

InSilicoSpectro: An Open-Source Proteomics Library

Jacques Colinge,^{*,†,#} Alexandre Masselot,^{*,†,#} Pablo Carbonell,^{§,∇} and Ron D. Appel[‡]

Upper Austria University of Applied Sciences at Hagenberg, Bioinformatics Department, Hauptstraße 117, A-4232 Hagenberg, Austria, Geneva Bioinformatics S.A., Av. de Champel 25, CH-1206 Genève, Switzerland, Higher Polytechnic School of Alcoy, Polytechnic University of Valencia, E-03801 Alcoy, Alicante, Spain, and University of Geneva and Swiss Institute of Bioinformatics, Rue Michel-Servet 1, CH-1211 Genève, Switzerland

We present a new proteomics open-source project, InSilicoSpectro, aimed at implementing recurrent computations that are necessary for proteomics data analysis. Illustrative examples are mass list file format conversions, protein sequence digestion, theoretical peptide and fragment mass computations, graphical display, matching with experimental data, isoelectric point estimation, and peptide retention time prediction. The project library is written in Perl, a widely used scripting language in bioinformatics, and it offers a unique framework of integrated objects to implement complex proteomics data analyses. For instance, only a few lines of code are required to digest a protein with fixed and variable modifications, label peptides with ¹⁸O, compute the fragmentation spectra and display their match with experimental spectra. We believe that InSilicoSpectro will be of great help to bioinformaticians, without detailed knowledge of proteomics specifics, and to mass spectrometrists with computer programming interest as well.

Keywords: open-source • Perl • computations • algorithm • theoretical spectrum

1. Introduction

The processing of proteomics mass spectrometry (MS) data requires special computations that are out of the scope of classical software developed in bioinformatics or regular software industry. Typical MS data processing tasks comprise raw spectra analysis, file format conversions, mass list-based computations, and storage of raw data and analysis results.

We believe that there is a need for a program library dealing with MS-dedicated computations that would simplify the development of computer programs for proteomics. This would allow computer specialists, or bioinformatics specialists, without in-depth knowledge of proteomics specifics, to support their life science colleagues more efficiently. In particular, the availability of a reliable and comprehensive MS library should allow them to focus on the functionalities of new software and to ignore the details of certain (tedious) algorithms.

Moreover, a simple to use MS library would also make mass spectrometry specialists, with minimal training in computer programming, able to implement useful tools by their own. This is certainly useful in laboratories without sufficient access to bioinformatics specialists.

InSilicoSpectro is a new proteomics open-source project intended to cover common operations in file format conversions, protein sequence digestion, theoretical mass spectra computations and comparisons, text/graphic display, stable isotope labeling^{1,2} (SIL) and tagging,³ peptide retention time predictions in reverse phase high-pressure liquid chromatography (RP-HPLC) columns, low-quality spectra filtering, and a few other related topics. The problems of raw data processing, storage, and database searching are not addressed by the InSilicoSpectro project. See Figure 1 for a schematic view of InSilicoSpectro scope. InSilicoSpectro is released under the LGPL license and it is available from a dedicated web site at <http://insilicospectro.vital-it.ch>. The LGPL license allows for free reuse of the code, including in commercial programs such as Phenyx (<http://www.phenyx-ms.com/>) that uses the library for its Perl computations.

The scope of InSilicoSpectro matches a domain where the functionalities of available programs can be regarded as insufficient. Raw spectra processing and database searching/peptide de novo sequencing are classical problems addressed by several commercial and free programs. Nonetheless, techniques such as SIL and tagging, or the development of site-specific tools for displaying or further analyzing data, requires computations that are not, or partially only, supported by available programs. Given the huge diversity of needs and sample processing technologies, it is very difficult—if not impossible—to develop fully generic programs that would handle all possible tasks. Consequently, we decided to develop a library of software modules and to let users combine them to suit their specific needs. By hiding the complicated com-

* To whom correspondence should be addressed. J.C. Tel: +43(0)-723638882720. Fax: +43(0)723638882499. E-mail: jacques.colinge@fh-hagenberg.at. A.M. Tel: +41(0)227029900. E-mail: alexandre.masselot@genebio.com.

[†] Upper Austria University of Applied Sciences at Hagenberg.

