ProteoModlR

Paolo Cifani, Mojdeh Shakiba and Alex Kentsis

May 19, 2016

Software documentation

Contents

1 Overview

ProteoModlR is an open-sourced R suite that facilitates extracting relative protein concentration and differential stoichiometry of post-translational chemical modifications from quantitative proteomics experiments.

1.1 Applicability

ProteoModlR processes quantitative proteomics data enabling researchers with minimal experience with quantitative mass spectrometry to assess differential activation of functional cellular processes. ProteoModlR offers several modes of normalization to correct for different sources of error, and enables calculating differential protein expression and stoichiometry of post-tranlsational modification even from sparsely annotated datasets. The software can be easily integrated with existing proteomics software, such as MaxQuant and Skyline, as well as with statistical and pathway analysis tools.

1.2 Functionalities

ProteoModlR performs three user-customizable functions: *quality control*, *normalization* and *analysis*. In *quality control* the data is checked for the presence of all specified columns in the correct order. Data is then filtered to remove peptides without any valid (i.e. < 0) intensity measurement and classified based on the possibility to perform exact or approximate stoichiometry and abundance calculations according to the modification specified by the user (e.g. "phosphorylation" or "acetylation"). The filtered and classified data is exported as a CSV file into the working directory (file name), along with a bar graph summarizing the number of peptides and proteins remaining after each step of quality control.

The second step is *normalization*. Here, the user can choose from four normalization options: normalization to isotopologue, normalization to equimolar isotopologue, normalization to internal reference peptide(s), and/or normalization to total ion current. If none are selected the data is returned unchanged. Otherwise, the normalized values replace the input intensities in the data. The normalized data is then exported as a CSV file into the working directory (file name).

The third step is *analysis*, where peptides site occupancy and/or abundance is calculated. Depending on the user input on whether to restrict the analysis to exact calculations or to allow for approximate output (see section 3.2), the software selects the appropriate peptides based on the classification performed in quality control. The analyzed data is exported as a CSV file into the working directory (file name).

1.3 Use alongside other computational tools

ProteoModlR accepts as input any CSV formatted file with the appropriate columns. This type of file can be produced by a variety of softwares, such as Skyline and MaxQuant. The output of ProteoModlR is also in a CSV format and is thus compatible with pathway analysis tools, such as Cytoscape, and with statistical analysis software for downstream analyses.

1.4 Availability

ProteoModlR and its documentation are available for download at
http://github.com/kentsisresearchgroup/ProteoModlR.

1.5 Software Requirements

ProteoModlR is an R suite and requires the following R packages, all of which are available through CRAN (https://cran.r-project.org): 'plyr','ggplot2','reshape2'.

2 Input File

2.1 Input File Format

The input file must contain 12 columns in the following order: `Protein`, `Peptide`, `Pathway`, `Modification`, `Gene.Name`, `Position`, `Protein.Name`, `Condition`, `PatientID`, `Label`, `Run`, `Intensity`, even if the content of the columns are left empty. Skyline users are encouraged to customize the output to fit this format. MaxQuant output (modification specific peptides table) can be easily re-formatted using external CSV editors.

1. `Protein`: This column stores the protein identifier, preferably as UniProt ID [www.uniprot.org]. If multiple identifiers are listed, the program takes the first 6 characters corresponding to the base of the first UniProt ID.
2. `Peptide`: this column stores the peptide sequence in single letter code.
3. `Pathway`: This column may store the pathway identifier to which each protein corresponds. This classification is entirely for post-analysis purposes and will simply follow the remainder of the data to the output. It can be used afterwards for pathway analysis in other software.
4. `Modification`: This column stores the modification(s) for to a given peptide. Use of UniMod nomenclature [www.unimod.org] is highly recommended. Note that the software tolerates minor variations in the modification identifiers. The use of dash in the modification ID (e.g. di-methyl instead of dimethyl) should be avoided. Peptides with no detected modification should be labeled "unmodified".
5. `Position`: This column may store the position within the peptide to which the modification in column 4 belongs.
6. `Gene.Name`: This column may store the gene name or gene ontology (GO) identifier for each protein.
7. `Protein.Name`: This column may store the protein name.
8. `Condition`: This column stores the experimental condition to which each peptide/protein corresponds. For instance, in a case-control study, this will be "healthy" or "diseased", while for a timepoint study this will be "T1", "T2, etc.
9. `PatientID`: This column may store the unique patient/sample identifier.
10. `Label`: This column stores the isotopologue labeling for each peptide (e.g. "H" for heavy-labeled and "L" for light-labeled isotopologues). For label-free datasets, label all entries as "L".
11. `Run`: This column stores the replicate ID for a given sample. For example if an experiment was performed in 3 technical replicates, each measurement may be labeled 1 to 3 in this column.
12. `Intensity`: This column stores the original, untransformed measure of abundance (i.e. MS intensity) as calculated using other software such as Skyline.

