Towards a more accurate differential analysis of multiple imputed proteomics data with mi4limma

Marie Chion, Christine Carapito, Frédéric Bertrand

Towards a more accurate differential analysis of multiple imputed proteomics data with mi4limma

Marie Chion, Christine Carapito and Frédéric Bertrand

Abstract

Imputing missing values is common practice in label-free quantitative proteomics. Imputation replaces a missing value by a user-defined one. However, the imputation itself is not optimally considered downstream of the imputation process. In particular, imputed datasets are considered as if they had always been complete. The uncertainty due to the imputation is not properly taken into account. Hence, the `mi4p` package provides a more accurate statistical analysis of multiple-imputed datasets. A rigorous multiple imputation methodology is implemented, leading to a less biased estimation of parameters and their variability thanks to Rubin's rules. The imputation-based peptide's intensities' variance estimator is then moderated using

Marie Chion

CNRS - University of Strasbourg, France, e-mail: chion@math.unistra.fr

Christine Carapito

CNRS - University of Strasbourg, France, e-mail: ccarapito@unistra.fr

Frédéric Bertrand

University of Technology of Troyes, France, e-mail: frederic.bertrand@utt.fr

Bayesian hierarchical models. This estimator is finally included in moderated t-test statistics to provide differential analyses results.

Key words Label-free quantitative proteomics, Differential analysis, Missing values, Multiple imputation, Moderated t-testing

1 Introduction

Current statistical methods used in label-free quantitative proteomics rely on peptides' intensities, measured by liquid chromatography coupled to tandem mass spectrometry (LC-MS/MS). While major instrumental improvements over past years have allowed great progress in terms of sensitivity, dynamic range and acquisition speed, the generated datasets remain incomplete and contain a variable proportion of missing values. Usual statistical tools do not optimally consider peptides, which intensities are missing in some conditions, although they might be particularly interesting in differential analyses [1]. Imputation methods have been described and are currently applied in state-of-the-art quantification software tools [2–6]. Imputation consists of replacing a missing value with a value derived using a user-defined formula (such as the mean, the median or a value provided by an expert, thus taking into account the knowledge of the user). However, the uncertainty due to the imputation is currently not properly taken into account, as the imputed dataset is considered as if it has always been complete in further statistical analysis.

Multiple imputation [7] partially addresses this issue by generating several imputed datasets. A recommendation takes the number of imputed datasets as the percentage of missing values in the original dataset [8]. These datasets are then used to build a combined estimator of the vector of parameters of interest, by usually using the mean of the estimators among all the imputed datasets (see Figure 1). This combined estimator is known as the first Rubin's rule. The second Rubin's

rule states a formula to estimate the variance-covariance matrix of the combined estimator, decomposing it as the sum of the intra-imputation variance component and the between-imputation component.

Common statistical methods generally conclude with a study of differences in protein abundances between different conditions either using Student or Welch tests, or using more sophisticated approaches such as the moderated-t testing techniques, which are based on empirical Bayesian approaches [9].

The `mi4p` package suggests an enhanced version of this approach which accounts for the variability arising from the missing data and the imputation process. The protocol is composed of four main steps: (1) the missing values of the quantitative dataset are imputed thanks to multiple imputation algorithms, (2) leading to an estimation of the parameters of interest and their variance-covariance matrix for each peptide, (3) the variance-covariance matrices are projected to get an univariate parameter of dispersion for each peptide, (4) which are used for a more accurate testing procedure through moderated-t testing procedure.

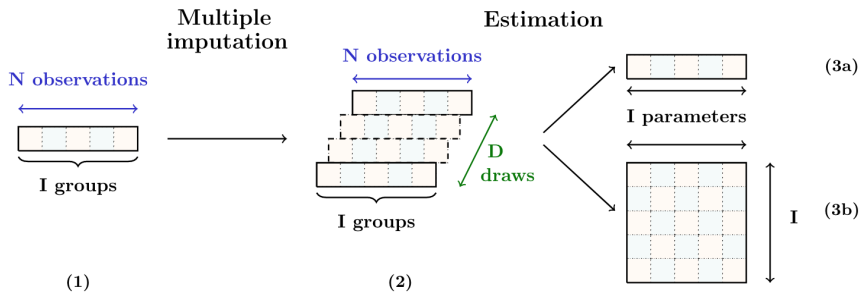

Fig. 1 Multiple imputation strategy. (1) Initial dataset with missing values. It is supposed to have **N** observations that are split into **I** groups. **(2)** Multiple imputation provides **D** estimators for the vector of parameters $\boldsymbol{\beta}$ of interest. **(3a)** The **D** estimators are combined using the first Rubin's rule to get the combined estimator. **(3b)** The estimator of the variance-covariance matrix of the combined estimator is provided by the second Rubin's rule.

