Parsing mzIdentML files using mzID

Thomas Dybdal Pedersen

October 15, 2013

mzID is a parser for the mzIdentML file format defined by HUPO. The mzIdentML file format is designed to be a standardized way of reporting results from peptide identification analyses used in proteomics. The file format is XML compliant and the parser relies heavily on the `XML` package.

mzID is designed to be applicable to all instances of mzIdentML files. As there is a multitude of different ways to arrive at a protein identification result, the structure and content of different mzIdentML files can vary greatly. If an mzIdentML file that causes errors during parsing is encountered, please contact the package maintainer in order to get this sorted out.

The mzID function

The meat of the mzID package is the mzID function which takes in a single path to an mzIdentML and reads it into an object of class mzID:

```
mzResults <- mzID("http://psi-pi.googlecode.com/svn/trunk/examples/1_1examples/Mascot_NA_e
mzResults

## An mzID object
##
## Software used:   Mascot (version: 2.2.03)
##                  Mascot Parser (version 2.3.0.0)
##
## Rawfile:         file:///est_coding_test.mgf
##
## Database:        file:///C:/inetpub/mascot/sequence/EST_mini/current/EST_mini_20080623.
##
## Number of scans: 4
## Number of PSM's: 4
```

The structure of the mzID object is somewhat related to the structure of the mzIdentML XML specification, with slots holding different parts of the information:

```
showClass("mzID")

## Class "mzID" [package "mzID"]
##
## Slots:
```

```
##
## Name:      parameters            psm       peptides       evidence
## Class: mzIDparameters        mzIDpsm   mzIDpeptides   mzIDevidence
##
## Name:        database
## Class:   mzIDdatabase
```

The first slot contains a class that holds all information related to how the analysis was carried out - both the location of data files and the parameters used by the software. The psm slot contains a class that stores the scans and the related PSM's. This is usually the meat of the mzID class. The last three slots contains information on the peptides searched for, together with the possible modifications (the peptides slot), the proteins that constituted the database (the database slot) and a link between those two (the evidence slot).

The flatten function

As most people usually think of peptide identifications in a flat tabular format, the `mzID` package comes with a function to flatten an mzID object into a `data.frame` with a row for each PSM in the object. This function is aptly named `flatten`.

```
flatResults <- flatten(mzResults)
names(flatResults)
```

```
##  [1] "spectrumID"               "calculatedMassToCharge"
##  [3] "chargeState"              "experimentalMassToCharge"
##  [5] "rank"                     "passThreshold"
##  [7] "mascot:expectation value" "mascot:score"
##  [9] "pepSeq"                   "modified"
## [11] "modification"             "start"
## [13] "end"                      "pre"
## [15] "post"                     "frame"
## [17] "isDecoy"                  "length"
## [19] "accession"                "description"
## [21] "sequence"                 "databaseFile"
```

```
nrow(flatResults)
```

```
## [1] 4
```

```
# The length of an mzID object is the number of PSM's
length(mzResults)
```

```
## [1] 4
```

As can be seen from the column names of the flattened results, care should be taken when interpreting the different columns, as ambiguity can arise when combining all the information in the object into a flat structure. For instance the column named 'length', could refer to both the length of the peptide sequence

and the length of the protein from where it came (or the length of the nucleotide sequence coding for the protein). Inspection of the content reveal that the column indeed refers to the length of the nucleotide sequence coding for the protein.

```
flatResults$length
```

```
## [1] 163 495 380 495
```

```
nchar(flatResults$sequence)
```

```
## [1] 163 495 380 495
```

```
substr(flatResults$sequence, 1, 10)
```

```
## [1] "CATGTGTCTA" "GTGCACTGAC" "CTTGAAGTTN" "GTGCACTGAC"
```

This would also be apparent if one were deeply familiar with the mzIdentML specification and new that length only gets reported for the proteins, but this kind of knowledge is not expected from the regular user. Thus a bit of attention is required when inspecting the results.

Future work

- mzID is intended as a very barebone parser to be used by other packages and the feature list is thus kept short. Together with the mzR package it provides parsing of the two most common file types used in proteomics. The third filetype: mzQuantML (for storing quantitative information from LC-MS/MS analyses) should be supported by a separate package in the future.
- Automatic extraction of mzR-compatible acquisition number based on the SpectrumID and IDFormat.
- Software awareness so that the parser understood what type of output is expected from e.g. a Mascot search.
- Support for other file types could be neat, but is not a high priority at the moment.