2.2 Simulated test datasets

ProteoModlR was tested using two simulated datasets for the normalization and analysis modules respectively. Both datasets were modeled on experimental intensity values obtained experimentally by targeted detection on a Thermo Orbitrap Fusion mass spectrometer of peptides from protein MEF2C, plus TUB1 and H1 as reference proteins. The datasets are available for download http://github.com/kentsisresearchgroup/ProteoModlR.
Both dataset contains intensity values across three samples.
The *normalization* dataset simulates deviations from equimolarity introduced by different sources of technical and biological variability. The *analysis* dataset simulates missing measurements of different classes of peptides, for comparison with a fully annotated dataset.

3 Workflow

3.1 Quality control and Normalization

Once the data is made available in the working environment, the following code can be used to call both the *normalization* and the *quality control* functions with the desired inputs:

```
  Normalize(data, iso.norm = "", equim.iso.norm = "", internal.norm = "", tot.current=
"", mod="")
```

Quality control check for the presence and order of all required columns in the input files. If any chemoform of a given peptide bears the modification specified in `mod=""` (f.ex. `mod="phosphorylation"`) all peptides with the same sequence and containing the modification will be classified as *Modified*, while all peptides with the same sequence and not containing the modification will be classified as *Not-Modified.* If none of the chemoforms of a peptide contains the specified modification, the peptide is classified *Abundance*. After classification *Quality control* sums for each peptide the intensity of all *modified* and *not-modified* or *abundance* chemoforms.

`Normalize` takes in the data as a data frame. While the data file provided does not need to be exported from any specific program, the data should contain the 12 columns covered in Section 2. In addition to the data, the user can specify 5 arguments that determine the type of normalization to be performed (if any):

1. `iso.norm` takes a character string corresponding to the reference isotopologue to which the other isotopologue will be normalized (e.g. `iso.norm = "H"`). The string must match the corresponding annotation in the "Label" column. This normalization equalizes, for each chemoform independently, the intensities of reference isotopologues (f.ex. "H"), preserving the L/H ratio for each peptide. Normalized intensities (f.ex. light "L" peptides) are calculated as

$$Normalized\ Intensity\ nI_L^i = \frac{M_I^{iH} \ \times \ I_L}{I_H^i}$$

   where I^i indicates the intensity of a peptide i either in is light L or heavy H isotopologue, and M_I^{iH} indicates their median intensity for the heavy H peptides for chemoform i.

2. `equim.iso.norm` takes a character string corresponding to the reference isotopologue to which the other isotopologue will be normalized (e.g. `iso.norm = "H"`). The string must match the corresponding annotation in the "Label" column. This normalization first equalizes the intensities of all reference isotopologues (f.ex. "H") replacing the corresponding values with their median intensity calculated over the entire data frame. Normalized intensities (f.ex. light "L" peptides) are calculated as

$$Normalized\ Intensity\ nI_L^i = \frac{M_I^{DF} \ \times \ I_L}{I_H^i}$$

   where I^i indicates the intensity of a peptide i either in is light L or heavy H isotopologue, and M_I^{DF} indicates the median intensity calculated over the entire dataset.

