Uniwersytet Wrocławski
Wydział Matematyki i Informatyki
Instytut Matematyczny
specjalność: analiza danych

Natalia Pludra

Imputation of missing values using principal component analysis for metabolomics data

Praca licencjacka
napisana pod kierunkiem
dr Michała Burdukiewicza

Wrocław 2023

Contents

1 Introduction

Metabolites are small molecules produced during metabolic processes in biological systems. In comparison with genes and proteins, they evolve the most quickly and can change from second to second depending on, for example, eaten food or taken drugs. Metabolites can be grouped into two types: primary and secondary. The most important to organism functioning are primary metabolites involved in an organism's growth, development, and reproduction. Examples of secondary metabolites are pigments, antibiotics, peptides, and growth hormones. They may not be as crucial but the lack or insufficiency could lead to the impairment of the organism [5].

The complete set of metabolites in a biological sample is known as the metabolome. It is a complex and dynamic entity influenced by a wide range of factors, including genetics, environmental factors, and disease states. It can provide valuable insights into the biochemical and physiological state of a cell, tissue or organism.

A study that focuses on analyzing metabolomes is called metabolomics. It is a multidisciplinary field that draws upon a variety of scientific disciplines, including chemistry, biology, bioinformatics, and statistics. It is closely related to other omics fields, such as genomics and proteomics, but differs in its focus on the end products of cellular processes rather than the genes or proteins involved in those processes. Metabolomics is a powerful tool for advancing our understanding of biological systems, especially human organisms and their changes, and can potentially revolutionize fields such as medicine or pharmaceutics. Via the rapid development of technology platforms and advanced researches in this field, it is now possible to answer questions that could not be fully addressed by the other omics.

Metabolomics could be used to characterize all measurable molecules in a sample (untargeted metabolomics) or to analyze a specific set of chemically characterized metabolites (targeted metabolomics). In medicine, metabolomics can be used to identify biomarkers of disease or to study the effects of drugs and other interventions on metabolism. It can also be used to develop personalized treatment plans for individuals based on their unique metabolic profiles. For these applications, metabolites can be extracted from blood, urine, tissues and eyes and need appropriate preparation.

Metabolome generates a huge range of data which is an opportunity to do complex research and data analysis but also makes metabolomics a difficult and labor-intensive field. Because of the highly diverse molecular identity of metabolites, it is necessary to use complex analytical tools such as mass spectrometry (MS) and nuclear magnetic resonance (NMR) spectroscopy.

MS is more sensitive than NMR. The first technique quantifies metabolites at picograms per milliliter level, whereas NMR detects metabolites in the order of micrograms per milliliter. Nevertheless, NMR shows remarkable reproducibility, making it a more robust analytical approach.

However, one point to consider is that these techniques can still measure

only a fraction of the total metabolome, suggesting that there are still problems associated with metabolite extraction and resolution rather than technique sensitivity. For this reason, the combined use of NMR and MS could increase the number of detected metabolites [6].

One of the major challenges in metabolomics is the frequent presence of missing values in datasets. This problem is often caused by biological and/or technical reasons, for example, random errors during the data acquisition process, suboptimal data preprocessing, and limits of quantification or detection [10].

Imputing missing values in metabolomics data is a crucial step in data preprocessing. Missing values can significantly impact the quality of metabolomics data and reliable data analysis. For example, incomplete datasets reduce the sample size and statistical power of the analysis. Missing values can also introduce bias in biomedical studies leading to inaccurate conclusions. To avoid these negative consequences of the occurrence of missing values in metabolomics data, it is very important to handle them in a proper way. Imputing missing values provides estimated values that are based on statistical or computational methods, thus minimizes the impact of missingness on the analysis results. There are many techniques of imputation that differ due to the origin of missing values.

The simplest way is to remove whole rows in the data matrix corresponding to the missing values, but some important features which may have crucial biological functions might be lost. Another procedure that has a low efficiency is to replace the missing value with zero or with the limit of detection. There are also approaches based on mean and variance, but they cannot deal with outliers producing misleading results in the presence of outliers in the detection of markers. There are a few robust approaches that deal with both missing values and outliers such as robust SVD and weighted least square approach (WLSA). Nevertheless, they are computationally expensive. A method that shows robust performance is a k-nearest neighbor (KNN) and related methods: sequential K-nearest neighbor (SKNN) and iterative K-nearest neighbor (IKNN). Widely employed for global missing value imputations are Bayesian principal component analysis (BPCA), singular value decomposition (SVD) and partial least squares (PLS). Metabolomics data analysis uses also parametric expectation-maximization (EM) imputation and random forest (RF) imputation which shows great potential as a non-parametric method for high-dimension data sets [7] [11].

2 Methodology

2.1 Types of missing data

The problem of missing data is relatively common in almost all research. There are three main types of missing data according to the mechanisms of missingness: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). Each type named above requires different methods for imputing missing values.

2.1.1 Missing completely at random

MCAR is defined as when the occurrence of missing values is unrelated to any observed or unobserved variables. This means that the probability the data are missing is unrelated to both the particular value and the group of observed responses. There is no possibility to predict these missing values from the remaining know variables. This type reduces the sample of the study and consequently, the statistical power, but does not introduce bias. The assumption that the data are missing completely at random is generally not reasonable unless there is a comprehensive understanding of the mechanisms that generate the data [2] [4] [9]. MCAR can be described mathematically as below.

Assume that $\mathbf{y} = (y_1, y_2, ..., y_n)$ is a realisation of the random sample and δ_i is an indicator function defined by

$$\delta_i = \begin{cases} 1, & \text{if } y_i \text{ is observed} \\ 0, & \text{otherwise.} \end{cases}$$

Then values missing completely at random occurs when

$$P(\delta|\mathbf{y}) = P(\delta).$$

When examining metabolomics data, values that are considered as MCAR stem from random errors and stochastic fluctuations that occur during the data collection process, such as incomplete derivatization or ionization [10].

