-lab correspondent, and we describe the how each affects the output of the experiment and, thus, the simulation.

TSNMatEX - The first parameter is the incidence matrix of the bipartite graph representation of our ISI. It is our model true state of nature. In the wet-lab experiment, this maybe some cell line or tissue under some specified conditions. The goal of AP-MS technology is to estimate $TSNMatEX$, so too then does our simulation technology.

vBaitsEX - Just as AP-MS technology uses a subset of proteins from the cell or tissue of interests as bait, so too does our simulation use a subset of proteins as baits. This parameter is given as a character vector composed of the protein names used as baits which is identical to names used for the rows of the $TSNMatEx$.

vDeformed - In AP-MS, baits need to be tagged so that they can be identified at the end of the affinity process. Certain baits have been experimentally verified to lose normal functionality when tagged. Thus deformed baits are known to interact with very few proteins in the experimentation giving rise false negative observations. The vDeformed parameter is also a character vector of protein names; they are a subset of the vBaitsEX vector.

vSticky - In the course of the experiment, some baits are found to falsely interact with a large number other proteins. Proteins that have high occurences of interactions with a large number of other proteins are called *sticky*, and as this name suggest, these proteins deliver a high number of false positive interactions. The vSticky parameter is also a subset of vBaitsEX.

rateFP - Through any experiment, there are a number of stochastic elements by which errors can occur. How sensitive the technology, how specific the technology, etc all contribute to error in the observed data. The rateFP parameter is a scalar in the unit interval dictating the probability of a false positive interaction between bait b and protein p.

rateFN - Analogous to rateFP, rateFN is the probability that the technology (simulation) will record a false negative interaction.

rateD - For each deformed bait, we need to describe how deformed the protein has become, i.e. what proportion of prey will each deformed bait miss. rateD is a named vector of scalars (of the same size as vDeformed) estimating this proportion for each deformed bait.

rateS - Analogous to rateD, rateS is a vector of scalars estimating how the proportion of proteins to which each proteins interacts.

missedProtHMSPCI - The ISI's bipartite graph might not include all the proteins, since proteins not involved in any non-trivial complexes need not be recorded in the bipartite graph. These proteins need to be re-inserted to the in silico model organism's bipartite graph since they would be present in the wet-lab organism. In the simulation, these prpteins will contribute to any false positive interactions within the simulation.

seed - The *seed* is not used in the wet-lab experiments. It is a computational parameter that sets the seed for the random integer generator for the purpose of reproducible data sets. seed is any three digit positive integer.

The output from `runSimulators()` does not represent a bipartite graph. The simulation derives the protein-protein interaction graph by $TSNMatEX \otimes TSNMatEX^T$. This interaction graph does not represent direct binary protein-protein interactions, but rather protein

co-membership interaction. It is from this matrix that the simulated output is derived. This output reflects the data output of an actual AP-MS experiment, and it is given as the incidence matrix representation where each row indexed by the bait proteins and the columns represents all proteins found in the cell.

```
> data(simEX)
> data(vBaitsEX)
> data(missedProtEX)
> vDeformed <- vBaitsEX[2]
> vSticky <- vBaitsEX[5]
> rateFP <- 0.1
> rateFN <- 0.25
> rateD <- 0.5
> rateS <- 0.66
> seeed <- 237
> runSimulators(simEX, vBaitsEX, vDeformed, vSticky, rateFP, rateFN,
+     rateD, rateS, missedProtEX, seeed)
```

	YAL015C	YAL017W	YAL021C	YAL036C	YAR007C	YAR019C	YBL021C	YBL026W	YBL036C
YAL015C	1	0	0	0	0	0	0	0	0
YAL017W	0	1	0	0	0	0	0	0	0
YAL021C	0	1	1	0	0	0	0	0	0
YAL036C	0	0	0	1	0	1	1	0	0
YAR007C	1	0	0	0	1	0	0	0	0

	YBL049W	YBL056W	YBL088C	YBR017C	YBR055C	YBR059C	YBR082C	YBR083W	YBR088C
YAL015C	0	0	0	0	0	0	1	0	0
YAL017W	0	0	1	0	0	0	0	0	1
YAL021C	0	0	0	0	0	1	0	1	0
YAL036C	0	0	0	0	0	0	0	0	1
YAR007C	0	0	0	0	0	0	0	0	0