[‡] Geneva Bioinformatics S.A.

[§] Polytechnic University of Valencia.

[∇] University of Geneva and Swiss Institute of Bioinformatics.

[#] Authors contributed equally to this work.

[∇] This work was done while P.C. was visiting Geneva Bioinformatics and the University of Geneva

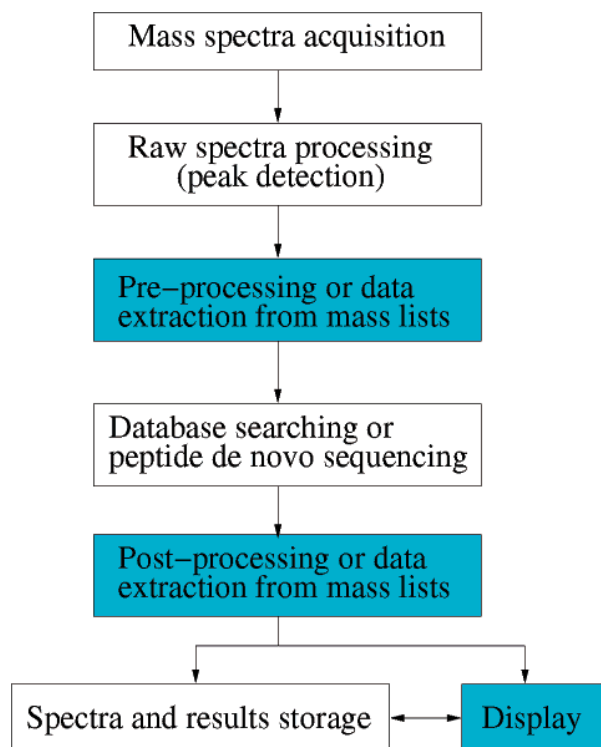


Figure 1. Schematic overview of mass spectrometry data generation and processing. The steps covered by InSilicoSpectro are colored.

putations in the library code, we are convinced—and we actually experienced—that relatively complex data processing tasks can be implemented rapidly, safely, and easily.

To reach a large audience of potential users and to limit the required computer programming skills, we decided to use a scripting language, namely Perl, as the main programming language and interface to the library. Perl is widely used in bioinformatics and this should facilitate the integration with existing bioinformatics infrastructures. Moreover, Perl is an extraordinary tool for rapid software development and our professional activities learned us that life science research often imposes short dead-lines. Perl is a stable language on many platforms, notably Linux, Microsoft Windows, and Mac OS X. In addition, the development of the future Perl 6 compiler, i.e., Parrot (<http://www.parrotcode.org/>), should allow the Python community to use InSilicoSpectro Perl modules quite transparently, which would make us able to support almost all bioinformatics scripting language developers.

The current version of InSilicoSpectro includes modules for MS data management, essentially mass list file format conversions, protein sequence digestion, fixed and variable protein modifications representation, theoretical mass computations for peptide mass fingerprinting (PMF) and tandem mass spectrometry (MS/MS), graphic display, peptide retention time predictions, and protein isoelectric point (pI) estimations. Partial support for SIL is already implemented although it is still an ongoing part of the project as well as low-quality spectra filtering and further graphical outputs. InSilicoSpectro comprises several examples and simple tools to illustrate library use including a file format converter, a simple web site to digest protein sequences and compute peptide theoretical fragmentation spectra, a program to extract statistics about fragment peak intensities, and many short examples that directly illustrate

specific functionalities of the libraries. We plan to maintain and develop InSilicoSpectro over the coming years and, hopefully, users feedback and comments will also influence its evolution.

2. Materials and Methods

InSilicoSpectro Perl library modules are organized according to their functions. At the more general level there is a module named InSilicoSpectro.pm that provides general functionalities for initializing all the other modules (lecture of configuration files). More specialized modules are grouped in three folders: “Spectra” for mass list-related modules, “InSilico” for computational modules, and “Utils” for a few utility modules. We plan to implement in the future some computation intensive routines in C++ but they will always be wrapped by a Perl interface.