2.1.2 Missing at random

Data are regarded to be MAR when the probability that the values are missing depends entirely on the set of observed responses but not on the missing data itself. It is a more realistic assumption than MCAR. Data that are missing at random should not be ignored as they may or may not introduce bias to analysis [2] [4].

Mathematically, missing values are ranked as MAR when

$$P(\delta|\mathbf{y}) = P(\delta|\mathbf{y_{obs}}),$$

where $\mathbf{y_{obs}} = \{y_i : \delta_i = 1\}$ [3].

In metabolomics, this type of missing data occurs during suboptimal data preprocessing, for example, inaccurate peak detection and deconvolution of co-eluting compounds [10].

It can be challenging to differentiate between the two categories of missing values and certain imputation techniques may be applied for both MCAR and MAR data.

2.1.3 Missing not at random

MNAR data occur when the mechanism for generating missing data is related to events or factors which are not measured. This means that the probability of a value being missing is related to the unobserved data. Similarly as with MAR data, analysis of a data set containing MNAR data may or may not result in bias [4] [9].

MNAR means that

$$P(\delta|\mathbf{y}) \neq P(\delta|\mathbf{y_{obs}}).$$

The most common MNAR data in metabolomics are the values below the limit of detection [10].

Usually, two or three types of missing values occur in a dataset. Figure 1 shows a visualization of all mentioned types and combinations.

2.2 Imputation of missing values using PCA

2.2.1 Definition

Principal Component Analysis (PCA) is a statistical method for reducing dimensions in high-dimensional data matrices. It is especially useful when visualization of such data is needed. Employing this technique, it is possible to create a two-dimensional plot capturing most of the information from the n-dimensional data matrix. Principal components are also employed as predictors in a regression model. Moreover, it is a powerful tool for imputing missing values in datasets.

PCA algorithm is mainly based on linear algebra. The first principal component is a direction along which the observations from the data vary the most, which means they have the largest variance. For p-dimensional dataset $\mathbf{X}_{n\times p}$ with columns $X_1, X_2, \ldots, X_p$ it can be summarized mathematically as

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \ldots + \phi_{p1}X_p,$$

where $\phi_{11}, \phi_{21}, \ldots, \phi_{p1}$ are the principal component loadings. Scores of the first principal component can be written as

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \ldots + \phi_{p1}x_{ip}, \quad for\ i = 1, 2, ..., n,$$

where x_{ij}, $i = 1, 2, ..., n$, $j = 1, 2, ..., p$ is the element of matrix $\mathbf{X}$ from ith row and jth column.

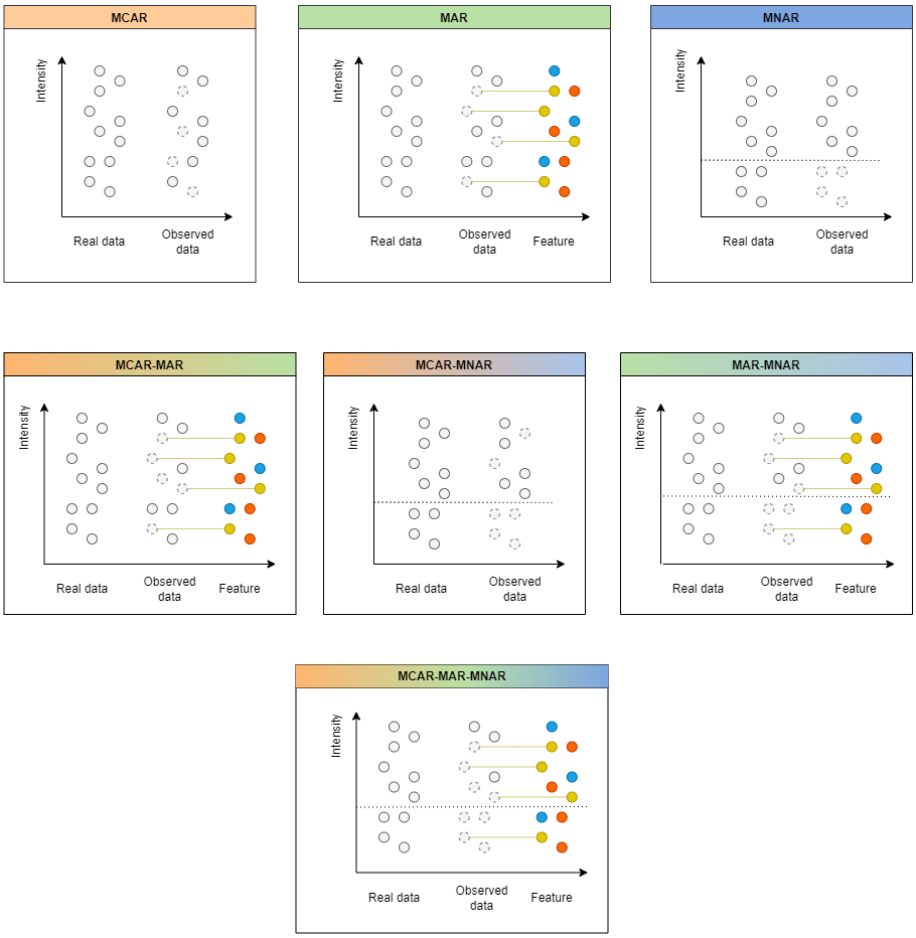

Figure 1: Visualization of types of missing values by mgr Krystyna Grzesiak

The idea is to determine such loading vector $\phi_{\mathbf{1}} = (\phi_{11}, \phi_{21}, ..., \phi_{p1})$ that $Var(Z_1)$ is the largest possible. The required assumptions are listed below:

- $\sum_{j=1}^{p} \phi_{j1}^2 = 1$. This assumption is needed to prevent arbitrarily increasing the variance;
- Variables $X_1, X_2, \ldots, X_p$ are normalized (centered to have mean zero with standard deviation equal to one). Scaling variables is important when they are measured in different units. Excluding this step leads to high first principal component loading related to the variable with the largest variance which has an effect on the results.