2 Material

2.1 Requirements

The workflow presented in this protocol is implemented under the `R` package `mi4limma`. To use it, the `R` environment is required [10]. For a better user experience, the `R Studio` integrated development environment is recommended [11].

2.2 Data format - Quantitative data

The quantitative data should be provided as a data frame or a matrix. Rows should describe the peptides and columns the biological samples. Thus, each cell of the matrix contains the measured (or missing) abundance of the peptide in the considered sample. Although statistical analysis at the peptide-level is recommended, the methodology described in this chapter can be used at protein-level. A schematic view of the quantitative dataset is pictured in Figure 2.

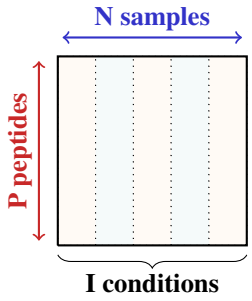

Fig. 2 Schematic representation of a quantitative dataset to be provided. There should be P rows corresponding to P peptides and N columns corresponding to N samples, which are spread between **I** conditions.

2.3 Data format - Experimental data

The experimental data should be provided as a two-columns data frame or matrix. The first column should contain the names of the biological samples and should be named "Sample.Name", the second column should contain the names of the corresponding considered condition and should be named "Condition".

2.4 Data format - Imputed data

The multiple imputed data should be provided as an array of as many matrices as the D draws used for multiple imputation. Each imputed matrix should be of the same size as the quantitative data. A schematic view of imputed data is pictured in Figure 3.

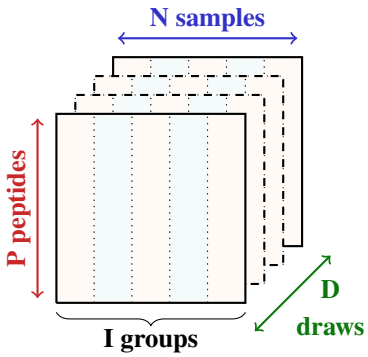

Fig. 3 Schematic representation of the imputed datasets to be provided. An array of D matrices corresponding to the D draws in the multiple imputation algorithm should be yielded. Each matrix should have P rows corresponding to P peptides and N columns corresponding to N samples, which are spread between **I** conditions.

2.5 Package install and loading

The `mi4p` package requires the following packages to be successfully installed: `BiocManager`, `DAPAR`, `emmeans`, `imp4p`, `limma` and `mice` [5, 12–16].

```
install.packages(c("BiocManager","emmeans","imp4p","mice"))
BiocManager::install(c("DAPAR","limma"))
```

The `mi4p` package itself can be downloaded from the GitHub repository (*see* **Note 1**). The following command lines should be executed in the R console:

1. Install the `devtools` R package:

```
install.packages("devtools")
```

2. Install the `mi4p` package from GitHub:

```
library(devtools)
install_github("mariechion/mi4p")
```

3. Load the `mi4p` package:

```
library(mi4p)
```

3 Methods

3.1 Multiple imputation

Multiple imputation consists of imputing D times the missing values in the original quantitative dataset. This results in D imputed datasets. Multiple imputation is provided in `mi4p`, using the `multi.impute` function:

```
multi.impute(data, metadata, imp.meth, nb.imp)
```

The `data` argument refers to the original quantitative dataset that contains missing values. The `metadata` argument refers to the experimental dataset. The `imp.meth` argument denotes the chosen multiple imputation algorithm. The `mi4p` package is for now compatible with algorithms from the `imp4p` and `mice` packages [16, 17] (*see* **Note 2**). The default algorithm is set to `imp4p`. The `nb.imp` argument describes the number of draws to be done. By default, it is equal to the percentage of missing values in the original quantitative dataset. The `multi.impute` function returns an array of as many imputed matrices as `nb.imp`.