The general design of the modules follows the object oriented programming (OOP) model and most of the modules are class definitions actually. The module that implements most of the theoretical mass computation routines supports a dual OOP and procedural programming model. Functions in this module are more susceptible to be strongly integrated into proprietary software using InSilicoSpectro and we decided not to force users to adhere to a specific model. Other modules functions should be used in a standard and simple way thus limiting the complexity of the interface with proprietary developments. Hence, we believe that procedural programming developers should not be disturbed too much if they have to use them via (elementary) objects.

InSilicoSpectro modules make use of some Perl modules that are not part of the standard Perl distribution, such as Statistics:Regression, XML:Twig, GD, and IA:NNFlex. These modules are available from CPAN, the Comprehensive Perl Archive Network, and constitute de facto an extension of the language that increases the stability and reliability of the overall InSilicoSpectro project Perl code. InSilicoSpectro requires GD and Zlib, two libraries for graphics plot and data compression that must be installed from rpms or similar precompiled packages (or source code alternatively).

We have developed a simple and minimal class hierarchy to represent protein sequences and peptides (as digestion product) in a way that, on one hand, fits the needs of the computations we perform and, on the other hand, stays relatively neutral in its design. Thus, it should be possible to combine the latter classes with existing projects at users sites, e.g., via multiple inheritance, or to use them as the basis of more sophisticated objects. The protein sequence class has some compatibility with BioPerl⁴ objects in the sense that it has a constructor that accepts an object of class Bio::Seq as parameter. This eventually gives access to some nice BioPerl functionalities such as sequence query over the Internet, without forcing InSilicoSpectro users to use BioPerl. All other InSilicoSpectro objects are rather specific to the field or proteomics computations and they are not especially designed to be compatible with other class hierarchies, though their design follows the usual good practice rules.

InSilicoSpectro Perl code is documented mainly via pod and a collection of simple and focused examples. A few tools illustrate the usage of the modules to realize simple tasks. The web site will feature a news area to report evolution of the library and new versions, and a bug tracking system is in place for users to inform us about abnormalities.

3. Results

In this section, we show how some illustrative and relatively complex computations can be made easy by using InSilicoSpectro modules. More examples are provided with the library itself.

File Format Conversion. Existing formats, generally defined by instrument manufacturers, export at least basic data: mass over charge and intensity. Historically, some of these file formats have been extended to include supplementary data such as charge state, signal-to-noise ratio, peak width, etc. In certain cases, supplementary data appear in comment lines since the original format was not made to be extended. Community-wide efforts try to define standards covering most needs: mzData, from the HUPO-PSI consortium (<http://psidev.sourceforge.net/ms/#mzdata>), or mzXML (http://sashi-mi.sourceforge.net/software_glossolalia.html).

Until these undergoing normalization projects converge to a unique format that would be supported by all instruments proprietary software, there is a need to manage the existing peak list formats and their most common extensions. InSilicoSpectro offers a framework to deal with extracted peak lists in a unified manner by means of a generic Perl class—and its specialized descendents—able to represent mass lists with a description of the data coming with each peak. Perl objects of this class can be given handlers responsible for reading and writing mass lists in many formats. If necessary, InSilicoSpectro library users can write their own handlers.

The unifying Perl object for mass lists not only stores data and converts from/into various file formats, it tries to “reduce” redundant MS/MS spectra as well. For example, in the dta file format, a MS/MS spectrum acquired from a parent ion, whose charge could not be uniquely determined, is typically stored several times in several files as dta does not allow multi-charges. Such a classical case is an uncertainty between double and triple charges. When reading the data, the handler checks for redundancy (compatible parent masses, same fragment masses) and merges the spectra in a single multi-charged spectra object in computer memory. When such an object is written into a file, an intelligent conversion is performed: if the file format does not allow multi-charges, as dta, then several spectra are written, one per charge state, otherwise the flexibility of the file format is exploited as when writing in the mgf format. A mass list format converter comes with InSilicoSpectro modules both as a useful tool and an example. File format conversion tools usually convert one format to another but they do not let the user specify input and output formats freely.