It consists in solving below problem

$$\max_{\phi_{11},\phi_{21},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\}$$

using eigen decomposition [8]. The solution is an eigenvector of the covariance matrix for $\mathbf{X}$ associated with the largest eigenvalue. This solution requires that the covariance matrix has real and non-negative eigenvalues and its eigenvectors are orthogonal. Below facts show that the covariance matrix achieves these assumptions.

Fact 1. *Symmetric matrix has real eigenvalues.*

Proof. Let A be a real symmetric matrix and $\lambda \in \mathbb{C}$ be an eigenvalue of A, so there exists a vector $x \neq 0$ that achieves equation

$$Ax = \lambda x. \tag{1}$$

If we take the complex conjugates of both sizes of this equation, we get

$$A\overline{x} = \overline{\lambda}\overline{x},$$

because assumption that A is real indicates $\overline{A} = A$.
A is also symmetric, so we can take the transpose of both sizes and get

$$\overline{x}^T A = \overline{\lambda}\overline{x}^T.$$

Now, let's multiply both sizes by x

$$\overline{x}^T Ax = \overline{\lambda}\overline{x}^T x.$$

Using (1) we obtain

$$\overline{x}^T \lambda x = \overline{\lambda}\overline{x}^T x$$

$$\lambda \overline{x}^T x = \overline{\lambda}\overline{x}^T x.$$

Since $x \neq 0$, we can divide both sides by $\overline{x}^T x$. Thus,

$$\lambda = \overline{\lambda},$$

which means that λ is real. □

Fact 2. *All eigenvalues of the symmetric and positive-semidefinite matrix are real and non-negative.*

Proof. Let A be a real symmetric and positive-semidefinite matrix and λ be an eigenvalue of A, so there exists a vector $x \neq 0$ that achieves equation

$$Ax = \lambda x.$$

We know from Fact 2 that all eigenvalues of A are real because A is symmetric.

Using the fact that A is symmetric and positive-semidefinite we have

$$0 \leq x^T Ax = x^T(\lambda x) = \lambda x^T x.$$

Because $x^T x$ is always a positive number, we get $\lambda \geq 0$.

□

Fact 3. *Symmetric and positive-semidefinite matrix is diagonalizable and its eigenvectors form an orthogonal basis.*

Fact 4. *The covariance matrix is symmetric and positive-semidefinite.*

It is possible to construct up to $min(n-1, p)$ principal components in $n \times p$ dimensional data matrix, where n is the number of rows (observations) and p is the number of columns (features). Every following principal component is uncorrelated with the previous ones and has the highest remaining variance. This means that, for example, loading vector $\phi_2 = (\phi_{12}, \phi_{22}, \ldots, \phi_{p2})$ should be orthogonal to ϕ_1.

Principal components can be also interpreted as surfaces that are closest to the observations in the dataset. This approach will be useful in the next subsection when handling missing values in datasets.

Let M be the number of the first principal components. Then

$$x_{ij} \approx \sum_{m=1}^{M} z_{im}\phi_{jm}$$

is the M-dimensional approximation to the observation x_{ij}. When M is large, the approximation is good. Equality $x_{ij} = \sum_{m=1}^{M} z_{im}\phi_{jm}$ occurs when all possible principal components are used: $M = min(n-1, p)$.

The problem described earlier can be characterized as looking for the above approximation with the smallest residual sum of squares

$$\min_{\mathbf{Z} \in \mathbf{R}^{n \times M}, \mathbf{\Phi} \in \mathbf{R}^{p \times M}} \left\{ \sum_{j=1}^{p} \sum_{i=1}^{n} \left(x_{ij} - \sum_{m=1}^{M} z_{im}\phi_{jm} \right)^2 \right\},$$

where $\mathbf{Z}$ is a matrix of all principal components and $\mathbf{\Phi}$ is a matrix of all loadings.

Usually, there is no need to compute all principal components, but only the first few that capture a sufficient amount of variance. This means that choosing the number of principal components depends on the data and expected results. For example, when PCA is applied to make a better visualization of the data,

it is the smallest number needed to get a good understanding of the data. However, in some cases, the number of principal components can be treated as a parameter and should be selected using, for example, cross-validation [8].

2.2.2 The proportion of variance explained

When PCA is applied to data, it is obvious that some information from the dataset is lost. The first few principal components contain most but not the whole variance from the data. It is possible to compute the proportion of variance explained (PVE) by each principal component.

The total variance present in a data set is defined as

$$\sum_{j=1}^{p} Var(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2.$$

The above formula requires that the variables have mean zero (have been centered previously).

The variance explained by the mth principal component is given by

$$\frac{1}{n} \sum_{i=1}^{n} z_{im}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2.$$

Then, the PVE of the mth principal component can be written as

$$\frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2} = \frac{\sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}.$$

Computing the cumulative PVE of the first M principal components requires summing the above formula over each of the first M PVEs. Each PVE is a positive number and PVEs of all (maximum for a given dataset) principal components sum to one [8].

2.2.3 PCA with missing values

PCA is an appropriate method for imputing missing values when the missingness is random. To solve the problem with incomplete data the second interpretation of principal components is needed: they provide the best approximation to the data matrix. It is possible to estimate a missing observation x_{ij} using $\hat{x}_{ij} = \sum_{m=1}^{M} z_{im} \phi_{jm}$.

Suppose that $\mathcal{O}$ is the set of all observed pairs of indices (i, j) and the rest of the observations x_{ij} in a dataset are missing. Then the problem to solve is as below:

$$\min_{\mathbf{Z} \in \mathbf{R}^{n \times M}, \mathbf{\Phi} \in \mathbf{R}^{p \times M}} \left\{ \sum_{(i,j) \in \mathcal{O}} \left(x_{ij} - \sum_{m=1}^{M} z_{im} \phi_{jm} \right)^2 \right\}.$$

It is difficult to solve this problem because of incomplete data, but it is possible to do that using an iterative algorithm.

