Case study: Adding external functionality to Proteomatic

This document demonstrates the necessary steps to add a script to Proteomatic which calls an external, third-party script to convert OMSSA CSV result files to the mzIdentML file format.

The HUPO website provides a list of tools exporting mzIdentML files. For OMSSA, a csv2mzIdentML Perl script is listed. This Perl script is contained in a project called “web-based-multiplesearch”, hosted at Google Code: http://code.google.com/p/web-based-multiplesearch/ [1].

In order to add the functionality to Proteomatic, two steps are necessary:

1. Add the “web-based-multiplesearch” package to the CLI tools atlas, which is a stand-alone project and included into the Proteomatic scripts project as a Git submodule.
2. Add a script description and the actual script to the Proteomatic scripts repository.

In the following, the outlined steps are described.

Additions to the CLI tools atlas

The “web-based-multiplesearch” package must be added to the CLI tools atlas. A YAML file is created to describe the whole package:

cli-tools-atlas/packages/ext.web-based-multiplesearch.yaml

```
title: web-based-multiplesearch
version: "beta-2011-02-06"
link: 'http://code.google.com/p/web-based-multiplesearch/'
download: http://web-based-multiplesearch.googlecode.com/files/FDRapp.zip
path: ''
```

Next, a second YAML file is required to describe the tool inside the package. Because the tool of interest is a Perl script and consequently, there are no different versions for different platforms, no distinction between platforms must be made.

cli-tools-atlas/packages/ext.web-based-multiplesearch.csv2mzidentml.yaml

```
binary: 'FDRapp/src/Parsers/csv2mzIdentML.pl'
```

Proteomatic script implementation

For the script, the converter script type is chosen, thus every input CSV file will be converted into a mzIdentML file. The filename of the script should be “csv-to-mzidentml”, and it will be implemented in Ruby. First, a YAML-formatted description for the script is created:

proteomatic-scripts/include/properties/csv-to-mzidentml.yaml

```
group: Proteomics/Identification/OMSSA/Miscellaneous

title: Convert OMSSA CSV to mzIdentML

description: >
  Convert OMSSA CSV output files to the mzIdentML file format.
  <b>Attention:</b> Only peptide identifications are reported in the output
  file, details about the database search (mass tolerances, enzyme, ...) are
  not saved.

type: converter
```

```
input:
  - key: csvFiles
    label: OMSSA PSM
    formats: [csv]
    min: 1

defaultOutputDirectory: csvFiles

output:
  - key: csvFiles
    label: mzIdentML files
    format: xml-mzidentml
    filename: '#{basename}.mzid'

needs:
  - lang.perl
  - ext.web-based-multiplesearch
```

The description denotes that the “web-based-multiplesearch” package is required (`ext.web-based-multiplesearch`), and also Perl for executing the csv2mzIdentML.pl script (`lang.perl`). Also, the translation from input filenames to output filenames is defined: the `.csv` file suffix gets replaced by `.mzid`.

Finally, the script is implemented:

proteomatic-scripts/csv-to-mzidentml.rb

```
#! /usr/bin/env ruby

require './include/ruby/proteomatic'
require './include/ruby/externaltools'
require 'fileutils'

class CsvToMzIdentML < ProteomaticScript
    def run()
        @output.each do |inPath, outPath|
            puts "Converting #{inPath}..."
            paramFilePath = tempFilename('param-')
            tempFilePath = tempFilename('out-')
            # write parameter file for csv2mzidentml Perl script, using default values
            File::open(paramFilePath, 'w') do |f|
                f.puts "Param,cvTerm,Accession,Value"
                f.puts "Software name,OMSSA,MS:1001475,N/A,,"
                f.puts "Provider,N/A,N/A,N/A,,"
                f.puts "Parent mass type,parent mass type mono,MS:1001211,N/A,,"
                f.puts "Fragment mass type,fragment mass type mono,MS:1001256,N/A,,"
                f.puts "Enzyme,Trypsin,MS:1001251,N/A,,"
                f.puts "Missed cleavages,N/A,N/A,2,,"
                f.puts "Fragment search tolerance plus,search tolerance plus value,MS:1001412,0.5,,"
                f.puts "Fragment search tolerance minus,search tolerance minus value,MS:1001413,0.5,,"
                f.puts "Parent search tolerance plus,search tolerance plus value,MS:1001412,0.02,,"
                f.puts "Parent search tolerance minus,search tolerance minus value,MS:1001413,0.02,,"
                f.puts "PSM threshold,no threshold,MS:1001494,N/A,,"
                f.puts "Input file format,OMSSA csv file,MS:1001399,N/A,,"
                f.puts "Database file format,FASTA format,MS:1001348,N/A,,"
                f.puts "Decoy database regex,decoy DB accession regexp,MS:1001283,__td__decoy_,,"
                f.puts "Spectra data file format,Mascot MGF file,MS:1001062,N/A,,"
                f.puts "Spectrum ID format,multiple peak list nativeID format,MS:1000774,N/A,,"
            end

            # build the command in the following way and execute it:
            # [perl] [csv2mzidentml.pl] [input files] [parameters] [output file]
            command = "\"#{ExternalTools::binaryPath('lang.perl.perl')}\" " +
                "\"#{ExternalTools::binaryPath('ext.web-based-multiplesearch.csv2mzidentml')}\" " +
                "\"#{inPath}\" \"#{paramFilePath}\" \"#{tempFilePath}\""

            runCommand(command)
```

```
            # now remove some information from the input file which we cannot provide at this point
            File::open(outPath, 'w') do |fout|
                cut = []
                cutList = ['provider', 'auditcollection', 'additionalsearchparams', 'enzymes',
                           'masstable', 'fragmenttolerance', 'parenttolerance']
                File::open(tempFilePath, 'r') do |f|
                    f.each_line do |line|
                        test = line.downcase.strip
                        if test[0, 1] == '<'
                            tag = /\w+/.match(test)
                            if tag
                                tag = tag.to_a.first
                                (test[0, 2] == '</') ? nil : cut << tag if cutList.include?(tag)
                            end
                        end
                        fout.puts line if cut.empty?
                        if test[0, 1] == '<'
                            tag = /\w+/.match(test)
                            if tag
                                tag = tag.to_a.first
                                (test[0, 2] == '</') ? cut.pop : nil if cutList.include?(tag)
                            end
                        end
                    end
                end
            end

            # delete temporary files
            FileUtils::rm_f(paramFilePath)
            FileUtils::rm_f(tempFilePath)
        end
    end
end

script = CsvToMzIdentML.new
```

The script consists of three steps:

1. Prepare the config file for the csv2mzidentml Perl script.
2. Run the Perl script.
3. Remove false information from the output file. This must be done because the Perl script requires more information than can be deduced from the input file, e. g. the enzyme used for *in silico* digestion. To resolve this issue, default values are provided (e. g. trypsin) and subsequently removed from the output file.

Results

This implementation converts OMSSA CSV files to mzIdentML files, denoting for every peptide/spectral match (PSM) whether it is a target or decoy hit, if applicable. However, because the script can be run at any time in a MS/MS data evaluation pipeline, there is no way of knowing the parameters of the database search (mass tolerances, enzyme, ...) from the input CSV file. As a result, this information is not contained in the output file.

The addition of the discussed files to the Proteomatic scripts directory will enable the new script in Proteomatic without requiring recompilation. On Windows, Perl will be downloaded automatically if it is not yet available on the user's computer. Furthermore the web-based-multiplesearch package will be downloaded and unpacked automatically upon loading the script.

References

[1] Wedge DC, Krishna R, Blackhurst P, Siepen JA, Jones AR, Hubbard S. *FDRAnalysis: A tool for the integrated analysis of tandem mass spectrometry identification results from multiple search engines.* J Proteome Res. 2011 Jan 11. [Epub ahead of print] PubMed PMID: 21222473.