3.2 Estimation

The objective of multiple imputation is to estimate from D drawn datasets the vector of parameters of interest and its variance-covariance matrix. Notably, accounting for multiple-imputation-based variability is possible thanks to Rubin's rules, which provide an accurate estimation of these parameters. In `mi4p`, the vectors of parameters of interest are the vectors of the peptides' intensity mean in each condition considered. There are as many vectors to be estimated (and as many corresponding variance-covariance matrices) as the number of peptides in the quantitative dataset.

1. The first Rubin's rule leads to a combined estimator of the vector of intensity means (*see* **Note 3**). To compute the estimators for all peptides in the quantification dataset, the `rubin1.all` function should be used:

```
rubin1.all(imp.data, metadata, funcmean)
```

   The `imp.data` argument refers to the array of imputed matrices and the `metadata` argument to the experimental dataset. The `funcmean` argument specifies the method for mean estimation. Here the default `funcmean` function is `meanImp_emmeans` and relies on the estimated marginal means algorithm (*see* **Note 4**). The `rubin1.all` function returns a list of estimated vector of intensity means in each condition for all peptides in the quantitative dataset (*i.e.* the length of the returned list equals the number of rows of `imp.data`). To return only the combined estimator for a specific peptide, the `rubin1.one` function should be used:

```
rubin1.one(peptide, imp.data, metadata, funcmean)
```

   The `peptide` argument denotes the row index of the considered peptide in the quantitative dataset.

2. The second Rubin's rule leads to a combined estimator of the variance-covariance matrix for each estimated vector of parameters of interest (*see* **Note 5**). The idea behind this rule is to decompose the variability into two components: the within-imputation variability and the between-imputation variability. To compute the estimators for all peptides in the quantification dataset, the `rubin2.all` function should be used:

```
rubin2.all(imp.data, metadata, funcmean, funcvar)
```

   The `imp.data` argument refers to the array of imputed matrices and the `metadata` argument to the experimental dataset. The `funcmean` and `funcvar` arguments specifies the method for mean and variance-covariance estimation respectively.

Here the default `funcmean` and `funcvar` functions are `meanImp_emmeans` and `within_variance_comp_emmeans`, both relying on the estimated marginal means algorithm (*see* **Note 6**).

To return the within-imputation component only (respectively the between-imputation component) for all peptides, the `rubin2wt.all` function (respectively the `rubin2bt.all` function) should be used:

```
rubin2wt.all(imp.data, metadata, funcvar)
rubin2bt.all(imp.data, metadata, funcmean)
```

The `rubin2.all`, `rubin2wt.all` and `rubin2bt.all` functions return lists of square matrices. The length of the list equals to the number of peptides considered, *i.e.* to the number of rows in `imp.data`. The size of the matrices is equal to the number of conditions considered, *i.e* to the number of levels of the "Condition" factor in the `metadata` dataset. To return only the combined estimator for a specific peptide, the `rubin2.one` function should be used:

```
rubin2.one(peptide, imp.data, metadata, funcmean, funcvar)
```

The `peptide` argument denotes the row index of the considered peptide in the quantitative dataset. Likewise, to return the within-imputation component and/or the between-imputation component for a specific peptide, the `rubin2wt.all` and `rubin2bt.all` functions should be used:

```
rubin2wt.one(peptide, imp.data, metadata, funcvar)
rubin2bt.one(peptide, imp.data, metadata, funcmean)
```

The `rubin2.one`, `rubin2wt.one` and `rubin2bt.one` functions return a square matrix. The size of the matrix is equal to the number of conditions considered, *i.e* to the number of levels of the "Condition" factor in the `metadata` dataset.

3.3 Projection

State-of-the-art tests, including Student's t-test, Welch's t-test and moderated t-test, rely on the variance estimation. Here, the variability induced by multiple imputation is described by a variance-covariance matrix. Therefore, a projection step is required to get a univariate variance parameter (*see* **Note 7**). This step is performed using the `proj_matrix` function:

```
proj_matrix(VarRubin.mat, metadata)
```

The `VarRubin.mat` denotes a variance-covariance matrix, as computed with `rubin2.one`, or a list of variance-covariance matrices, as computed with `rubin2.all` *see* Subheading 3.2. The `metadata` argument refers to the experimental dataset. The `proj_matrix` function returns either a variance estimator for a given peptide, or a list of variance estimators for all the peptides considered.