The below algorithm for estimating missing values corresponds to an expectation-maximization algorithm and is often named the EM-PCA algorithm.

The iterative PCA algorithm is the following:

1. Complete missing values with zeros so the elements of the new matrix $\tilde{X}$ are
$$\tilde{x}_{ij} = \begin{cases} x_{ij}, & \text{if } (i,j) \in \mathcal{O} \\ 0, & \text{if } (i,j) \notin \mathcal{O} \end{cases}$$

2. Repeat steps (a)-(c) until (2) falls below chosen threshold ϵ

   (a) Solve
   $$\min_{\mathbf{Z} \in \mathbf{R}^{n \times M}, \mathbf{\Phi} \in \mathbf{R}^{p \times M}} \left\{ \sum_{j=1}^{p} \sum_{i=1}^{n} \left(\tilde{x}_{ij} - \sum_{m=1}^{M} z_{im} \phi_{jm} \right)^2 \right\}. \tag{2}$$

   (b) Replace each $\tilde{x}_{ij}$ for $(i,j) \notin \mathcal{O}$ with $\sum_{m=1}^{M} \hat{z}_{im} \hat{\phi}_{jm}$, where $\hat{z}_{im}, \hat{\phi}_{jm}$ are the solutions of (1).

   (c) Compute
   $$\sum_{(i,j) \in \mathcal{O}} \left(x_{ij} - \sum_{m=1}^{M} \hat{z}_{im} \hat{\phi}_{jm} \right)^2. \tag{3}$$

3. Complete original matrix X with estimated missing values $\tilde{x}_{ij}$, $(i,j) \notin \mathcal{O}$ [8].

2.2.4 Example

Assume that $\mathbf{X}$ is a matrix with $n = 10$ rows and $p = 2$ columns

$\mathbf{X_1}$	$\mathbf{X_2}$
5.0	4.0
4.0	NA
3.0	2.0
NA	3.0
2.0	1.0
1.0	NA
3.0	5.0
2.0	NA
4.0	1.0
NA	2.0

Matrix $\mathbf{X}$ contains five missing values (NA) which we want to replace.
First, we need to standardize the dataset. To do that we will:

- calculate the mean of each column and subtract it from each value of that column,
- divide the result by the standard deviation of each column.

$$x_{new} = \frac{x - \overline{x}}{\sigma_x}$$

	$\mathbf{X_1}$	$\mathbf{X_2}$
mean	3	2.57
sd	1.31	1.51

Dataset after standardizing looks like this

$\mathbf{X_1}$	$\mathbf{X_2}$
1.5275	0.9449
0.7638	NA
0.0000	-0.3780
NA	0.2835
-0.7638	-1.0394
-1.5275	NA
0.0000	1.6063
-0.7638	NA
0.7638	-1.0394
NA	-0.3780

Second, we should replace *NA* with chosen value. Let's use 0 in our example which will result in the below matrix

$\mathbf{X_1}$	$\mathbf{X_2}$
1.5275	0.9449
0.7638	0.0000
0.0000	-0.3780
0.0000	0.2835
-0.7638	-1.0394
-1.5275	0.0000
0.0000	1.6063
-0.7638	0.0000
0.7638	-1.0394
0.0000	-0.3780

Now we can calculate PCA.

1. First step is calculating the covariance matrix for the dataset using the below formula

$$Cov(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})$$

which results in the matrix:

	$\mathbf{X_1}$	$\mathbf{X_2}$
$\mathbf{X_1}$	0.7777778	0.1603751
$\mathbf{X_2}$	0.1603751	0.6666667

2. Then we will calculate eigenvalues and eigenvectors of the above covariance matrix solving equation $Av = \lambda v$, where λ is called eigenvalue associated with eigenvector v.

Eigenvalues of our covariance matrix:

0.8919472 0.5524972

and eigenvectors:

$\mathbf{v_1}$	$\mathbf{v_2}$
-0.8146554	0.5799453
-0.5799453	-0.8146554

We decided to use only the first principal component, so we need one eigenvector in our further analysis. We are interested in an eigenvector that is associated with the largest eigenvalue: v_1.

3. Orthogonal projection to principal component

We have to calculate the orthogonal projection of missing values to the computed principal component. To do that we will use an orthogonal projection matrix in a base formed by eigenvectors from our previous computations. We have to multiply matrix PMP^{-1} by each row of matrix $\mathbf{X}$, where P is the transition matrix to eigenbasis and M denotes the orthogonal projection matrix on the x-axis in the standard basis.

Then we have to reverse standardization: multiply the new data by the standard deviation of the original data (each column independently) and add the mean of the original data.

$\mathbf{X_1}$	$\mathbf{X_2}$
4.911839	4.361020
3.337761	3.116973
2.766195	2.833557
2.773909	2.715573
1.693373	2.069825
1.038764	1.480339
3.993671	3.707385
1.805096	2.025884
3.020700	3.014736
2.301454	2.379236

Now we can replace missing values in the original dataset with the results of PCA.

Our dataset looks like this

$\mathbf{X_1}$	$\mathbf{X_2}$
5.000000	4.000000
4.000000	3.116973
3.000000	2.000000
2.773909	3.000000
2.000000	1.000000
1.000000	1.480339
3.000000	5.000000
2.000000	2.025884
4.000000	1.000000
2.301454	2.000000

To get better results we will repeat the above steps until the difference between the previous and next estimated value is very small. The final result:

$\mathbf{X_1}$	$\mathbf{X_2}$
5.000000	4.0000000
4.000000	4.0438511
3.000000	2.0000000
2.890013	3.0000000
2.000000	1.0000000
1.000000	-0.3734165
3.000000	5.0000000
2.000000	1.0990060
4.000000	1.0000000
2.146649	2.0000000

Figure 2 shows the change in the estimated values of missing entries and the lines that fit the data best in each iteration. We can see that with each iteration, the estimated points are closer to the corresponding line.