	YBR094W	YBR103W	YBR109C	YBR114W	YBR125C	YBR130C	YPR181C	YML014W	YNL042W
YAL015C	1	0	1	0	0	0	0	0	0
YAL017W	0	0	0	0	0	0	0	0	0
YAL021C	0	0	0	0	0	1	0	0	0
YAL036C	0	0	0	0	1	0	0	0	0
YAR007C	0	0	0	0	0	0	0	1	0

	YPR180W	YHR050W	YIL069C	YGL015C	YJL087C	YOL145C	YML101C
YAL015C	0	0	0	0	0	0	0
YAL017W	0	0	0	0	0	0	0
YAL021C	0	0	0	0	0	0	0
YAL036C	0	0	0	0	0	0	0
YAR007C	0	0	0	0	0	0	0

3 Complex Estimation Algorithm

From the simulated data, the *simulatorAPMS* package calls the `findComplexes()` function in the *apComplex* package, a complex co-membership algorithm. For detailed instructions concerning the *apComplex* package, the user can refer to its own vignette. One of the functionality of the *simulatorAPMS* is its ability to test the effectiveness of the complex estimation algorithms. Though the *simulatorAPMS* defaults to *apComplex* as the complex estimator, the user can chose to use any other complex estimator.

```
> data(runSimEX)
> data(vBaitsEX)
> runAPComplex(runSimEX, vBaitsEX)
```

```
[1] "Finding Initial Maximal BH-complete Subgraphs"
[1] "Combining Complex Estimates"
        Complex1 Complex2 Complex3 Complex4 Complex5 Complex6
YAL015C        1        0        0        1        0        0
YAL017W        0        0        0        0        1        0
YAL021C        1        0        0        0        0        0
YAL036C        0        1        0        0        0        0
YAR003W        0        0        1        0        0        0
YAR007C        1        1        1        1        1        1
YAR019C        0        0        0        0        0        1
YBL021C        0        0        0        0        1        1
YBL026W        0        0        0        0        0        1
YBL036C        0        0        0        0        1        1
YBL049W        0        0        0        0        0        1
YBL056W        0        0        0        1        0        1
YBL088C        0        0        0        0        0        1
YBR055C        0        0        0        1        0        1
YBR059C        0        0        0        0        0        1
YBR082C        0        0        0        1        0        1
YBR083W        0        0        0        1        1        1
YBR088C        0        0        0        0        0        1
YBR094W        0        0        0        1        0        1
YBR109C        0        0        0        1        0        1
YBR114W        0        0        0        0        1        1
YBR125C        0        0        0        0        0        1
YBR130C        0        0        0        1        0        1
YML014W        0        0        0        0        1        0
YHR050W        0        0        0        0        1        1
YGL015C        0        0        0        0        1        1
YOL145C        0        0        0        0        1        0
YML101C        0        0        0        0        0        1
```

The output from *apComplex* is the incidence matrix representation of the bipartite graph of proteins to protein complexes. This output is an estimate for the in silico model organism

A^*. After calculating this esimate, it is necessary to ascertain how accurate this estimate is - either with respect to the in silico model organism or compared to other estimation algorithms.

4 Statistical Tools

The last component of *simulatorAPMS* is its statistical tools. It is necessary to compare the simulated test results with the in silico model organism's true state of nature, A. When we have calculated an estimate, $\hat{A}$ using *apComplex*, the first step in the comparison between A and $\hat{A}$ is calculating three statistics between the complexes $C_i \in A$ and $K_j \in \hat{A}$: (1) $C_i \cap K_j$; (2) $C_i \setminus K_j$; and (3) $K_j \setminus C_i$. To calculate these three statistics, we call the `runCompareComplex()` function.

```
> data(simEX)
> data(APComEX)
> runCompareComplex(simEX, APComEX)

$intersect
       Complex1 Complex2 Complex3 Complex4 Complex5 Complex6
MBME1         1        1        0        0        0        0
MBME2         1        1        0        0        0        0
MBME3         1        1        0        0        0        0
MBME4         0        0        0        0        1        0
MBME5         1        1        1        1        1        1
MBME6         0        0        0        0        0        1
MBME7         0        0        0        0        0        1
MBME8         0        0        0        0        0        1
MBME9         0        0        0        0        0        1
MBME10        0        0        0        0        0        0

$cminusk
       Complex1 Complex2 Complex3 Complex4 Complex5 Complex6
MBME1         0        0        1        1        1        1
MBME2         0        0        1        1        1        1
MBME3         0        0        1        1        1        1
MBME4         1        1        1        1        0        1
MBME5         0        0        0        0        0        0
MBME6         1        1        1        1        1        0
MBME7         1        1        1        1        1        0
MBME8         1        1        1        1        1        0
MBME9         1        1        1        1        1        0
MBME10        1        1        1        1        1        1

$kminusc
       Complex1 Complex2 Complex3 Complex4 Complex5 Complex6
MBME1         2        8        2        2       10       21
```

```
MBME2        2        8        2        2       10       21
MBME3        2        8        2        2       10       21
MBME4        3        9        2        2        9       21
MBME5        2        8        1        1        9       20
MBME6        3        9        2        2       10       20
MBME7        3        9        2        2       10       20
MBME8        3        9        2        2       10       20
MBME9        3        9        2        2       10       20
MBME10       3        9        2        2       10       21
```

