RpepXML: an R interface to the pepXML format for peptide identification.

Laurent Gatto
lg390@cam.ac.uk
Cambridge Centre for Proteomics – EBI
www.bio.cam.ac.uk/proteomics – www.ebi.ac.uk

Abstract

This vignette describes the data structures and associated methods and functions to import and inspect peptide identification data in the pepXML format.

Keywords:~mass spectrometry, proteomics, MSMS, identification, XML.

1. Introduction

From the pepXML webpage[1]:

> pepXML is an open data format developed at the SPC/Institute for Systems biology[2] for the storage, exchange, and processing of peptide sequence assignments of MS/MS scans. pepXML is intended to provide a common data output format for many different MS/MS search engines and subsequent peptide-level analyses. Several search engines already have native support for outputting pepXML and converters are available to transform output files to pepXML.

Note that the HUPO Proteomics Standards Initiative[3] (PSI) has also develop an exchange standard for database search results, called mzIdentXML[4]. It is not yet as widely used as pepXML but should supersede the latter. A RmzIdentML library will be developed at a later stage.

Currently, only the generic MSMS identification results of the msms_pipeline_analysis is implemented. This package has currently been tested and developed around Mascot[5] search results.

2. Getting data into R

The functionality of RpepXML will be illustrated with a dummy example file that is provided with the package. It is parsed with the parseMSMSpepXML function, that takes a file name as first parameter. The second one defined whether a progress bar should be displayed. This function returns an object of class MSMSpepXML.

```
> library("RpepXML")
> pepxml <- system.file("extdata","tinySearch.pepXML",package="RpepXML")
> ms <- parseMSMSpepXML(pepfile=pepxml, verbose=FALSE)
> class(ms)
```

[1]http://tools.proteomecenter.org/wiki/index.php?title=Formats:pepXML
[2]http://tools.proteomecenter.org/wiki/index.php?title=Main_Page
[3]http://www.psidev.info/
[4]http://www.psidev.info/index.php?q=node/319
[5]http://www.matrixscience.com/

```
[1] "MSMSpepXML"
attr(,"package")
[1] "RpepXML"
```

3. Data structures and manipulation

The classes implemented mimic a simplified version of the XML structure for the `msms_pipeline_analysis` element (see the (older) docs[6], for a browsable description). The structure of the 4 classes is described below, starting with the main, high-level, data structure. Each class and slots are described in the respective on-line manuals.

The classes are versioned individually and the class version can be retrieved with `classVersion()`.

3.1. MSMSpepXML

```
> getClass("MSMSpepXML")

Class "MSMSpepXML" [package "RpepXML"]

Slots:

Name:           pepFile      searchEngine      sampleEnzyme    searchDatabase
Class:        character         character         character         character

Name:    spectrumQueries .__classVersion__
Class:              list          Versions

Extends: "Versioned"
```

`MSMSpepXML` is the main class that holds general meta-data about the search like the search engine, the sample enzyme and the original database used to query the MSMS spectra. The name of the `pepXML` file also recorded. All the spectrum queries are stored as a list.

```
> ms

Object of class MSMSpepXML
 File: tinySearch.pepXML
 Database: erwinia_pro.fasta
 Engine: MASCOT
 Enzyme: Trypsin
 4 spectrum queries
 4 peptides identified

> pepFile(ms)

[1] "/tmp/RtmpjNJHhZ/Rinst784c78f098e9/RpepXML/extdata/tinySearch.pepXML"

> searchEngine(ms)

[1] "MASCOT"
```

[6] http://sashimi.sourceforge.net/schema_revision/pepXML/Docs/pepXML_v18.html

```
> sampleEnzyme(ms)

[1] "Trypsin"

> searchDatabase(ms)

[1] "erwinia_pro.fasta"
```

The spectrum queries can be accessed using the `spectrumQueries` method or the `[` operator. The former returns a list that can be subsetted as usual. The latter return a list when multiple spectra are selected or and object of class `SpectrumQuery` if one query is provided.

```
> class(spectrumQueries(ms))

[1] "list"

> sq <- ms[3]
> class(sq)

[1] "SpectrumQuery"
attr(,"package")
[1] "RpepXML"
```

3.2. SpectrumQuery

```
> getClass("SpectrumQuery")

Class "SpectrumQuery" [package "RpepXML"]

Slots:

Name:      searchResults        spectrumId         startScan           endScan
Class:              list         character           integer           integer

Name:    precNeutralMass     assumedCharge        queryIndex .__classVersion__
Class:           numeric           numeric           integer          Versions

Extends: "Versioned"
```

`SpectrumQuery` objects hold the information about specific spectra that have been queried. Note that all pieces of information may not be provided in the `pepXML` file.

```
> sq

Object of class SpectrumQuery
 Spectrum id: 186_c660.3871_u662.3625_r58.85
 Start - end scans: 0 - 0
 Precursor neutral mass: 659.3798
 Assumed charge: 1
 Query index: 3
 2 search result(s)
 3 search hits(s)
 6 proteins(s)
```

```
> precNeutralMass(sq)
```

```
[1] 659.3798
```

```
> assumedCharge(sq)
```

```
[1] 1
```

```
> spectrumId(sq)
```

```
[1] "186_c660.3871_u662.3625_r58.85"
```