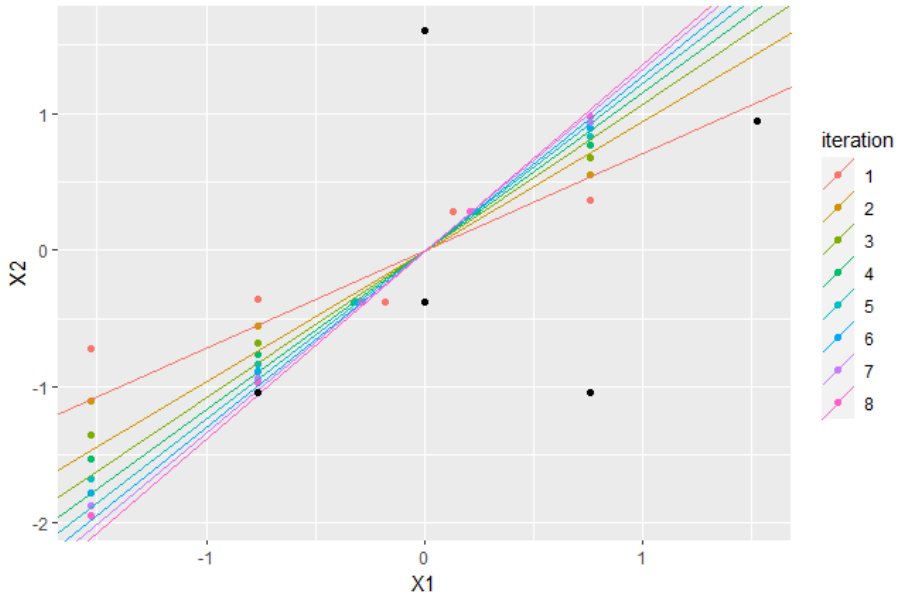

Figure 2: Observed values (black points) and estimations of missing values in each iteration (colored points). The lines show the first principal components.

3 Simulation

To show the efficiency of imputing missing values using PCA, we will apply this algorithm to real data. The dataset used in this simulation comes from Metabolomics Workbench - a platform that serves as an international repository for metabolomics data. This data is available at the NIH Common Fund's National Metabolomics Data Repository (NMDR) website, the Metabolomics Workbench, *https://www.metabolomicsworkbench.org*, where it has been assigned Project ID PR001022. The data can be accessed directly via its Project DOI: 10.21228/M8KT26. This work is supported by NIH grant, U2C- DK119886.

The dataset has 187 rows and 66 columns. In the simulation, we randomly remove values from this dataset in three proportions so that the missing values form 5%, 10% and 15% of the whole dataset. Then, we impute missing values using three methods: EM-PCA, regularized iterative PCA, and imputation missing values with the mean of each variable (column) in the dataset.

To impute missing values based on the iterative PCA algorithm we use *imputePCA()* function from the *missMDA* package from the R statistical software. We compare the results of the iterative PCA algorithm ("EM"), which was described in Chapter 2, and the regularized iterative PCA algorithm ("Regularized") available as a parameter "method" in this function. The second method prevents the occurrence of overfitting problem, which EM-PCA doesn't handle, especially when there are many missing values. These two algorithms are very similar, but they differ in the imputation step (in the algorithm described in the 2.2.3 section it is step 2(b)): regularized iterative PCA requires replacing the missing value with regularized (shrunk) fitted value. The aim of this procedure is to remove the noise of the last dimensions by subtracting the noise variance estimated by the mean of the last eigenvalues [1].

We estimated a number of dimensions for PCA with *estim_ncpPCA()* function from the *missMDA* package, but the returned number was 0. In this case, we compare the results of PCA with the number of dimensions equal to $2, 4, 6, 8$.

Figure 3 shows the mean squared error computed from the formula

$$MSE = \frac{\sum_{(i,j)\notin\mathcal{O}} (x_{ij} - \tilde{x}_{ij})^2}{k},$$

where k is the number of missing values. Each row of the graph contains results with a different number of principal components used in the algorithms and each column contains results of different proportions of missing values in the dataset.

We can see that in cases with 2 and 4 principal components, the boxplots are very similar for each method. This means that when the number of dimensions is small, PCA imputes missing values that are very close to the mean of the relevant variable. Increasing this number, we get better results for both PCA methods - the mean squared error values decrease. Analyzing the influence of the number of missing values in the dataset, we can see that the shortest