Theoretical Mass Computations and Digestions. The “simple” operation of digesting a protein sequence becomes complicated when supporting special enzymes, protein modifications, and SIL. InSilicoSpectro comes with a configuration file that contains enzyme definitions. This XML file can be extended at will by the users. Enzyme definitions can be created with a lot of flexibility as illustrated by a special trypsin definition intended to analyze data obtained from a ^{18}O labeling^{5,6} pipeline:

We see that we can define the enzyme specificity by means of a cleavage site and a constraint on the next amino acid (cleaves at K or R when not followed by P). The enzyme can be defined as cleaving at the N- or C-terminus of the cleavage site. The enzyme specificity can be alternatively specified as a Perl regular expression. The CTermGain and NTermGain tags can be used to define atom gains/losses (negative numbers are

```
<oneCleaveEnzyme name="Trypsin_018modif">
  <site>
    <cleavSite>KR</cleavSite>
    <adjacentSite>^P</adjacentSite>
    <terminus>C</terminus>
  </site>
  <CTermGain>OH</CTermGain>
  <NTermGain>H</NTermGain>
  <CTermModif>018</CTermModif>
</oneCleaveEnzyme>
```

possible) at peptide termini. The example above is the standard case but special enzymes might require such a flexibility in order that we can compute peptide masses, atomic composition and fragmentation spectra correctly. Finally, CTermModif (and NTermModif) allow us to define enzymes that introduce modifications in the peptides. In the example above, we define a ^{18}O modification (not shown), with mass shift +4 Da, and we use the CTermModif tag to declare that this special version of trypsin causes this modification at the C-Terminus. Alternatively, we could introduce no modification and use the CTermGain tag to introduce supplementary isotopes.

Digestions can be performed with an enzyme, with possible missed cleavages, or without enzyme (all possible peptides), or with an enzyme but by generating half-enzymatic peptides. All the peptides generated come with a precise description of the modifications they carry as well as the atoms at their termini (see explanation above). In particular, it is possible to define modifications on a protein and to digest it by having the modifications correctly transferred to the peptides. Therefore, the digestion procedures are fully compatible with subsequent PMF and MS/MS mass computations. In case variable modifications are defined, all possible combinations (MS/MS) or numbers (PMF) are generated. For an example, see Figure 2.

Fragment mass computations can be configured via an XML file and comprise all sorts of fragments,⁷ charge states and losses, including multiple and combined ones. For instance, we can define γ fragments with variable number of H_3PO_4 losses, which are of interest for phosphopeptides,⁸ as follows:

```
<oneSeries name="b">
  <terminus>N</terminus>
  <firstFragment>2</firstFragment>
  <lastFragment>2</lastFragment>
</oneSeries>
<oneLoss name="H3P04">
  <residues aa="STY"/>
  <formula>H-3P-10-4</formula>
</oneLoss>
<oneFragType name="y-H3P04*" series="y" charge="1">
  <loss name="H3P04" repeat="-1"/>
</oneFragType>
```

So far, the only supported internal fragments are immonium ions but we designed the library such that it will be a limited effort to include more general internal fragments. Fragmentation spectra can be exported in HTML, text, or $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ format to assist in document preparation or web site development.

All enzymes, post-translational modifications, and fragment types can be configured through XML files or dynamically added at run-time by calling appropriate functions or methods.

It is possible to match experimental masses with theoretical spectra with variable precision and to manipulate the resulting matches in Perl. Moreover, the matches can be exported in text

(A)

```
MCTMACTKGIPRKQWEMMKPCKADFCV
::Cys_CAM::Oxidation:::.....(*)Oxidation:::.....

Tryptic digestion (nmc=1) with O18 (modif O18_twice):
Peptide          Start Stop  Mass      Modifications
MCTMACTK.G       0   7   964.383  ::Cys_CAM::Oxidation:::018
K.QWEMMKPCK.A   13  22  1369.632  :::.....018
K.QWEMMKPCK.A   13  22  1385.627  :::Oxidation:::018
K.ADFCV          23  27  553.221  :::.....
MCTMACTKGIPR.K   0  11  1387.642  ::Cys_CAM::Oxidation:::.....018
K.GIPR.K         8  12  573.394  :::.....018
R.KQWEMMKPCK.A  12  22  1497.727  :::.....018
R.KQWEMMKPCK.A  12  22  1513.722  :::Oxidation:::018
K.QWEMMKPCKADFCV 13  27  1900.813  :::.....
K.QWEMMKPCKADFCV 13  27  1916.808  :::Oxidation:::.....
```

