RpsiXML: Application Examples

Jitao David Zhang and Tony Chiang

May 19, 2021

Abstract

RpsiXML allows the communication between protein interaction data stored in PSI-MI XML format and the statistical and computational environment of R and Bioconductor. In the vignette *RpsiXML*, we introduced how to read in PSI-MI XML 2.5 files with *RpsiXML*. In this vignette, we illustrate the use of *RpsiXML* with example. These applications demonstrate the power of the package in analyzing protein-protein interaction networks (PPIN).

1 Introduction

The systematic mapping of protein interactions by bait-prey techniques contributes a unique and novel perspective on the global picture of cellular machines. We introduce R and Bioconductor package *RpsiXML* as an interface between standarized protein-protein interaction data and statistical and computational environment, and a collection of tools for statistical analysis of graph representations of these data. In this vignette we use an small toy protein protein interaction network to illustrate some aspects of the package usage.

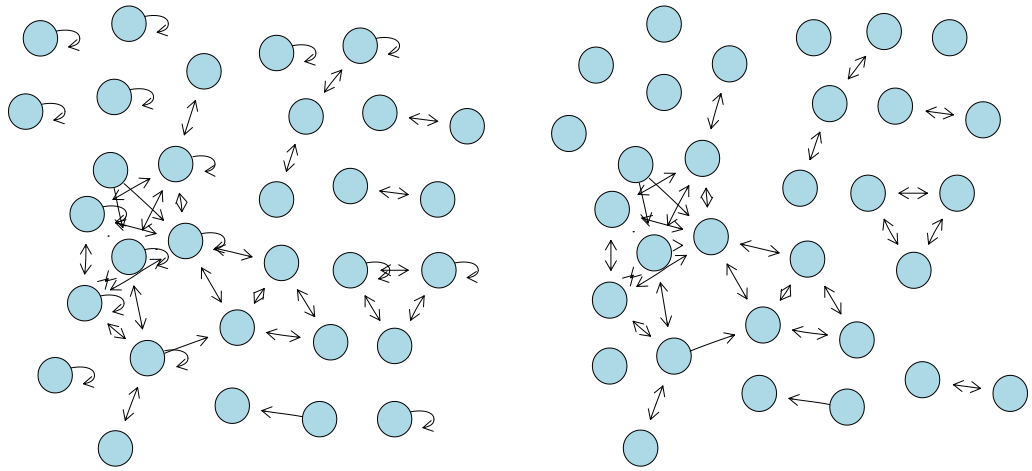
To this end we first load neccessary packages.

```
> library(RpsiXML)
> library(ppiStats)
> library(Rgraphviz)
> library(RBGL)
```

2 Statistics of Protein-Protein Interaction Networks

Upon reading the protein-protein interaction data into R environment, one intuitive approach to study it is to query the statistics of the network. In this section we will use tools implemented in *RpsiXML*, *graph*, *ppiStats* and *RBGL* to study the some statistical characteristics of a sample dataset provided by *IntAct*. And *Rgraphviz* package is used to visualize some results.

```
> xmlDir <- system.file("/extdata/psi25files",package="RpsiXML")
> intactxml <- file.path(xmlDir, "intact_2008_test.xml.gz")
> x <- psimi25XML2Graph(intactxml, INTACT.PSIMI25, verbose=FALSE)
```