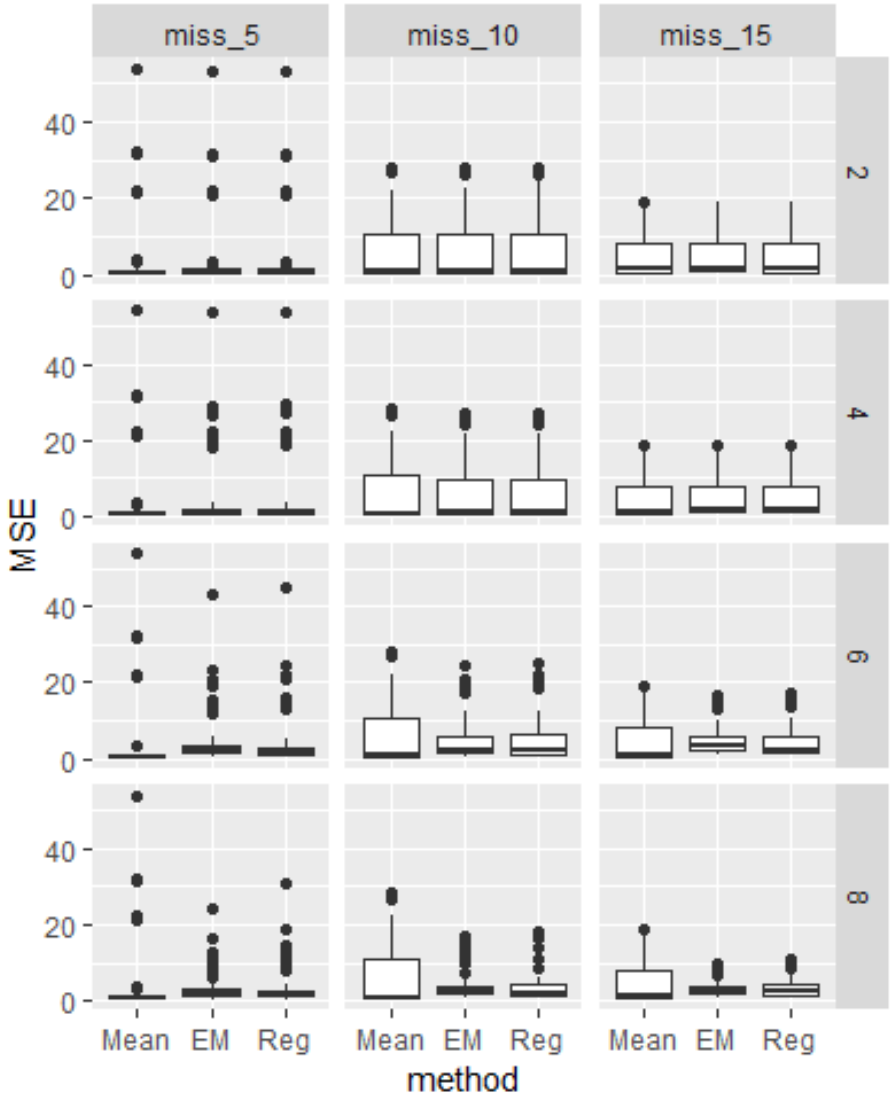

Figure 3: Boxplots of mean squared error of all three methods split by the number of principal components used in iterative PCA and three cases: when there are 5%, 10% and 15% missing values in the dataset. Boxplots were made based on 100 repetitions.

interquartile ranges occur when there are 5% of missing values. In this case, there are also a lot of outliers. Comparing the three methods of imputing missing values, we see that when the number of principal components is sufficiently large, the interquartile ranges are shorter for PCA methods than for the mean method.

Medians are very small in all cases. Both PCA methods have similar results, but we can conclude that regularized PCA is slightly better (in most cases values have a smaller median).

Figure 4 shows the time of running commands responsible for imputing missing values. We can see that the times needed to impute missing values with the mean are very close to zero in all cases. Even when there are 15% of missing values in the dataset, the program runs fast, which is a strength of this method. Analysis of times of running iterative PCA leads to a conclusion that the time the program needs to run increases when the number of missing values increases. Adding more principal components also makes the program run more slowly. Comparing two iterative PCA algorithms, we can see that regularized PCA is more efficient - it takes less time than EM-PCA in most cases.

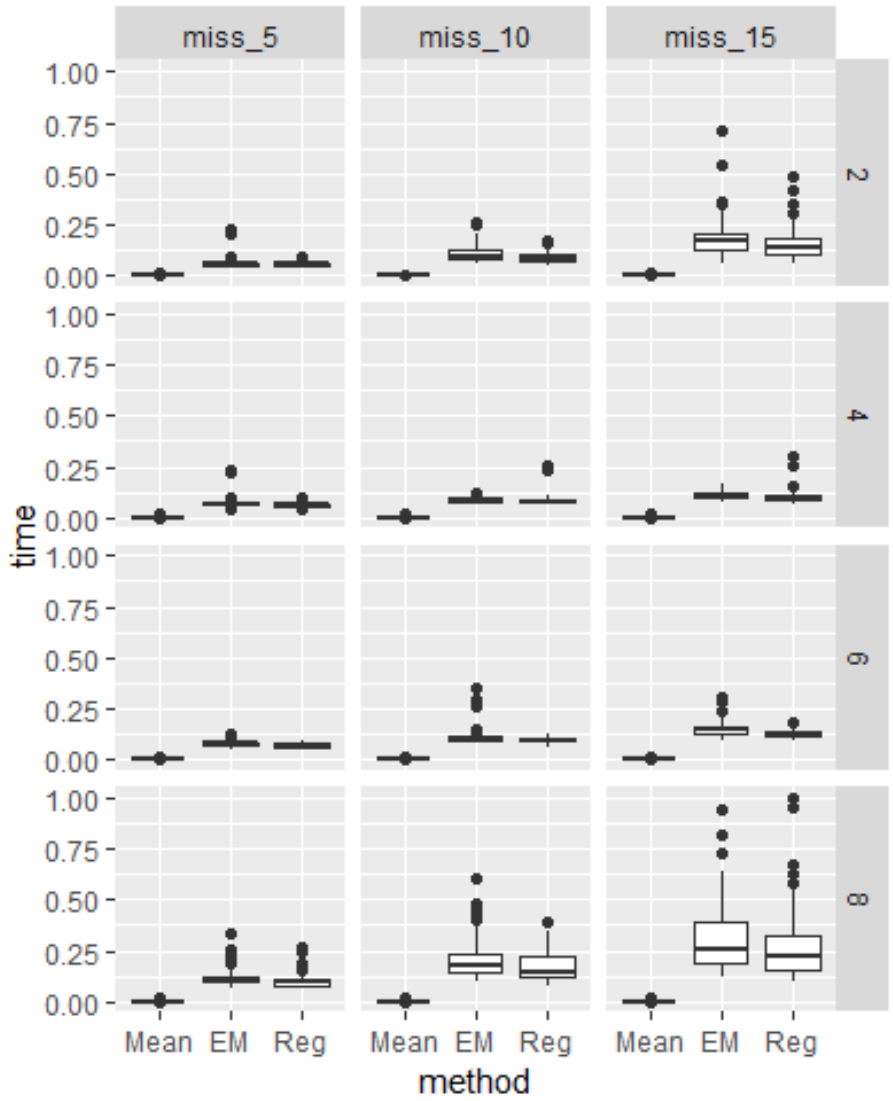

Figure 4: Boxplots of time of imputation missing values by all three methods split by the number of principal components used in iterative PCA and three cases: when there are 5%, 10% and 15% missing values in the dataset. Boxplots were made based on 100 repetitions.