3.4 Moderated t-test

Several testing methods can be used. For gene expression data, the recommended method is moderated t-testing [18]. In `mi4p`, the projected variance from multiple imputation serves as an input to the usual moderated t-test. This step is performed using the `mi4limma` function:

```
mi4limma(imp.data, metadata, VarRubin.S2)
```

The `imp.data` argument refers to the array of imputed datasets. The `metadata` refers to the experimental dataset. The `VarRubin.S2` corresponds to the list of projected variance estimator for each peptide, as computed with the `proj_matrix` function *see* Subheading 3.3. The `mi4limma` function returns a list of p-values and a list of log-transformed fold change for all peptides.

3.5 Complete workflow

As an alternative to the step-by-step workflow described above, the complete `mi4p` workflow can be run with a single command:

```
mi4p.uni(data, metadata, imp.meth)
```

The `data` argument refers to the quantitative dataset, the `metadata` argument refers to the experimental dataset and the `imp.meth` specifies the imputation method to be used *see* Subheading 3.1. The `mi4p.uni` function returns a list of p-values and a list of log-transformed fold change for all peptides.

The `mi4p.uni` function includes the four steps described above: multiple imputation (*see* Subheading 3.1), estimation (*see* Subheading 3.2), projection (*see* Subheading 3.3) and moderated t-testing (*see* Subheading 3.4). A synoptic view of the functions which can be used in each step is provided in Table 1.

Table 1 Overview of the functions included in `mi4p` package

	Imputation	Estimation	Projection	Test
For one specific peptide		rubin1.one rubin2.one rubin2wt.one rubin2bt.one	proj_matrix	
For all peptides	multi.impute	rubin1.all rubin2.all rubin2wt.all rubin2bt.all	proj_matrix	mi4p
	mi4p.uni			

3.6 Example use case

A detailed example use case of the workflow presented above can be found in the vignette of the `mi4p` package. It can be accessed using the following command:

```
vignette("mi4p")
```

The vignette will be updated along with the package.

4 Notes

1. The `mi4p` package is being regularly updated. It is therefore recommended to reinstall the package to use the most recent version.
2. While the only suggested algorithms for multiple imputation are taken from `imp4p` and `mice` packages [16, 17], the user can choose any other algorithm and recall the imputed matrices in the next steps, under the aforementioned constraints, *see* Subheading 2.4.
3. Let $\hat{\beta}_{p,d}$ be the estimated vector of parameters for peptide p in the d-th imputed dataset. The first Rubin's rule gives the combined estimator for peptide p through the D imputed datasets such as:

$$\hat{\beta}_p = \frac{1}{D} \sum_{d=1}^{D} \hat{\beta}_{p,d} \tag{1}$$

4. The `meanImp_emmeans` function computes the estimated marginal means for specified factors or factor combinations in a linear model for a given imputed dataset. Estimated marginal means are also known as least-squares means or predicted marginal means and are predictions from a linear model over a reference grid.
5. The second Rubin's rule gives the combined estimator of the variance-covariance matrix for each estimated vector of parameters of interest for peptide p through the

D imputed datasets such as:

$$\hat{\Sigma}_p = \frac{1}{D}\sum_{d=1}^{D} W_d + \frac{D+1}{D(D-1)}\sum_{d=1}^{D}(\hat{\beta}_{p,d} - \hat{\beta}_p)^T(\hat{\beta}_{p,d} - \hat{\beta}_p) \quad (2)$$

where W_d denotes the variance-covariance matrix of $\hat{\beta}_{p,d}$, *i.e.* the variability of the vector of parameters of interest as estimated in the d-th imputed dataset.

6. The `within_variance_comp_emmeans` function computes the symmetric variance-covariance matrix of the marginal means estimator for a given imputed dataset.

7. To keep all the pieces of information contained in the variance-covariance matrix, an extended version of the workflow presented in this chapter to the multivariate case is currently being implemented in `mi4p`. This multivariate extension will make it possible to fully take into account the effect of the imputation process, and thus the presence of missing values, on the precision of the estimate.

Acknowledgements This work was funded through a PhD grant (2018-2021) from the Agence Nationale de la Recherche (ANR) [ANR-11-LABX-0055_IRMIA].

