RpsiXML: An R programmatic interface with PSI

Jitao David Zhang and Tony Chiang

May 19, 2021

Abstract

We demonstrate the use and capabilities of the software package *RpsiXML* by examples. Thie package provides a programmatic interface with those databases which adhere to the PSI-MI XML2.5 standardization for molecular interactions. Each experimental dataset from any of the databases can be read into R and converted into a *psimi25Graph* object upon which computatational analyses can be conducted.

1 Introduction

Molecular interactions play an important role in the organizational and functional hierarchy of cells and tissues. Such molecular interaction data has been made publicly available on a wide variety of public databases. Statistical and computational analysis of these datasets necessitates the automated capability of downloading, extracting, parsing, and converting these data into a uniform structure.

Recently, the *Protein Standardization Initiative* has developed the PSI-MI 2.5 XML schema for documenting molecular interaction data. While XML is a particularly good format for data storage and exchange, it is less amenable to compuatational analysis. The contents of the XML files need to be parsed and transformed in structures upon which computation analysis is more feasible and apropros.

The Bioconductor software package *RpsiXML* serves as a programmatic interface between the R statistical environment and the PSI-MI 2.5 XML files. This software should be able to parse the XML files from any database which implement a valid PSI-MI 2.5 schema; currently, the databases that are supported by *RpsiXML* are:

1. IntAct
2. MINT
3. DIP
4. HPRD

5. BioGRID
6. MIPS/CORUM
7. MatrixDB
8. MPact

We plan to support other databases which are now porting to the PSI-MI XML2.5 schema.

In this vignette, we demonstrate the basic functionalities of *RpsiXML*.

2 Preliminaries

2.1 Obtaining the XML Files

Each of the data repositories has its own FTP or download site from which the PSI-MI 2.5 XML files can be obtained. Here we list each database as well as the location of each FTP or download site:

IntAct	*ftp://ftp.ebi.ac.uk/pub/databases/intact/current/psi25*
Mint	*ftp://mint.bio.uniroma2.it/pub/release/psi/2.5/current/*
DIP	*http://dip.doe-mbi.ucla.edu/dip/Download.cgi*
HPRD	*http://www.hprd.org/download*
The BioGRID	*http://www.thebiogrid.org/downloads.php*
CORUM/MIPS	*http://mips.gsf.de/genre/proj/corum*
MatrixDB	*not published yet*
MPact	*ftp://ftpmips.gsf.de/yeast/PPI*

The DIP repository requires one to create a login account before accessing the data. Each of the PSI-MI XML2.5 files should be downloaded to the local file directory which is accessible by the R environment.

We have downloaded XML files from each of the molecular interaction data repository listed above; we have, however, modified these datasets by truncating most of the data so as to provide the user with helpful sample XML files that is not to large. If the user has the source package, these sample XML files can be found within the *inst/estdata/psi25files/* directory of the package. Otherwise, once the package has been loaded, each file can be loaded into an R session with the following calls:

(First we load the package) (Then we create a path to the files)

```
> xmlDir <- system.file("/extdata/psi25files",package="RpsiXML")
> gridxml <- file.path(xmlDir, "biogrid_200804_test.xml.gz")
```

The first line of code above creates a path to the desired directory while the second line constructs the path to the *biogrid* sample file in a platform independent manner.

Table 2.1 gives the names of the sample XML files with the corresponding database repository:

IntAct	intact_2008_test.xml intact_complexSample.xm
Mint	mint_200711_test.xml
DIP	dip_2008_test.xml
HPRD	hprd_200709_test.xml
The BioGRID	biogrid_200804_test.xml
CORUM/MIPS	mips_2007_test.xml
MatrixDB	matrixdb_20080609.xml

It should be noted that *IntAct* has two different kinds of sample XML files: the "test" file stores the bait-prey interaction data while the "complex" file stores manually curated protein complex membership information. For the scope of this vignette, we will focus on the *IntAct* files as our working examples, but we encourage the reader to work explore the other files as well.