4 Conclusion

To sum up the conducted simulation, we point strengths and limitations of imputing missing values using PCA.

First of all, the results show that iterative PCA methods better estimate missing values than the mean method when we use a larger number of principal components. However, by increasing this number, we need to expect an increase in the time of running the program. PCA is not the best option when we want to use a small number of principal components - we obtain the same results as using the mean method but it takes more time to run. This simulation also shows that regularized PCA performs better than EM-PCA - it has usually lesser MSE and lesser running time. The difference is not large, but the regularized method has also an important advantage - it handles the overfitting problem.

As we see in the simulation, the PCA method handles missing values well. It could be used in studies as an effective method for the imputation of missing values, but it is recommended to conduct more research and simulations on real data.

5 R code for the simulations

2.2.4 Example

```
library(tidyr)
library(ggplot2)
library(dplyr)

X1 <- c(5, 4, 3, NA, 2, 1, 3, 2, 4, NA)
X2 <- c(4, NA, 2, 3, 1, NA, 5, NA, 1, 2)
X <- matrix(c(X1, X2), ncol = 2)

# Standardize data
X_norm <- scale(X)

# Replace NA with 0
X_replaced_0 <- X_norm
X_replaced_0[is.na(X_norm)] <- 0

# PCA
cov_matrix <- cov(X_replaced_0)
colnames(cov_matrix) <- c("X1", "X2")
rownames(cov_matrix) <- c("X1", "X2")

D <- eigen(cov_matrix)
eigenvec <- D$vectors[,which.max(D$values)]
P <- D$vectors
M <- matrix(c(1,0,0,0), nrow=2)
projection_matrix <- P%*%M%*%solve(P)
after_pca <- t(projection_matrix%*%t(X_replaced_0))
after_pca <- data.frame(after_pca)
colnames(after_pca) <- c("X1", "X2")

# Reverse standardization
attr_x <- attributes(X_norm)
data_after_pca <- after_pca*attr_x[['scaled:scale']] +
  attr_x[['scaled:center']]

# Replace NA with results of PCA
data_new <- X
data_new[is.na(X)] <- data_after_pca[is.na(X)]
data_new <- data.frame(data_new)
colnames(data_new) <- c("X1", "X2")

# Iterations
do_pca <- function(dataset, missing_values){
  new_values <- rep(0, sum(missing_values))
```

```
  r <- 1
  results <- list()
  i <- 1

  while(r > 10^(-2)){
    cov_matrix <- cov(dataset)
    D <- eigen(cov_matrix)
    eigenvec <- D$vectors[,which.max(D$values)]
    P <- D$vectors
    M <- matrix(c(1,0,0,0), nrow=2)
    projection_matrix <- P%*%M%*%solve(P)

    after_pca <- t(projection_matrix%*%t(dataset))
    r <- sum((new_values - after_pca[missing_values])^2)
    new_values <- after_pca[missing_values]
    dataset[missing_values] <- after_pca[missing_values]
    colnames(dataset) <- c("x", "y")
    results[[i]] <- list(dat = dataset,
                         slope = eigenvec[2]/eigenvec[1])
    i <- i+1
  }
  results
}

prepare_data <- function(dataset){
  missing_values <- is.na(dataset)

  # Standardizing data
  standardized_data <- scale(dataset)

  # Replacing NA with 0
  replaced_0 <- standardized_data
  replaced_0[missing_values] <- 0

  attr_x <- attributes(standardized_data)

  list(dat = replaced_0, miss_val = missing_values, attr = attr_x)
}

my_data <- prepare_data(X)[["dat"]]
my_miss_val <- prepare_data(X)[["miss_val"]]

results <- do_pca(my_data, my_miss_val)
```

```r
results_slope <- lapply(results,
                        function(i) data.frame(slope = i[[2]])) %>%
  bind_rows() %>%
  mutate(iter = 1:n())

results_dat <- lapply(1:length(results),
                      function(id) data.frame(results[[id]][1],
                                              iter = id)) %>%
  bind_rows()

estimated_miss <- as.data.frame(results[[1]]$dat)[!rowSums(is.na(X))
                                                  > 0,]

p <- ggplot()+
  geom_point(data = results_dat, aes(x = dat.x, y = dat.y,
                                     colour = as.factor(iter)))+
  geom_abline(data = results_slope, aes(slope = slope, intercept = 0,
                                        color = as.factor(iter)))+
  geom_point(data = estimated_miss,
             aes(x = x, y = y), color="black")+
  labs(x="X1", y="X2", color="iteration")


input_na_using_PCA <- function(dataset){
  missing_values <- is.na(dataset)

  # Standardizing data
  standardized_data <- scale(dataset)

  # Replacing NA with 0
  replaced_0 <- standardized_data
  replaced_0[missing_values] <- 0

  # PCA
  after_pca <- do_pca(replaced_0, missing_values)
  after_pca <- after_pca[[length(after_pca)]][[1]] # last iteration

  # Reversing standardization
  attr_x <- attributes(standardized_data)
  after_pca <- after_pca*(attr_x[['scaled:scale']]) +
    attr_x[['scaled:center']]

  # Replacing NA with results of PCA
  data_new <- dataset
```

```
  data_new[missing_values] <- after_pca[missing_values]
  data_new
}

# input_na_using_PCA(X)
```