3. `internal.norm` takes in a vector of character strings corresponding to the peptide sequence of the reference peptides to which other peptides are normalized (e.g. `internal.norm = "ATDVIVP"`). If more than one reference peptide is indicated (e.g. `internal.norm = c("ATDVIVP","AAATDVI")`), a geometric mean of the corresponding intensity values is taken. This normalization equalizes the intensities for the specified peptides in each sample. Normalized intensities for sample i are calculated as

$$Normalized\ Intensity\ nI^i = \frac{\overline{I_R^{eq}} \ \times \ I^i}{\overline{I_R^i}}$$

where $\overline{I_R^{eq}}$ indicates the (geometric mean) intensity of equalized reference peptide(s) R, $\overline{I_R^i}$ indicates the (geometric mean) intensity of reference peptide(s) R in sample i, and I^i indicates the non-normalized intensity in sample i.

4. `tot.current` takes in a boolean input indicating whether or not normalization to total ion current is to be made (e.g. `tot.current = T`). This normalization equalizes the sum of all intensities in each sample, used as a proxy for total ion current. Total Ion Current and normalized intensities for sample i are calculated as

$$Total\ Ion\ Current\ TIC^i = \Sigma I^i;\text{ and}$$

$$Normalized\ Intensity\ nI^i = \frac{M_{TIC} \ \times \ I^i}{TIC^i}$$

where I^i indicates the intensity of a peptide i, M_{TIC} indicates the median TIC, and TIC^i indicates the TIC for sample i.
`mod` takes in a string corresponding to the protein modification of interest (e.g. `mod="phospho"`)

Lastly, any of the normalization options can be left unused by leaving the quotation empty or specifying the argument as false in the case of `tot.current`, but a modification must be selected for any analysis to be performed by the program.
The function `Normalize` will internally call the function for quality control and perform the filtering and classification needed for the next step. The data outputted from both the quality control step and the normalization step is saved in the working directory.

3.2 Analysis

The function `Analyze` calculates the approximate and/or exact site occupancy and/or abundance of the peptides:

```
Analyze(data, stoich="", abund="", ref.state="")
```

The first argument in the function call corresponds to the output of the function `Normalyze`. `stoich` takes in either `"Exact"` or `"Approximate"`, depending on the user's preference for calculating the site occupancy of peptides for which either calculations are possible. `abund` also takes in either `"Exact"` or `"Approximate"`, depending on the user's preference for exact or approximate abundance calculations. Relative abundance results are outputted as Δ-fold-change to facilitate

downstream analyses.
Exact calculations are performed as follows:

$$\textit{Exact Abundance } A^i = \frac{I_Q^i}{I_Q^{Ref}}; \text{ and}$$

$$\textit{Exact Stoichiometry } S^i = \frac{I_M^i}{I_{NM}^i + I_M^i};$$

where: I_Q^i indicates the intensity in sample$i$ of a peptide which in the quality control module was annotated as "abundance" with respect to the user specified modification; I_Q^{Ref} indicates the intensity of the same peptide in the reference sampleRef ; $I_x^i M$ indicates the intensity in sample i of a peptide x bearing the user specified modification; I_{xNM}^i indicates the intensity in sample i of a peptide x not bearing the user specified modification.

If some of the terms above are missing in one or more sample, and the user choses to perform approximate analysis, the respective calculations are performed as follows:

$$\textit{Approximate Abundance } a^i = \frac{I_{NM}^i + I_M^i}{I_{NM}^{Ref} + I_M^{Ref}};$$

(if no peptide with unmodified chemoform is available for a given protein) or

$$\textit{Approximate Stoichiometry } s^i = \frac{I_M^i}{\bar{I_Q^i}}$$

(if no non-modified chemoform is available for a modified peptide).

where: $\bar{I_Q^i}$ indicates the mean intensity in sample$i$ of peptides which in the quality control module were annotated as "abundance" with respect to the user specified modification

It is important to note that the decision as to whether exact or approximate calculations can be performed for a given peptide is made in the quality control stage, depending on the modification of a peptide and the existence of its unmodified form in the raw dataset. As such, if for instance exact abundance calculations are selected by the user, peptides for which only approximate abundance can be calculated are eliminated in the output of the function `Analyze`. The last argument in the `Analyze` function call is `ref.state`, which provides the user with the option to normalize to a reference condition (e.g. `ref.state="Disease"`).