(B)

```
my $protein = new InSilicoSpectro::InSilico::AASequence(
    sequence=>'MCTMACTKGIPRKQWEMMKPCKADFCV',
    modif=>':Cys_CAM::Oxidation:::.....(*)Oxidation:::.....',
    AC=>'12345');
print "Protein:\n$protein", modifToString($protein->modif(),
    $protein->getLength()),
    "\n\nTryptic digestion (nmc=1) with O18 (modif O18_twice):\n";
my @result = digestByRegExp($protein->$protein, nmc=>1,
    enzyme=>InSilicoSpectro::CleavEnzyme::getFromDico('Trypsin_O18modif'));
foreach (@result){
    print "$_\t", join("\t", $_->start(), $_->end(), $_->getMass(),
        modifToString($_->modif())), "\n";
}
}
```

Figure 2. (A) An example of the digestion of a short protein sequence, with modifications, by trypsin such that tryptic peptides are modified with two ^{18}O at their C-Terminus (see text); a default minimum mass of 500 Da is imposed. Modifications are indicated in a special string with one position per amino acid plus two extra position for terminal modifications; (*) indicates a variable modification. Computations are made in view of MS/MS and hence the modifications are localized, a simple count for PMF is another option of the library. Note that C-Terminal peptides have no ^{18}O . (B) The short Perl code necessary to compute this example.

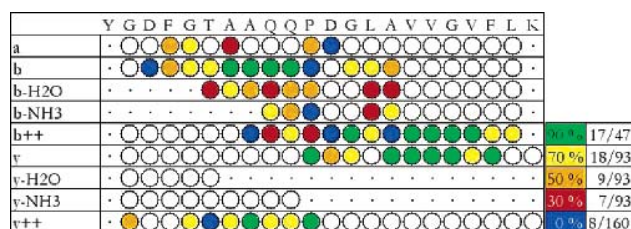


Figure 3. Example of the tabular graphic representation of the match between theoretical and experimental MS/MS masses. The color code indicates experimental peak intensities after normalization. The graphical output code allows various intensities normalizations (no, log, relative, rank) and lets the user employ a default color scale or define her/his own (variable colors and number of bins). In the graphic output, it is also possible to display the color scale itself as well as a count of the number of peaks matched in each intensity bin versus the total number of experimental peaks in the same bin. Note that it happens that certain peaks match more than one theoretical masses but the counts are always correct, i.e., it may have more color disks than the total count. The choice of the character font automatically adjusts the image size. There are fonts for screen display as well as for printed documents (TrueType(TM) fonts).

or graphical format; the graphical format comes with a flexible scheme to normalize peak intensities and define color scales, see Figure 3.

Peptide RP-HPLC Retention Times and Protein pIs. Several authors already published methods aimed at predicting peptide retention times in a RP-HPLC column. The main motivation

of doing such predictions is to help in the validation of peptide identification since, provided the predictions are precise enough, strong discrepancies between experimental and predicted times may indicate spurious identifications. The prediction of protein *pI* serves the same goal in the context of electrophoretic gels. Obviously, the ability to predict retention times and *pIs* has additional applications.

InSilicoSpectro contains two modules implementing published methods of RP-HPLC predictions: Hodges and co-workers⁹⁻¹¹ based their predictions on the sum of hydrophobicity contributions from every residue (coefficient sets empirically determined by several other authors available from the module); Petritis et al.¹² used a neural network to infer the retention time on the basis on the amino acid composition. Refer to the original publications for more details and an estimation of each method performance and see Figure 4 for an example of how the InSilicoSpectro code is called.

For the three proposed methods, the user can recompute and save her/his own parameters, by using the ad hoc algorithm, provided sufficient training data are available. In case not enough data are available, it is possible to re-calibrate predicted times only by using a small set of validated retention times (every method predicts normalized retention times as actual times depend on laboratory specific settings).