3 Simulation

```
library(missMDA)
library(dplyr)
library(tidyr)
library(ggplot2)

# Preparing data
dane <- read.table(file = "C:\\Users\\natal\\Desktop\\PracaLicencjacka\\AN002512.txt",
                   sep="\t", header = TRUE)

dane <- dane[,3:ncol(dane)]
colnames(dane) <- paste0("X", 1:66)
dane_m <- as.matrix(dane)

num_obs <- dim(dane)[1]*dim(dane)[2]

# Imputation of missing values
compare_methods <- function(miss_prop, dat, nb){
  # Removing random elements
  dane_with_miss <- dat
  dane_with_miss[sample(1:num_obs, miss_prop*num_obs)] <- NA
  miss_ind <- is.na(dane_with_miss)
  atr <- attributes(scale(dane_with_miss))

  # Imputation of missing values using PCA
  # Regularized
  start_pca_reg <- Sys.time()
  # nb_reg <- estim_ncpPCA(dane_with_miss, scale=TRUE,
  #                        method = "Regularized")
  # comp_reg <- imputePCA(dane_with_miss, ncp=nb_reg$ncp, scale=TRUE,
  #                       method = "Regularized")
  comp_reg <- imputePCA(dane_with_miss, ncp=nb, scale=TRUE,
                        method = "Regularized")
  end_pca_reg <- Sys.time()

  # EM
  start_pca_em <- Sys.time()
```

```r
  # nb_em <- estim_ncpPCA(dane_with_miss, scale=TRUE, method = "EM")
  # comp_em <- imputePCA(dane_with_miss, ncp=nb_em$ncp, scale=TRUE,
  #                      method = "EM")
  comp_em <- imputePCA(dane_with_miss, ncp=nb, scale=TRUE,
                       method = "EM")
  end_pca_em <- Sys.time()

  # Imputation of missing values using mean
  start_mean <- Sys.time()
  imputed_mean <- dane_with_miss
  for(i in 1:dim(imputed_mean)[2]){
    imputed_mean[is.na(imputed_mean[,i]),i] <- atr$`scaled:center`[i]
  }
  end_mean <- Sys.time()

  # MSE
  mse_pca_reg <- sum((comp_reg$completeObs - dat)^2)/sum(miss_ind)
  mse_pca_em <- sum((comp_em$completeObs - dat)^2)/sum(miss_ind)
  mse_mean <- sum((imputed_mean - dat)^2)/sum(miss_ind)

  c(mse_pca_reg, mse_pca_em, mse_mean,
    end_pca_reg-start_pca_reg, end_pca_em-start_pca_em,
    end_mean-start_mean)
}

# Results
repeat_simulation <- function(nb){
  k <- 1
  res_df <- data.frame(miss_5 = 0, miss_10 = 0, miss_15 = 0, nb = 0,
                       iter = 0, method = 0)
  set.seed(8)
  for(i in 1:100){
    res_df[k:(k+5),1:3] <- as.data.frame(lapply(c(0.05, 0.1, 0.15),
                                                compare_methods,
                                                dat = dane_m,
                                                nb = nb))
    res_df[k:(k+5),4] <- nb
    res_df[k:(k+5),5] <- i
    res_df[k:(k+5),6] <- c("PCA_Reg", "PCA_EM", "Mean",
                           "T_pca_reg", "T_pca_em", "T_mean")
    k <- k+6
  }
  res_df
}
```

References

[1] Julie Josse and François Husson. "Handling missing values in exploratory multivariate data analysis methods". en. In: 153.2 (2012).

[2] Hyun Kang. "The prevention and handling of the missing data". In: *Korean Journal of Anesthesiology* 64.5 (May 2013). Publisher: The Korean Society of Anesthesiologists, pp. 402–406. DOI: 10.4097/kjae.2013.64.5.402. URL: https://synapse.koreamed.org/articles/1155616 (visited on 04/23/2023).

[3] Jae-Kwang Kim. "Statistical Methods for Handling Incomplete Data Chapter 2: Likelihood-based approach". en. In: ().

[4] Christina Mack, Zhaohui Su, and Daniel Westreich. *Types of Missing Data*. en. Publication Title: Managing Missing Data in Patient Registries: Addendum to Registries for Evaluating Patient Outcomes: A User's Guide, Third Edition [Internet]. Agency for Healthcare Research and Quality (US), Feb. 2018. URL: https://www.ncbi.nlm.nih.gov/books/NBK493614/ (visited on 04/23/2023).

[5] *Metabolite*. en-US. Oct. 2019. URL: https://www.biologyonline.com/dictionary/metabolite (visited on 04/08/2023).

[6] *Metabolite - an overview — ScienceDirect Topics*. URL: https://www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/metabolite (visited on 04/08/2023).

[7] Md. Shahjaman et al. "rMisbeta: A robust missing value imputation approach in transcriptomics and metabolomics data". en. In: *Computers in Biology and Medicine* 138 (Nov. 2021), p. 104911. ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2021.104911. URL: https://www.sciencedirect.com/science/article/pii/S0010482521007058 (visited on 04/15/2023).

[8] Fariha Sohil, Muhammad Umair Sohali, and Javid Shabbir. "An introduction to statistical learning with applications in R: by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, New York, Springer Science and Business Media, 2013, $41.98, eISBN: 978-1-4614-7137-7". en. In: *Statistical Theory and Related Fields* 6.1 (Jan. 2022), pp. 87–87. ISSN: 2475-4269, 2475-4277. DOI: 10.1080/24754269.2021.1980261. URL: https://www.tandfonline.com/doi/full/10.1080/24754269.2021.1980261 (visited on 05/03/2023).

[9] Zachary Warnes. *Missing Value Handling — Missing Data Types*. en. Jan. 2023. URL: https://towardsdatascience.com/missing-value-handling-missing-data-types-a89c0d81a5bb (visited on 04/23/2023).

[10] Runmin Wei et al. "Missing Value Imputation Approach for Mass Spectrometry-based Metabolomics Data". en. In: *Scientific Reports* 8.1 (Jan. 2018). Number: 1 Publisher: Nature Publishing Group, p. 663. ISSN: 2045-2322. DOI: 10.1038/s41598-017-19120-0. URL: https://www.nature.com/articles/s41598-017-19120-0 (visited on 04/15/2023).

[11] Machelle D. Wilson et al. "Imputation of Missing Values for Multi-Biospecimen Metabolomics Studies: Bias and Effects on Statistical Validity". en. In: *Metabolites* 12.7 (July 2022), p. 671. ISSN: 2218-1989. DOI: 10.3390/metabo12070671. URL: https://www.mdpi.com/2218-1989/12/7/671 (visited on 04/15/2023).