The return value of `runCompareComplex` is a list containing three matrices. The first component of the list is a matrix of intersection values between the complexes of A and $\hat{A}$. The second and third components are a matrix of the difference values between the complexes of A and $\hat{A}$ and a matrix of the difference values between the complexes of $\hat{A}$ and A respectively. Once we have caluculated these statistics, we can compute similarity measures between the complexes of A and $\hat{A}$.

The first method involves using the Jaccard similarity coefficient. This coefficient is a simple measure of simularity between two complexes by taking the ratio of overlapping sections of the complexes to combining the two complexes. For complex C_i and K_j, we define the Jaccard Matrix as JC with the $\{i, j\}th$ entry as

$$JC_{ij} = \frac{|C_i \cap K_j|}{|C_i \cup K_j|} \tag{1}$$

To call the `JaccardCoef` function, we only need the return value of the `runCompareComplex` function:

```
> data(simEX)
> data(APComEX)
> CC = runCompareComplex(simEX, APComEX)
> JaccardCoef(CC)

        Complex1  Complex2 Complex3 Complex4 Complex5   Complex6
MBME1  0.3333333 0.1111111      0.0      0.0      0.0 0.00000000
MBME2  0.3333333 0.1111111      0.0      0.0      0.0 0.00000000
MBME3  0.3333333 0.1111111      0.0      0.0      0.0 0.00000000
MBME4  0.0000000 0.0000000      0.0      0.0      0.1 0.00000000
MBME5  0.3333333 0.1111111      0.5      0.5      0.1 0.04761905
MBME6  0.0000000 0.0000000      0.0      0.0      0.0 0.04761905
MBME7  0.0000000 0.0000000      0.0      0.0      0.0 0.04761905
MBME8  0.0000000 0.0000000      0.0      0.0      0.0 0.04761905
MBME9  0.0000000 0.0000000      0.0      0.0      0.0 0.04761905
MBME10 0.0000000 0.0000000      0.0      0.0      0.0 0.00000000
```

The Jaccard index has its benefits. The main benefit of the Jaccard index is that it is a very intuitive statistic. It is clear that if two complexes have almost indentical composition, the similarity measure ought to reflect this. It is pretty easy to see that if the two complexes

are identical, the Jaccard index is one, and if they are completely different, the index is zero. Because of its relatively easy structure, the index is a good measure of similarity. The matrix of Jaccard coefficients above has its rows indexed by the complexes of A and the columns indexed by the estimate $\hat{A}$.

The second method uses a variant of the Kullbach-Liebler formula to calculate the independence of two probability distributions. For a random protein p, we can calculate three probabilities: 1. the probability that p is in the complex C_i; 2. the probability that p is in the complex K_j; and 3. the probability that p is in both the complexes C_i and in K_j. With these three probabilities, we can calculate the degree of independence of (1) and (2) with respect to (3). Independence implies that the estimate is not be closely aligned to the truth. To call the `compIndep()`, we need three parameters, the matrix of the bipartite graph of A, the matrix of the bipartite graph of $\hat{A}$, and the intersection matrix of the function `runCompareComplex()`.

```
> data(simEX)
> data(APComEX)
> CC = runCompareComplex(simEX, APComEX)
> compIndep(simEX, APComEX, CC[[1]])
```

```
         Complex1   Complex2  Complex3  Complex4   Complex5    Complex6
MBME1  0.08481054 0.04086605 0.0000000 0.0000000 0.00000000 0.000000000
MBME2  0.08481054 0.04086605 0.0000000 0.0000000 0.00000000 0.000000000
MBME3  0.08481054 0.04086605 0.0000000 0.0000000 0.00000000 0.000000000
MBME4  0.00000000 0.00000000 0.0000000 0.0000000 0.03665163 0.000000000
MBME5  0.08481054 0.04086605 0.1010291 0.1010291 0.03665163 0.006974135
MBME6  0.00000000 0.00000000 0.0000000 0.0000000 0.00000000 0.006974135
MBME7  0.00000000 0.00000000 0.0000000 0.0000000 0.00000000 0.006974135
MBME8  0.00000000 0.00000000 0.0000000 0.0000000 0.00000000 0.006974135
MBME9  0.00000000 0.00000000 0.0000000 0.0000000 0.00000000 0.006974135
MBME10 0.00000000 0.00000000 0.0000000 0.0000000 0.00000000 0.000000000
```