Concerning *pI* estimations, they are computed by an iterative algorithm or by a theoretical approximation using the regressed dissociation constants.¹³ Several other authors later published different sets of parameters to weight acidic and basic residues

(A)

```
./computeRT.pl --method=Hodges --current=Guo86 data/test1
```

(B)

```
./computeRT.pl --method=Hodges --learnfrom=data/expdata
--saveparam=data/paramRT.xml
```

(C)

```
# Learn from experimental data
if ($learnfrom and !$readparam) {
  croak "Bad file $learnfrom" unless ReadFromFile($learnfrom,\@expseqs,\@exptimes);
  if ($method eq 'Hodges') {
    $rt->learn(data=>{\expseqs=>\@expseqs,exptimes=>\@exptimes},
              current=>'Test',overwrite=>0,comments=>'Test Hodges',
              %settings);
  } elsif ($method eq 'Petritis') {
    $rt->learn(data=>{\expseqs=>\@expseqs,exptimes=>\@exptimes},
              maxepoch=>30,sqrrerror=>1e-3,mode=>'quiet',
              nnet=>{\learningrate=>0.05},layers=>[\{nodes=>20\},\{nodes=>6\},\{nodes=>1\}],
              %settings);
  }
}

$rt->filter(filter=>10,error=>'relative',%settings) if exists $settings{filter};

# Calibrate data
if ($calibratefrom) {
  croak "Bad file $calibratefrom" unless ReadFromFile($calibratefrom,\@calseqs,\@caltimes);
  $sec=InSilicoSpectro::InSilico::ExpCalibrator->new(fitting=>'linear',%settings);
  $rt->calibrate(data=>{\calseqs=>\@calseqs,caltimes=>\@caltimes},calibrator=>$sec);
}

# Save coefficients
SaveParamFile($saveparam,$rt,$method,$current)if $saveparam;
```

Figure 4. (A) An example of RP-HPLC retention time computation for peptides stored in a file with the parameters published by Guo et al.⁹ Here, we call a small Perl script that makes use of the generic module available from InSilicoSpectro. (B) An example where we compute the optimal parameters for a given data set and save them into a file. C. The code in the Perl script to learn, calibrate and save.

for pI estimations, they are available from the module. See Figure 4 for an example. As for the retention time prediction methods, it is possible to re-calibrate predicted pIs to better fit one's own data.

Comparison with Previously Existing Systems. As we mentioned in the Introduction—and to our best knowledge—there exists no other software package with a similar scope as InSilicoSpectro. Tools accessible over the Internet propose functionalities that are part of InSilicoSpectro: PeptideMass, PeptideCutter, and ProtParam available from expasy;¹⁴ ProteinProspector.¹⁵ Nonetheless, their scope is more limited and, most important, it is not possible to modify these tools to integrate them with other tools to meet specific needs.

Open source database search engines OMSSA¹⁶ and XTandem¹⁷ certainly contain code to represent and digest protein sequences, and to compute theoretical spectra as well, but to use this code requires to “extract” it from its original context and to use it as a library. Although this is obviously feasible and not necessarily difficult, given the code is properly designed, it requires more advanced technical skills. Moreover, this would not cover everything we propose. The same is true concerning GNU polxmass,¹⁸ an open source program for mass spectrometry computations (mainly masses).

ProteomeCommons.org proposes a Java program to perform mass list file format conversions between several formats and some manipulations such as merging mass lists. The Institute

for Systems Biology open-source tools comprise a program (mzXML2Other) to convert mzXML into other formats (<http://www.systemsbio.org/>).

A commercial program that runs under Microsoft Windows only, GPMW (<http://welcome.to/gpmaw>), offers a rich set of functionalities for manipulating proteins sequences and interfacing with sequence databases and common bioinformatics tools such as Blast or ClustalW. GPMW functionalities related to mass calculations and sequence digestion overlap with most of InSilicoSpectro functionalities in this domain, though we cover more extensively what concerns the match with experimental data. GPMW is not a library and it is not open-source. Hence, GPMW does not allow its users to integrate it with their own programs or to modify it.

We also mention OpenMS (<http://open-ms.sourceforge.net/>), an open-source project in C++ that has a rather complementary focus compared to InSilicoSpectro. OpenMS covers many issues related to peak detection in raw spectra, data storage in a relational database, raw data visualization and compression, as well as MS profile comparisons.

4. Conclusions

We have developed a novel open-source Perl library covering many common needs in proteomics computations. The current version includes file format conversions, protein digestion, post-translational modifications management, theoretical mass

computations, graphical display, and peptide retention time prediction. The library will be extended over the coming years to include even more functionalities. Other existing projects either have a more limited scope and/or do not come as a library of reusable functions and objects.

