

# Reading PSI-25 XML file from IntAct with the *Rintact* package

Tony Chiang and Nianhua Li

October 22, 2008

## Abstract

This document serves as a user's vignette to the R package *Rintact*. We present examples of how to use the two main functions of *Rintact*, and also how to take the output data from these two functions and create the input data for statistical methods in proteomic analysis provided by other Bioconductor packages.

## 1 Introduction

*Rintact* is an R package mainly used to parse the PSI-25 files generated by the *IntAct* data repository which collects, curates and stores thousands of protein interactions. Currently, there are two main functions within *Rintact*:

1. `psi25interaction`
2. `psi25complex`

The first function, `psi25interaction`, takes either a PSI-25 XML file from IntAct or an URL containing the web address of where such an XML file can be obtained. The XML file must contain *binary* protein protein interaction data. Example for such data are direct physical interactions, complex co-membership, synthetic genetic interactions. The second function, `psi25complex`, also takes a PSI-25 XML file or URL as an input parameter, but the file must contain protein complex membership information. In principle, these two functions can take any XML file which adheres to the PSI-25 standards. We have constructed these functions, however, to work primarily with the *IntAct* PSI-25 XML files, as there are subtle implementation differences between repositories such as *IntAct* and *DIP*, although both use the PSI-25 standards. In this vignette, we shall demonstrate the use of these functions on the data generated by [Ewing et al.(2007)] and the manually curated protein complexes derived by the *IntAct* curators.

## 1.1 Loading R Libraries

We begin by loading the various R libraries with which we shall use. Our primary focus will be with the *Rintact* package, but we will also examine and exploit statistical methods found in various Bioconductor packages for the analysis of the interaction data obtain from *IntAct*.

```
> library("Rintact")
> library("graph")
> library("Rgraphviz")
> #library("ppiStats")
> library("RBGL")
> library("apComplex")
> library("xtable")
```

## 2 Obtaining the Interaction Information

### 2.1 psi25interaction

We first demonstrate the use of the function `psi25interaction`. We can either download the *IntAct* PSI-25 XML file onto a local directory or we can simply use the URL (of where the file can be obtained) as the input parameter. We have chosen the latter:

```
> url <- system.file("PSI25XML", "interactionSample.xml", package="Rintact")
> ewing <- psi25interaction(url)
```

Once the XML file has been parsed by `psi25interaction`, we can look at its overall structure.

```
> class(ewing)

[1] "interactionEntry"
attr(,"package")
[1] "Rintact"
```

We can see that the output of `psi25interaction` is an instance of the class *interactionEntry*. This class has 5 slots:

```
> slotNames(ewing)

[1] "organismName" "taxId"          "releaseDate"   "interactors"   "interactions"
```

Three of them contain simple character vectors:

```
> ewing@organismName
```

```
[1] "Homo sapiens"                "Human adenovirus E"  
[3] "Human papillomavirus type 1a"
```

```
> ewing@taxId
```

```
[1] "9606"    "130308" "10583"
```

```
> ewing@releaseDate
```

```
[1] "2007-04-27"
```

*organismName* records all the organisms for which interactions were assayed. For each organism, we have also included its taxonomy identification code. Because *IntAct* does not currently version its weekly release, we have added the *releaseDate* as a time stamp to act as a surrogate for the version number.

Let us investigate the structure of the *interactions* slot. This slot contains a list which holds all the binary interactions given within the XML file (along with information about each particular interaction). Each element of the list is an instance of the class *intactInteraction* class. This class has 9 slots:

```
> length(interactions(ewing))
```

```
[1] 5
```

```
> class(interactions(ewing)[[1]])
```

```
[1] "intactInteraction"  
attr(,"package")  
[1] "Rintact"
```

```
> interactions(ewing)[[1]]
```

```
interaction ( EBI-987168 ):
```

```
-----  
[ interaction type ]: pull down  
[ experiment ]: pubmed 16249186 , intact EBI-965562  
[ confidence value ]: NA  
[ bait ]: EBI-491274  
[ prey ]: EBI-448924  
[ neutral component ]: NA  
[ inhibitor ]: EBI-987160
```

```
> slotNames(interactions(ewing)[[1]])
```

```
[1] "intact"          "interactionType" "expPubMed"       "expIntAct"
[5] "confidenceValue" "bait"           "prey"            "inhibitor"
[9] "neutralComponent"
```

The various slots contain information which is relevant for each individual interaction. The *interactionType* slot details what manner of interaction was found between the bait protein and the prey protein, which are specified in the *bait* and *prey* slots. Another important attribute is the experimental confidence value given in the *confidenceValue* slot. This confidence value is reported by the experimenters; it does not report scores derived by third parties.