If any, the search results can be accessed with the `searchResuts` or `[` operator, that works as described above.

```
> sr <- sq[2]
```

3.3. SearchResult

```
> getClass("SearchResult")
```

```
Class "SearchResult" [package "RpepXML"]

Slots:

Name:         searchHits          searchId .__classVersion__
Class:              list           integer          Versions

Extends: "Versioned"
```

The `SearchResult` class holds the information about a specific search result and the hits that have been identified are available as a list of `SearchHit` objects. The latter can be accessed by the `searchHits` method or the `[` operator.

```
> sr
```

```
Object of class SearchResult
 Search id: 2
 2 search hit(s)
```

```
> sh <- sr[1]
```

3.4. SearchHit

```
> getClass("SearchHit")
```

```
Class "SearchHit" [package "RpepXML"]

Slots:

Name:            hitRank          proteins       pepSequence     modifications
```

```
Class:            integer         character          character        data.frame

Name:              scores .__classVersion__
Class:            numeric          Versions

Extends: "Versioned"
```

The actual results are stored as individual `SearchHit` instances. The class records the rank of the hit, the peptide sequence with the previous and next amino acids, possible modifications, associated proteins and scores associated to the hit. The scores are returned as a named vector to retrieve individual scores. The `ionScore` method allows to extract specifically the ion score for Mascot results.

```
> sh

Object of class SearchHit
 Rank: 1
 Sequence: K.TPVSEK.V
 Scores: 34.67 27 22 0.0089
 1 proteins: BSA
 0 modifications

> hitRank(sh)

[1] 1

> pepSequence(sh)

[1] "K.TPVSEK.V"

> scores(sh)

     ionscore identityscore homologyscore        expect
      34.6700       27.0000       22.0000        0.0089

> ionScore(sh)

ionscore
   34.67

> proteins(sh)

[1] "BSA"
```

4. Getting a general overview of the data

The number of spectrum queries and search results or hits can be extracted with the `nSpectrum-Queries`, `nSearchResults` and `nSearchhits` methods. Several methods can also be applied to parent classes to those that actually have the slot to retrieve the slot values from the whole set of objects.

```
> nSearchResults(sq)
```

```
[1] 2

> proteins(sq[1])

                     search_hit
                     "ECA0851"
alternative_protein "ECA3748"

> proteins(sq[2])

                     search_hit                    search_hit1
                          "BSA"                      "ECA0851"
search_hit.alternative_protein search_hit.alternative_protein
                      "ECA3174"                      "ECA3748"

> proteins(sq)

[1] "ECA0851" "ECA3748" "BSA"     "ECA0851" "ECA3174" "ECA3748"
```

`proteinSummary` and `proteinSummary` can be called on any of the classes to retrieve the number of inferred proteins and a table summary.

The `filterHits` method can filter any of the classes based on the ion score and the rank of a search hit. Except when filtering a `SearchHit`, where `TRUE` or `FALSE` is returned, depending if the hit passes the filtering criteria, `filterHits` returns an object of the same class.

```
> msf <- filterHits(ms,rank=2,ionscore=20)
> class(msf)

[1] "MSMSpepXML"
attr(,"package")
[1] "RpepXML"

> proteins(ms[3])

[1] "ECA0851" "ECA3748" "BSA"     "ECA0851" "ECA3174" "ECA3748"

> proteins(msf[3])

[1] "BSA"
```

An `MSMSpepXML` object can also be converted to a matrix with `as.matrix`. Only first rank hits and the first search results are considered and partial information is returned. The `all` parameter defined whether all spectrum queries should be returned or only those with search results.

```
> as.matrix(msf,all=TRUE)

  QueryIndex SpectrumId                      ionscore identityscore
1 "1"        "679_c403.2119_u403.2178_r63.12" ""       ""
2 "2"        "511_c402.2474_u402.2695_r41.13" ""       ""
3 "3"        "186_c660.3871_u662.3625_r58.85" "34.67"  "27"
4 "4"        "40_c402.7615_u402.7805_r41.44"  ""       ""
  homologyscore expect   Sequence     NumProteins Proteins
1 ""            ""       ""           "0"         ""
2 ""            ""       ""           "0"         ""
3 "22"          "0.0089" "K.TPVSEK.V" "1"         "BSA"
4 ""            ""       ""           "0"         ""
```

5. More details

A full list of functions and methods can be obtained with `help(package="RpepXML")`. Methods, functions and classes are documented individually.

6. Session information

- R version 2.15.0 (2012-03-30), `x86_64-pc-linux-gnu`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Biobase˜2.14.0, RpepXML˜0.3.3, XML˜3.93-0, plyr˜1.7.1
- Loaded via a namespace (and not attached): tools˜2.15.0