2.2 XML into R

There are two different methods for parsing the PSI-MI XML2.5 files:

2.2.1 parsePsimi25Interaction

The first method relies on the function `parsePsimi25Interaction` which systematically searches over the XML tree structure and returns the fields (nodes) of interest. First we obtain the XML file we wish to parse:

```
> intactxml <- file.path(xmlDir, "intact_2008_test.xml.gz")
> intactComplexxml <- file.path(xmlDir,"intact_complexSample.xml.gz")
```

and then we parse the file using the `parsePsimi25Interaction` function:

```
> intactSample <- parsePsimi25Interaction(intactxml, INTACT.PSIMI25,verbose=FALSE)
> intactComplexSample <- parsePsimi25Complex(intactComplexxml, INTACT.PSIMI25,verbose=FALSE
```

The two arguments taken by `parsePsimi25Interaction` is:

1. A character vector with the relative path to the file of interest (can also be an URL).

2. A supported data repository source file R object.

Because each database repository implements the PSI-MI XML2.5 standards in slightly varing ways, it is necessary to track these differences and to tell the parsing function which implementation to expect. Each of the database supported by RpsiXML has its own corresponding source class object (`INTACT.PSIMI25, MINT.PSIMI25, DIP.PSIMI25, HPRD.PSIMI25, BIOGRID.PSIMI25,` and `MIPS.PSIMI25`).

The output from the `parsePsimi25Interaction` is an object of type `psimi25InteractionEntry` or `psimi25ComplexEntry` depending on the type of input XML file. Each is a class used to carry all of the information obtained from the XML files be it interaction or complex. From each of these classes, we can obtain various types of information:

(*From the intactSample object*)

```
> interact <- interactions(intactSample)[1:3]
> interact

[[1]]
interaction ( EBI-1580529 ):
 ----------------------------------------------------------------
 [ source database ]: intact
 [ source experiment ID ]: EBI-1580529
 [ interaction type ]: spr
 [ experiment ]: pubmed 18296487
 [ participant ]: A4JYH2 A4JYL6
 [ bait ]: 8
 [ bait UniProt ]: A4JYL6
 [ prey ]: 4
 [ prey UniProt ]: A4JYH2
 [ confidence value ]: NA

[[2]]
interaction ( EBI-1580520 ):
 ----------------------------------------------------------------
 [ source database ]: intact
 [ source experiment ID ]: EBI-1580520
 [ interaction type ]: spr
 [ experiment ]: pubmed 18296487
 [ participant ]: Q7ZU88 A4JYH2
 [ bait ]: 13
 [ bait UniProt ]: Q7ZU88
 [ prey ]: 4
 [ prey UniProt ]: A4JYH2
 [ confidence value ]: NA
```

```
[[3]]
interaction ( EBI-1580511 ):
 ----------------------------------------------------------------
 [ source database ]: intact
 [ source experiment ID ]: EBI-1580511
 [ interaction type ]: spr
 [ experiment ]: pubmed 18296487
 [ participant ]: Q5J1R9 Q7SX76
 [ bait ]: 25
 [ bait UniProt ]: Q7SX76
 [ prey ]: 21
 [ prey UniProt ]: Q5J1R9
 [ confidence value ]: NA
```

```
> organismName(intactSample)[1:3]
```

```
[1] "Brachydanio rerio" "Homo sapiens"      "Rattus norvegicus"
```

```
> releaseDate(intactSample)
```

```
[1] "2008-03-28"
```

(*Looking within each interaction*)

```
> lapply(interact, participant)[1:3]
```

```
[[1]]
  A4JYH2   A4JYL6
"A4JYH2" "A4JYL6"

[[2]]
  Q7ZU88   A4JYH2
"Q7ZU88" "A4JYH2"

[[3]]
  Q5J1R9   Q7SX76
"Q5J1R9" "Q7SX76"
```

```
> sapply(interact, bait)[1:3]
```

```
[1] "A4JYL6" "Q7ZU88" "Q7SX76"
```

```
> sapply(interact, prey)[1:3]

[1] "A4JYH2" "A4JYH2" "Q5J1R9"

> sapply(interact, pubmedID)[1:3]

         2          2          2 
"18296487" "18296487" "18296487" 
```

Most of the information from the extracted data is self-explanatory; we will, however, highlight some important pieces. The `interact` object is the output of the `interactions` method which provides all the pertinent details for each interaction. This object is a list of the *psimi25Interaction* class. Upon each individual *psimi25Interaction* class, there exists methods to extract the individual pieces of information. The `participant` method returns the two proteins which were involved in the interaction. If available, the `bait` and `prey` methods returns the proteins which serve as their namesake. Each interaction is also indexed by the pubmed ID which can also be extracted.