After we have calculated a similarity measures, we need to align complexes of A to complexes of $\hat{A}$. To find an optimal alignment, we employ a version of a greedy algorithm we call `compBijection`. We find the largest entry in the similarity matrix (either Jaccard or the Kullbach-Liebler); its row and column correspond to particular complexes in A and in the estimate $\hat{A}$, and so we align these complexes. We then delete this row and column from the matrix and recursively call `compBijection()` on the smaller matrix. If there is a tie in one row (or one column), we chose larger complexes to align first, since the relative size of the complexes is an important indicator of how well the estimation algorithm works. To call the alignment function, we need three parameters: the matrix of A, the matrix of $\hat{A}$, and either the Jaccard Matrix or the Kullbach-Liebler Matrix:

```
> data(simEX)
> data(APComEX)
> CC = runCompareComplex(simEX, APComEX)
> JC = JaccardCoef(CC)
> runAlignment(simEX, APComEX, JC)
```

```
  ISMO-Complexes Est-Complexes Coefficient
1 "MBME5"        "Complex3"    "0.5"
2 "MBME1"        "Complex1"    "0.333333333333333"
3 "MBME2"        "Complex2"    "0.111111111111111"
4 "MBME4"        "Complex5"    "0.1"
5 "MBME6"        "Complex6"    "0.0476190476190476"
6 "MBME3"        "Complex4"    "0"
```

The output of the `runAlignment` function is a matrix where the row contains the alignment of the complexes of A and $\hat{A}$ with the similarity coefficient (whichever one the user decided to use) is given in the third column. Thus is it easy to see how well the estimation algoithm performed the complexes of the estimate are aligned with those of the ISI. There is one particular caveat for this function; the greedy algorithm is known not to return optimal alignments is some pathelogical cases though in most cases it does.

Where do we go

The *simulatorAPMS* is developed for many reasons, one of which is to test the significance of interactome estimation algorithms. The second part of the package uses the estimation algorithm *apComplex* as the tool to predict the ISI, but *simulatorAPMS* can just as easily test significance of any estimation algorithm. In fact, *simulatorAPMS* could compare and contrast the estimations of different estimation algorithms under variable conditions, i.e. different rates of FP/FN, using different baits, etc.

One of the most substantial elements to *simulatorAPMS* is that if an estimation algorithm can be shown to be highly accurate and statistically significant, we would then have a reasonably clear description of the interactome. Bait selection for AP-MS techonology is highly related to how much information one can discern from the interactome. The more we understand about the interactome, the more intelligent we can make our bait selections, and thus we gan garner more detail from the actual wet-lab experiment.

References

Gavin, A.-C., et al. (2002) Functional organization of the yeast proteome by systematic analysis of protein complexes, *Nature*, **415**, 141-147.

Ho, Y., et al. (2002) Systematic identification of protein complexes in *Saccharomyces cerevisiae* by mass spectrometry, *Nature*. **415**, 180-183.

Jiang, Y., Broach, J. (1999) Tor proteins and protein phosphatase 2A reciprocally regulate Tap42 in controlling cell growth in yeast, *EMBO Journal*, **18**, 2782-2792.

Scholtens, D., Gentleman, R. (2004) Making sense of high-throughput protein-protein interaction data, *Statistical Aplications in Genetics and Molecular Biology*, **3**, Article 39.

Scholtens, D., Vidal, M., Gentleman, R. Local modeling of global interacome networks, *Submitted*.