References

[1] O'Brien JJ, Gunawardena HP, Paulo JA, Chen X, Ibrahim JG, Gygi SP, Qaqish BF (2018) The effects of nonignorable missing data on label-free mass spectrometry proteomics experiments. Ann Appl Stat 12(4):2075–2095, https://doi.org/10.1214/18-AOAS1144

[2] Chang C, Xu K, Guo C, Wang J, Yan Q, Zhang J, He F, Zhu Y (2018) PANDA-view: an easy-to-use tool for statistical analysis and visualization of quantitative proteomics data. Bioinformatics 34(20):3594–3596, https://doi.org/10.

1093/bioinformatics/bty408

[3] Choi M, Chang CY, Clough T, Broudy D, Killeen T, MacLean B, Vitek O (2014) MSstats: An R package for statistical analysis of quantitative mass spectrometry-based proteomic experiments. Bioinformatics 30(17):2524–2526, https://doi.org/10.1093/bioinformatics/btu305

[4] Tyanova S, Cox J (2018) Perseus: A bioinformatics platform for integrative analysis of proteomics data in cancer research. In: Methods in Molecular Biology, vol 1711, Humana Press Inc., pp 133–148, https://doi.org/10.1007/978-1-4939-7493-1_7

[5] Wieczorek S, Combes F, Lazar C, Giai Gianetto Q, Gatto L, Dorffer A, Hesse AM, Coute Y, Ferro M, Bruley C, Burger T (2017) Dapar & prostar: software to perform statistical analyses in quantitative discovery proteomics. Bioinformatics 33(1):135–136, https://doi.org/10.1093/bioinformatics/btw580

[6] Webb-Robertson BJM, Wiberg HK, Matzke MM, Brown JN, Wang J, McDermott JE, Smith RD, Rodland KD, Metz TO, Pounds JG, Waters KM (2015) Review, evaluation, and discussion of the challenges of missing value imputation for mass spectrometry-based label-free global proteomics. Journal of Proteome Research 14(5):1993–2001, pMID: 25855118, https://doi.org/10.1021/pr501138h

[7] Little R, Rubin D (2019) Statistical Analysis with Missing Data, Third Edition, vol 26. https://doi.org/10.1002/9781119482260

[8] White IR, Royston P, Wood AM (2011) Multiple imputation using chained equations: Issues and guidance for practice. Statistics in Medicine 30(4):377–399, https://doi.org/10.1002/sim.4067

[9] Phipson B, Lee S, Majewski IJ, Alexander WS, Smyth GK (2016) Robust hyperparameter estimation protects against hypervariable genes and improves

power to detect differential expression. Ann Appl Stat 10(2):946–963, https://doi.org/10.1214/16-AOAS920

[10] R Core Team (2020) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, https://www.R-project.org/

[11] RStudio Team (2020) RStudio: Integrated Development Environment for R. RStudio, PBC., Boston, MA, http://www.rstudio.com/

[12] Morgan M (2019) BiocManager: Access the Bioconductor Project Package Repository. R package version 1.30.10, https://CRAN.R-project.org/package=BiocManager

[13] Lenth R (2020) emmeans: Estimated Marginal Means, aka Least-Squares Means. R package version 1.5.2-1, https://CRAN.R-project.org/package=emmeans

[14] Gianetto QG (2020) imp4p: Imputation for Proteomics. R package version 1.0, https://CRAN.R-project.org/package=imp4p

[15] Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015) limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Research 43(7):e47, https://doi.org/10.1093/nar/gkv007

[16] van Buuren S, Groothuis-Oudshoorn K (2011) mice: Multivariate imputation by chained equations in r. Journal of Statistical Software 45(3):1–67, https://doi.org/10.18637/jss.v045.i03

[17] Gianetto QG, Wieczorek S, Couté Y, Burger T (2020) A peptide-level multiple imputation strategy accounting for the different natures of missing values in proteomics data. bioRxiv p 2020.05.29.122770, https://doi.org/10.1101/2020.05.29.122770

[18] Smyth GK (12 Feb. 2004) Linear models and empirical bayes methods for assessing differential expression in microarray experiments. Statistical Applications in Genetics and Molecular Biology 3(1), https://doi.org/10.2202/1544-6115.1027