(*From the complexSample object*)

```
> comp <- complexes(intactComplexSample)[1:2]
> sapply(comp, complexName)

[1] "BCL-2 homodimer"        "BAD:BCL-2 heterodimer"
```

2.2.2 psimi25XML2Graph

While the parsing can be accomplished via the `parsePsimi25Interaction` function, the output of this function is not readily accessible for computational analysis. For this case, the function `psimi25XML2Graph` is a better choice. For instance we can construct the bait-prey graph from the *IntAct* sample XML file:

```
> intactGraph <- psimi25XML2Graph(intactxml, INTACT.PSIMI25,
+                                 type="interaction",verbose=FALSE)

1 Entries found
Parsing entry 1
  Parsing experiments: ..
  Parsing interactors:

  3% =>
  6% ==>
  10% ====>
  13% =====>
  16% ======>
  19% ========>
  23% =========>
  26% ==========>
  29% ============>
  32% =============>
  35% ==============>
  39% ================>
  42% =================>
  45% ==================>
  48% ===================>
  52% =====================>
  55% ======================>
  58% =======================>
  61% ========================>
  65% ==========================>
  68% ===========================>
  71% ============================>
  74% ==============================>
  77% ===============================>
  81% ================================>
  84% ==================================>
  87% ===================================>
  90% ====================================>
  94% ======================================>
  97% =======================================>
  100% ========================================>
  Parsing interactions:
...........................................................................................
```

```
> nodes(intactGraph)
```

```
 [1] "A0S0K7" "A2CF10" "A3KPA0" "A4JYD4" "A4JYD8" "A4JYF6" "A4JYF7" "A4JYG1"
 [9] "A4JYG2" "A4JYG3" "A4JYG4" "A4JYI5" "A4JYK9" "A4JYL3" "A4JYL6" "A4JYL8"
[17] "A4JYM3" "A4JYP5" "A4JYS0" "P04218" "P09326" "Q5J1R9" "Q5W433" "Q6NW92"
[25] "Q7SX76" "Q7ZU88" "Q90413" "Q90YM1" "Q9BZW8" "A4JYH2" "A4JYF3"
```

```
> degree(intactGraph)
```

```
$inDegree
A0S0K7 A2CF10 A3KPA0 A4JYD4 A4JYD8 A4JYF6 A4JYF7 A4JYG1 A4JYG2 A4JYG3 A4JYG4
```

```
     1      5      1      5      2      3      3      3      1      1      1
A4JYI5 A4JYK9 A4JYL3 A4JYL6 A4JYL8 A4JYM3 A4JYP5 A4JYS0 P04218 P09326 Q5J1R9
     2      2      2      8      1      1      1      1      1      1      4
Q5W433 Q6NW92 Q7SX76 Q7ZU88 Q90413 Q90YM1 Q9BZW8 A4JYH2 A4JYF3
     1      5      4      7      1      1      1      0      0

$outDegree
A0S0K7 A2CF10 A3KPA0 A4JYD4 A4JYD8 A4JYF6 A4JYF7 A4JYG1 A4JYG2 A4JYG3 A4JYG4
     1      5      1      5      2      3      3      3      1      1      1
A4JYI5 A4JYK9 A4JYL3 A4JYL6 A4JYL8 A4JYM3 A4JYP5 A4JYS0 P04218 P09326 Q5J1R9
     2      2      2      7      1      1      1      1      1      1      5
Q5W433 Q6NW92 Q7SX76 Q7ZU88 Q90413 Q90YM1 Q9BZW8 A4JYH2 A4JYF3
     0      5      3      6      1      1      1      2      1
```

And we can also build a protein complex membership hyper-graph from the sample complex XML file:

```
> intactHG <- psimi25XML2Graph(intactxml, INTACT.PSIMI25, type="complex",verbose=FALSE)
```

There is a caveat for the function `psimi25XML2Graph`; it does not decipher between the data within the XML file insomuch that if it is all bait/prey, then it will generate one large graph. If you are sure that you would like to take all the data and create one large graphical structure, then a call to this function is appropriate. Otherwise, if some of the data within the XML files should be separated, a call to this function is not recommended.