(a) Original graph (b) Self-loops removed

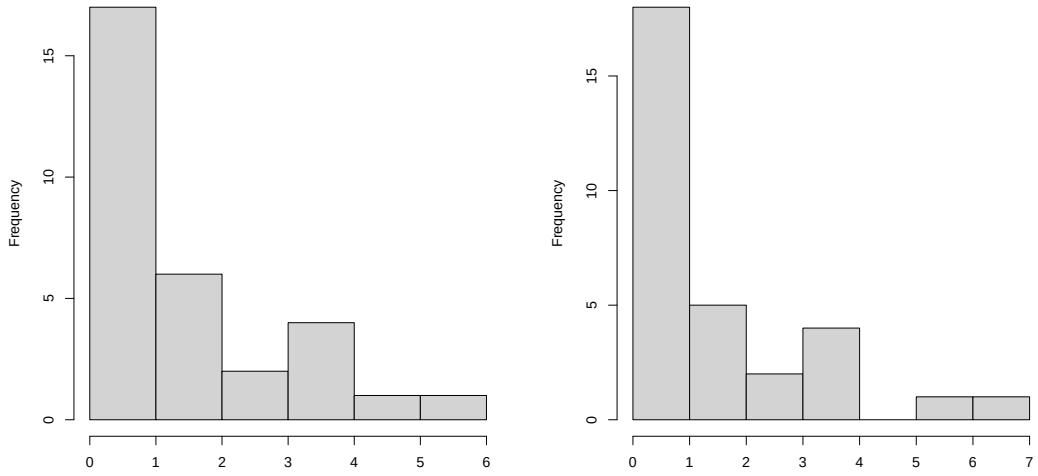
Figure 1: Sample dataset from *IntAct* data repository, before and after removing self-loops.

The graph contains 31 proteins (nodes) and 70 interactions (edges), as visualized in Figure 1(a), which suggests there are quite a few loops, namely edges that connect a vertex to itself. The proteins with such a self-loop interact with themselves, for example by forming dimers (or polymers). We count how many proteins have such an attribute.

```
> isSelf <- function(g) {
+   ns <- nodes(g)
+   sapply(ns, function(x) x %in% adj(g, x)[[1]])
+ }
> isSelfLoop <- isSelf(x)
> selfCount <- sum(isSelfLoop)
> print(selfCount)

[1] 16
```

The results show that 16 proteins interact with themselves. In the next steps, these self-loops are removed, since they raise special issues of certain graph statistics like degree



(a) Out-degree (b) In-degree

Figure 2: Histograms of degree distribution in the graph

for which we don't delve into details. The graph after removing loops are visualized in Figure 1(b).

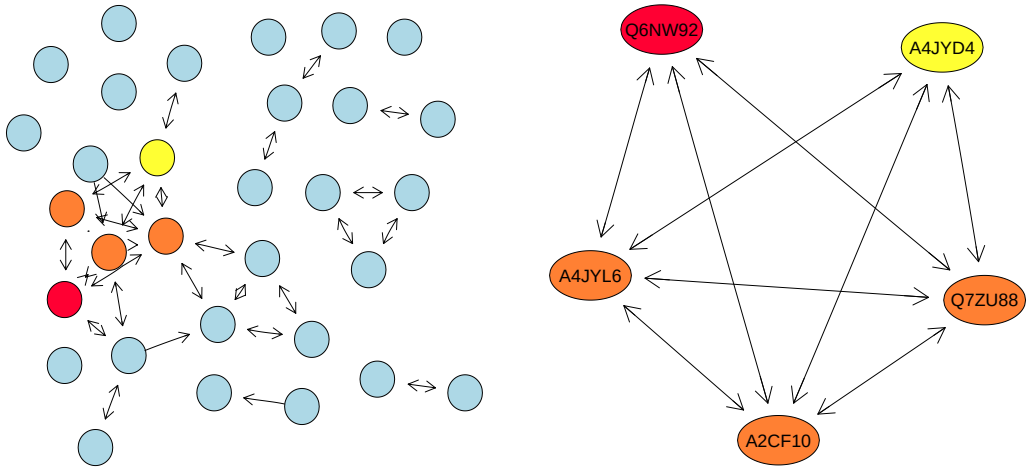
Even though it is a small network, the sample protein-protein interaction network seems to be like a scale-free network, whose degree distribution follows a power law, or at least skewed, as seen in Figure 2. Studies on larger protein-protein interaction networks suggests the human interactome may be of a scale-free network, although there is also discussion whether the interactome is rather geometric. We refer interested readers to related references.

One problem of special interest is to find cliques in protein interaction networks. A *clique* is a complete subgraph, i.e., there is an edge between every pair of vertices. *Maximum Clique* problem is to find the largest clique in a graph. We use `maxClique` function implemented in *RBGL* to find all cliques in the example graph.

```
> xu <- ugraph(xn)
> cls <- maxClique(xu)$maxCliques
> cs <- sapply(cls,length)
> cls[cs==max(cs)]

[[1]]
[1] "A4JYL6" "Q7ZU88" "A2CF10" "Q6NW92"
```

```
[[2]]
[1] "A4JYL6" "Q7ZU88" "A2CF10" "A4JYD4"
```



(a) sample graph (b) two cliques

Figure 3: Two cliques of size 4 found in the example network, colored with yellow (A4YJD4) and red (Q6NW92) individually. The two cliques share three common nodes, whose color is orange due to blending.