We can extract the names of the bait and prey proteins for all of the interactions in the *ewing* dataset:

```
> ewbait <- sapply(interactions(ewing), bait)
> ewprey <- sapply(interactions(ewing), prey)
```

We now have two character vectors, *ewbait* and *ewprey*, that are aligned to each other: the  $i^{th}$  protein in *ewprey* is found by the  $i^{th}$  protein in *ewbait*.

```
> ewbait
```

```
      intactId      intactId      intactId      intactId      intactId
"EBI-491274" "EBI-491274" "EBI-963841" "EBI-491274" "EBI-963841"
```

```
> ewprey
```

```
      intactId      intactId      intactId      intactId      intactId
"EBI-448924" "EBI-448924" "EBI-491274" "EBI-448924" "EBI-765551"
```

The *IntAct* accession codes are useful as unique and uniform identifiers in the *IntAct* repository, but we will usually want to translate them to other identifier schemes such as HUGO gene name Ensembl gene identifier. The PSI-25 XML files from *IntAct* contain a look-up table for this purpose. This look-up table is stored in the *interactors* slot of the *interactionEntry* object *ewing*, in the form of a character matrix. Its rows are indexed by the *IntAct* accession numbers of the molecules in the data structure, and its has 7 columns.

```
> interactors(ewing)
```

	uniprotId	geneName	fullName	locusName
EBI-491274	"P06400"	"RB1"	"Retinoblastoma-associated protein"	NA
EBI-987160	"Q6H1D8"	"E1A"	"E1A"	NA
EBI-448924	"Q01094"	"E2F1"	"Transcription factor E2F1"	NA

EBI-963841	"P06465"	"E7"	"Protein E7"	NA
EBI-765551	"000716"	"E2F3"	"Transcription factor E2F3"	NA
	orfName	organismName		taxId
EBI-491274	NA	"Homo sapiens"		"9606"
EBI-987160	NA	"Human adenovirus E"		"130308"
EBI-448924	NA	"Homo sapiens"		"9606"
EBI-963841	NA	"Human papillomavirus type 1a"	"10583"	
EBI-765551	NA	"Homo sapiens"		"9606"

The *IntAct* accession codes can be translated into any of the associated identifier schemes. Two further properties are given for each molecule: the organism in which the molecule is native and the corresponding taxonomy ID. Most of the interactions found in *IntAct* will be protein-protein interactions; other types of interactions, however, are also stored such as small molecule to protein interactions as well as gene-gene interactions. As a result, there will be times when a molecule cannot be mapped to a locus name or an ORF etc. We also remark that interactions have been tested between proteins of different organisms (i.e human protein against mice). Thus the organism attribute is vital to keep such interactions in the proper context.

Using the look-up table is quick and efficient because of the subsetting functionality of R. For instance, say we would like to translate the following *IntAct* accession codes

```
> wh = ewbait[3:4]
```

into gene names:

```
> interactors(ewing)[wh, "geneName"]
```

```
EBI-963841 EBI-491274
      "E7"      "RB1"
```

### 3 Obtaining Protein Complex Composition Information

Now we will demonstrate the parser function `psi25complex`. We remark here that the protein complexes which this function obtains have been manually curated from literature sources by *IntAct* curators.

The parameters of `psi25complex` are identical to those of `psi25interaction`, while its output only contains 3 slots:

```
> url2 <- system.file("PSI25XML/complexSample.xml", package="Rintact")
> comps <- psi25complex(url2)
> slotNames(comps)

[1] "releaseDate" "interactors" "complexes"
```

Again the *releaseDate* slot serves as a surrogate version number. The *interactors* slot again holds a look-up table that can be used to translate the *IntAct* accession codes. The *complexes* slot is a list of *intactComplex* objects. Each list entry is an instance of the class *intactComplex*, which itself has 7 slots.

```
> length(complexes(comps))

[1] 174

> class(complexes(comps)[[1]])

[1] "intactComplex"
attr(,"package")
[1] "Rintact"

> slotNames(complexes(comps)[[1]])

[1] "intactId"      "shortLabel"    "fullName"      "organismName" "taxId"
[6] "members"      "attributes"
```

These slots describe the multi-protein complex. The three most important ones are *fullName*, *attributes* and *members* slots. The *fullName* slot gives the exact name of the multi-protein complex while the *attributes* slots gives a short description as to the known functionality of the complex. The *interactors* slot gives the members of the complex and their multiplicity.

```
> complexes(comps)[[1]]

complex ( EBI-706546 )
-----
[ short label ]: bcl2_bcl2_human
[ full name ]: BCL-2 homodimer
[ organism ]: Homo sapiens
[ taxonomy ID ]: 9606
[ attributes ]:
curated-complex: Role of homodimer is unclear, may act as reservoir of protein f
or heterodimer formation.
complex-synonym: Bcl2 homodimer; Bcl-2:Bcl-2; Bcl2:Bcl2;
kd: 0.0
[ members ]:
          intActId multiplicity
intactId EBI-77694          2
```

```
> toLatex(sessionInfo())
```

- R version 2.8.0 (2008-10-20), i386-pc-mingw32
- Locale: LC\_COLLATE=English\_United States.1252;LC\_CTYPE=English\_United States.1252;LC\_MON
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, tools, utils
- Other packages: AnnotationDbi 1.4.0, apComplex 2.8.0, Biobase 2.2.0, DBI 0.2-4, graph 1.20.0, hypergraph 1.14.0, org.Sc.sgd.db 2.2.6, RBGL 1.18.0, Rgraphviz 1.20.0, Rintact 1.4.0, RSQLite 0.7-0, XML 1.98-1, xtable 1.5-4
- Loaded via a namespace (and not attached): cluster 1.11.11

Table 1: The output of `sessionInfo` on the build system after running this vignette.

## References

- [Gavin et al.(2007)] Gavin AC, Aloy P, Grandi P, Krause R, Boesche M, Marzioch M, Rau C, Jensen LJ, Bastuck S, Dümpelfeld B, et al. Proteome survey reveals modularity of the yeast cell machinery. *Nature* 2006, **440**:631–636.
- [Ewing et al.(2007)] Ewing EM et al. Large-scale Mapping of Protein-Protein Interactions by Mass Spectrometry. *Molecular Systems Biology* 2007, 3.