2.2.3 separateXMLDataByExpt

A different way to transform the XML data into graphs is to call the `searapteXMLDataByExpt` function. This function will parse bait-prey data into distinct graphs indexed by the pubmed IDs. Note that this function cannot be called upon XML files that record manually curated protein complexes since there is rarely an associated pubmed ID for this type of data.

```
> graphs <- separateXMLDataByExpt(xmlFiles = intactxml, psimi25source=INTACT.PSIMI25,
+                                  type="indirect", directed=TRUE, abstract=TRUE,
+                                  verbose=FALSE)

1 Entries found
Parsing entry 1
  Parsing experiments: ..
  Parsing interactors:
```

```
3% =>
6% ==>
10% ====>
13% =====>
16% ======>
19% ========>
23% =========>
26% ==========>
29% ============>
32% =============>
35% ==============>
39% ================>
42% =================>
45% ==================>
48% ===================>
52% =====================>
55% ======================>
58% =======================>
61% ========================>
65% ==========================>
68% ===========================>
71% ============================>
74% ==============================>
77% ===============================>
81% ================================>
84% ==================================>
87% ===================================>
90% ====================================>
94% ======================================>
97% =======================================>
100% ========================================>
Parsing interactions:
...........................................................................................
```

Now we look at the input parameters:

- xmlFiles - a character vector of the relative path to the PSI-MI XML2.5 files relative to the R working directory.
- psimi25source - A supported data repository source R object
- type - character either "direct" or "indirect" signaling the type of interaction wanted

- directed - a logical to determine if the graph returned is either directed or not
- abstract - a logical to determine whether or not the function should also get the abstract information for each dataset from NCBI

```
> graphs

$`18296487`
A graphNEL graph with directed edges
Number of Nodes = 31
Number of Edges = 70
[1] "psimi25Graph"
attr(,"package")
[1] "RpsiXML"
```

```
> abstract(graphs$`18296487`)

An object of class 'pubMedAbst':
Title: Large-scale screening for novel low-affinity extracellular
     protein interactions.
PMID: 18296487
Authors: KM Bushell, C Söllner, B Schuster-Boeckler, A Bateman, GJ
     Wright
Journal: Genome Res
Date: Apr 2008
```

It should be noted that if you are going to parse a large number of XML files, it is not recommended to automatically get the abstract information since NCBI has been known to refuse and later ban IP addresses that consistenly demand a high volume of information. For this reason, the `abstract` parameter has been set to FALSE as a default.

One can manually obtain the abstract information as follows:

```
> getAbstractByPMID(names(graphs))

$`18296487`
An object of class 'pubMedAbst':
Title: Large-scale screening for novel low-affinity extracellular
     protein interactions.
PMID: 18296487
Authors: KM Bushell, C Söllner, B Schuster-Boeckler, A Bateman, GJ
     Wright
Journal: Genome Res
Date: Apr 2008
```

3 Converting Node IDs

The bait/prey information (when downloaded and converted into an R graph object) is encoded by the UniProtKB identification schema. UniProtKB appears to be the most universal naming scheme, and so it offers consistency across databases. If there is a need to convert the names of the nodes from the UniProtKB IDs to some other naming scheme, there is two ways of doing so:

- use the R package *biomaRt*
- use the built in method `translateID`

The benefits of using *biomaRt* is that it lets you communicate with Biomart and obtain the latest annotations and translations. The drawback is that it is a non-trivial task and is beyond the scope of this vignette. The drawbacks of `translateID` is that only the naming schemes supported (i.e. arbitrarily chosen) by each database can be supported by *RpsiXML*. The benefit is the ease and simplicity of use.

```
> graphs1 <- translateID(graphs[[1]], to="intact")
> nodes(graphs1)
```

```
 [1] "EBI-1579594" "EBI-1579611" "EBI-1579361" "EBI-1579591" "EBI-1579656"
 [6] "EBI-1579619" "EBI-1579578" "EBI-1579667" "EBI-1579332" "EBI-1579335"
[11] "EBI-1579266" "EBI-1579373" "EBI-1579400" "EBI-1579353" "EBI-1579616"
[16] "EBI-1579370" "EBI-1579304" "EBI-1579664" "EBI-1579603" "EBI-915817"
[21] "EBI-714770"  "EBI-1579627" "EBI-1579381" "EBI-1579483" "EBI-1579511"
[26] "EBI-1579427" "EBI-1579413" "EBI-1579680" "EBI-1580565" "EBI-1579530"
[31] "EBI-1579465"
```

If a particular node cannot be mapped to the naming schema, it will retain the UniprotKB ID.

4 Conclusion

Once the XML files have been downloaded, parsed, and converted into R grpah objects, there are a number of applicable methods and tools available within R and Bioconductor upon which these data graphs can be analyzed. Some (but not all) packages include: *RBGL*, *ppiStats*, *apComplex*, etc.