In graph `xn`, 22 cliques are found. 2 of them is of size four, and 5 of size three. Figure 3 illustrate the two 4-cliques in the graph. The cliques may be explained by functional protein complexes, and the sharing nodes between cliques could be forming core facility of the complex.

3 Assess reciprocity of interactions

Bait to prey systems allow for the testing of an interaction between a pair of proteins in two directions. If bi-directionally tested, we anticipate the result as both positive or both negative. Failure to attain reciprocity indicates some form of error.

Here we assess the reciprocity of the interactions in the example network. We use the `assessSymmetry` function implemented in *ppiStats* to access the symmetry.

```
> sym <- assessSymmetry(xn, bpGraph=TRUE)
> head(sym$deg)
```

```
       nr no ni
A0S0K7  0  0  0
A2CF10  4  0  0
A3KPA0  1  0  0
A4JYD4  4  0  0
A4JYD8  2  0  0
A4JYF6  2  0  0
```

The `deg` element in the returned list is a $3 \times n$ matrix. The rows are indexed by each protein. Three columns gives the number of reciprocated edges, unreciprocated out-edges and unreciprocated in-edges, respectively.

Among the 31 nodes, TRUE nodes have all reprocated edges.4 edges out of 54are unidirectional, that is, may indicate some form of error.

For demonstration let us assume now all the protein in the network have been tested twice, e.g. both as viable bait and as viable prey. Then one could used method of moments to estimate false positive and false negative error probablities. The model is described in the vignette *Stochastic and systematic errors in PPI data, by looking at unireciprocated in- or out-edges* by W.Hubaer, T.Chiang and R.Gentleman. The estimated false positive and false negative rates are visualized in Figure 4

The estimated values of false positive (p_{FP}) and false negative (p_{FN}) rate are very small, this can be explained by the observation that most edges in the example graph are reprocated.

4 Discussion

We have shown how to use *RpsiXML* package to study protein-protein interaction data with statistical and mathematical tools implemented in R and Bioconductor with several examples.

This application example is dynamic, that is, we will implement new examples or revise old ones from time to time to demonstrate the use of package, depending on the feedback of users.

5 Acknowledgements

We would like to thank Wolfgang Huber, Robert Gentleman, Denise Scholtens, Sandra Orchard, Nick Luscombe, and Li Wang for their very helpful and insightful comments on both the software. We would like to thank the curators of *IntAct*, *MINT*, *DIP*, *HPRD*, *BioGRID*, *MatrixDB*, *CORUM*, and *MPact* in working with us and by guiding us through their molecular interaction repositories.

```
> nint <- 49:56
> nrec <- sum(deg[,1])
> nunr <- sum(deg[,2])
> ntot <- nrow(deg)
> est <- estErrProbMethodOfMoments(nint=nint, nrec=nrec, nunr=nunr, ntot=ntot)
> plot(est[, c("pfp2", "pfn2")], type="l", col="orange", lwd=2,
+      xlab=expression(p[FP]), ylab=expression(p[FN]),
+      xlim=c(-0.001, 0.005), ylim=c(-0.001, 0.045))
> abline(h=0, v=0, lty=2)
```