InSilicoSpectro design is intended to make its use as simple as possible including by non proteomics specialists or by non professional programmers. We believe it is a key feature of the project to provide help to a community as large as possible since people in proteomics data analysis have very diverse backgrounds. The choice of Perl as a programming language is motivated by its large acceptance in bioinformatics and its potential to support rapid tool development, which is exactly one of InSilicoSpectro ambitions.

In addition, InSilicoSpectro can be used for teaching purposes by hiding details of computations and thus giving students the opportunity to rapidly produce interesting tools and gaining direct experience in proteomics data handling. Moreover, the code itself could be used to illustrate the mechanics of certain computations at a later stage.

The InSilicoSpectro web site is located at <http://insilicospectro.vital-it.ch>. Additional information and links to download the library can be found there.

Acknowledgment. We thank Markus Müller and Manfred Heller, who tested the library and gave us early feedback, and the VITAL-IT team who kindly hosts our web site on their computers in Lausanne. We also thank the reviewers for their remarks and Koen van der Drift for testing the library with Mac OS X.

References

- (1) Julka, S.; Regnier, F. *J. Proteome Res.* **2004**, *3*, 350–363.
- (2) Gerber, S. A.; Rush, J.; Stemman, O.; Kirschner, M. W.; Gygi, S. P. *Proc. Natl. Acad. Sci. U.S.A.* **2003**, *100*, 6940–6945.
- (3) Gygi, S. P.; Rist, B.; Gerber, S. A.; Turecek, F.; Gelb, M. H.; Aebersold, R. *Nat. Biotechnol.* **1999**, *17*, 994–999.
- (4) Stajich, J. E. et al. *Genome Res.* **2002**, *12*, 1611–1618.
- (5) Yao, X.; Afonso, C.; Fenselau, C. *J. Proteome Res.* **2003**, *2*, 147–152.
- (6) Heller, M.; Mattou, H.; Menzel, C.; Yao, X. *J. Am. Soc. Mass Spectrom.* **2003**, *14*, 704–718.
- (7) Roepstorff, P.; Fohlman, J. *Biomed. Mass Spectrom.* **1984**, *11*, 601.
- (8) DeGnore, J. P.; Qin, J. *J. Am. Soc. Mass Spectrom.* **1998**, *9*, 1175–1188.
- (9) Guo, D. C.; Taneja, A. K.; Mant, C. T.; Hodges, R. S. *J. Chromatogr.* **1986**, *359*, 499–518.
- (10) Hearn, M. T.; Aguilar, M. L.; Mant, C. T.; Hodges, R. S. *J. Chromatogr.* **1988**, *438*, 197–210.
- (11) Mant, C. T.; Zhou, N. E.; Hodges, R. S. *J. Chromatogr.* **1989**, *476*, 363–375.
- (12) Petritis, K.; Kangas, L. J.; Ferguson, P. L.; Anderson, G. A.; Pasa-Tolic, L.; Lipton, M. S.; Auberry, K. J.; Strittmatter, E. F.; Shen, Y.; Zhao, R.; Smith, R. D. *Anal. Chem.* **2003**, *75*, 1039–1048.
- (13) Patrickios, C. S.; Yamasaki, E. N. *Anal. Biochem.* **1995**, *231*, 82–91.
- (14) Gasteiger, E.; Gattiker, A.; Hoogland, C.; Ivanyi, I.; Appel, R. D.; Bairoch, A. *Nucleic Acids Res.* **2003**, *31*, 3784–3788.
- (15) Clauser, K. R.; Baker, P.; Burlingame, A. L. *Anal. Chem.* **1999**, *71*, 2871–2882.
- (16) Geer, L.; Markey, S.; Kowalak, J.; Wagner, L.; Xu, M.; Maynard, D.; Yang, X.; Shi, W.; Bryant, S. *J. Proteome Res.* **2004**, *3*, 958–964.
- (17) Craig, R.; Cortens, J.; Beavis, R. *J. Proteome Res.* **2004**, *3*, 1234–1242.
- (18) Rusconi, F.; Belghazi, M. *Bioinformatics* **2002**, *18*, 644–645.

PR0504236