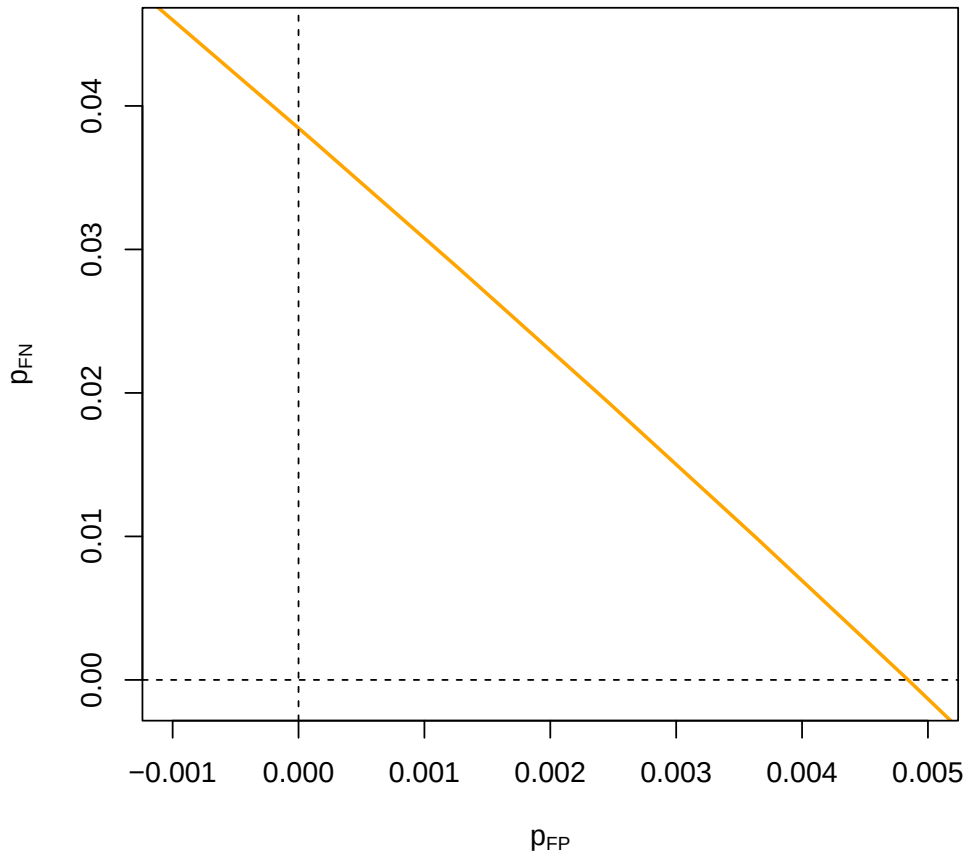


Figure 4: False positive (p_{FP}) and false negative (p_{FN}) rate of the sample graph estimated by the method of moments. The relative low false positive and false negative rates can be explained by the fact that most edges in the example graph are reprocated.

6 Session Info

The script runs within the following session:

```
R version 4.1.0 RC (2021-05-16 r80304)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Mojave 10.14.6

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
 [1] grid      parallel  stats4    stats     graphics  grDevices utils
 [8] datasets  methods   base

other attached packages:
 [1] Rgraphviz_2.37.0     ppiStats_1.59.0      ppiData_0.29.0
 [4] lattice_0.20-44      ScISI_1.65.0         apComplex_2.59.0
 [7] RBGL_1.69.0          graph_1.71.0         annotate_1.71.0
[10] GO.db_3.13.0         AnnotationDbi_1.55.0 IRanges_2.27.0
[13] S4Vectors_0.31.0     Biobase_2.53.0       BiocGenerics_0.39.0
[16] RpsiXML_2.35.0       XML_3.99-0.6

loaded via a namespace (and not attached):
 [1] KEGGREST_1.33.0        genefilter_1.75.0      splines_4.1.0
 [4] vctrs_0.3.8            Category_2.59.0        blob_1.2.1
 [7] survival_3.2-11        rlang_0.4.11           DBI_1.1.1
[10] bit64_4.0.5            RColorBrewer_1.1-2     GenomeInfoDbData_1.2.6
[13] zlibbioc_1.39.0        Biostrings_2.61.0      memoise_2.0.0
[16] fastmap_1.1.0          GenomeInfoDb_1.29.0    GSEABase_1.55.0
[19] Rcpp_1.0.6             xtable_1.8-4           cachem_1.0.5
[22] XVector_0.33.0         bit_4.0.4              png_0.1-7
[25] hypergraph_1.65.0      tools_4.1.0            bitops_1.0-7
[28] RCurl_1.98-1.3         RSQLite_2.2.7          org.Sc.sgd.db_3.13.0
[31] crayon_1.4.1           pkgconfig_2.0.3        Matrix_1.3-3
[34] httr_1.4.2             rstudioapi_0.13        R6_2.5.0
[37] compiler_4.1.0
```