Final Internship Report

DiOGenes Clinical Data Analysis

Maria Paula Caldas Rivera — M1 Economics
Toulouse School of Economics
2014-2015

Host institution	Inserm UMR 1048 - I2MC
Supervisors	Nathalie Viguerie and Nathalie Villa-Vialaneix
Academic tutor	Lucie Bottega

Contents

1. Introduction

This summer I worked as an intern in the Obesity Research Laboratory in the Institute of Cardiovascular and Metabolic Diseases (I2MC), under the joint supervision of Nathalie Viguerie (I2MC, INSERM) and Nathalie Villa-Vialaneix (MIAT, INRA). The objective of my internship was to conduct preliminary research on imputation methods for the clinical data coming from the DiOGenes trial, a longitudinal dietary protocol that took place in 2006.

In this report I explain the details of my work, and the analyses undertaken. It is organized as follows: Section 2 gives a brief presentation of my host institution, section 3 presents the DiOGenes project and the objective of my internship, finally, section 4 presents the statistical analysis performed and the main results.

2. Inserm, I2MC and the Obesity Research Laboratory

2.1. Inserm

The French National Institute of Health and Medical Research (Inserm) is a public scientific and technological institute which operates under the joint authority of the French Ministry of Health and French Ministry of Research[1]. The Inserm has a wide range of facilities, including 289 research units, 80% of which are set up within a hospital or university. There are currently three research units of the Inserm based at Toulouse. One of those units, the Institute of Cardiovascular and Metabolic Diseases, acted as my host institution for the period of April-June.

2.2. I2MC

The new Institute of Cardiovascular and Metabolic Diseases (I2MC) is based in Toulouse, France. It was founded in 2011 as a joint venture between the Inserm and the Université Toulouse III - Paul Sabatier, and it is based in the University Hospital Rangueil. Its research activity focuses on metabolic, cardiovascular and renal diseases. In particular, the main feature of I2MC is the gathering of scientists together with clinicians working on metabolic risk factors and their cardiovascular complications[2].

The I2MC is composed of 13 research teams working around three main areas:

- Intestine, adipose tissue, obesity and diabetes;
- Thrombosis, atherosclerosis and vessels;
- Heart and kidney.

2.3. The Obesity Research Laboratory

The Obesity Research Laboratory (team 4) is one of the research teams of the I2MC. The team works on the consequences of the excess of fat mass observed in obesity and aims at understanding the biological determinants and molecular mechanisms of obesity-related metabolic complications.

[1] Inserm website
[2] I2MC website

3. Context and Objective of the Internship

3.1. DiOGenes protocol

The DiOGenes project (Diet, Obesity and Genes) was a multi-disciplinary, multi-center research project to advance the understanding of how obesity can be prevented and treated from a dietary perspective. One of the central pillars of the project was the Pan-European Weight Loss Study, one of the largest longitudinal dietary interventions worldwide. It consisted of a randomised controlled trial comparing the effect of reduced-fat diets varying in glycemic index (high vs. low) and protein content (high vs. normal) on bodyweight in overweight and obese subjects after an initial 8% weight loss[1], and was carried out in 8 different centers across Europe.

The dietary intervention consisted of two phases. The first phase was an 8-week-low-calorie diet with the objective of 8% or more weight loss. In the second phase, which lasted 26 weeks, successful individuals were randomized into one of five *ad libitum* weight maintenance diets:

- four diets combining high and low protein content with high and low glycemic index of carbohydrates;
- a control diet according to the national dietary guidelines on healthy diets.

The measurements taken at the beginning and the end of phase one are referred to as CID1 and CID2, respectively. CID3 represents the measurements taken in after the 26-week *ad libitum* diet.

A large amount of data originated from this protocol. These data came both from clinical and psychological examinations as well as from collected tissue samples.

Clinical Data

The clinical data collected during the experimental protocol resulted in a data set of 614 observations (participants) and 444 variables. It is an example of what is commonly called *mixed-type data*, containing both numerical and categorical variables.

The variables were organized chronologically and labelled by time step. Although I worked mainly with the data obtained at CID1, I also explored the missing data structure of the variables at the time steps CID2 and CID3.

[1] https://clinicaltrials.gov/ct2/show/NCT00390637

3.2. Objective of the internship

The primary objective of the internship was to conduct an exploratory analysis of the clinical data collected during the DiOGenes trial, and to study different imputation methods that could be used for the clinical data.

My work consisted of three main steps:

- Perform an exploratory analysis of the clinical data, focusing on the time steps CID1, CID2 and CID3
- Research the methods that can be used to handle missing values for the clinical data.
- Do a comparison of different imputation methods for the data at CID1.
  - Analysis through PCA
  - Fit of the different methods

3.3. Tools and methods

R and reproducible research

The analysis of the data was performed using the statistical software R and the interface RStudio. One of the main advantages of using RStudio is that it enables the creation of dynamic reports thanks to its built-in RMarkdown authoring format. These reports can be generated in different output formats (html, pdf, tex or Word) and contain the code, output and graphics created in R.

The idea behind using R Markdown for my reports is that it guarantees the reproducibility of my analyses and facilitates collaboration with my supervisors. Since the document is compiled with the whole code being executed, the figures and graphics in the final report will be faithful to my code.

An extract of my reports can be found in Appendix A.

Packages

For the imputation of the data, I used the packages **VIM**, **missForest**, **impute** (from Bioconductor) and **mixOmics**. The first two packages also include some nice visualization tools which I used to explore the structure of the missing values.

Principal Component Analysis was performed using the package **FactoMineR**.

4. Statistical Analysis and Results

4.1. Missing data and imputation methods

Missing data are unavoidable in clinical research and can have a significant effect on the conclusions that can be drawn from analyzing the data. One simple and popular approach is to conduct a complete-case analysis, which means removing all the observations that have at least one missing value for one of the variables. This method is not advisable in most situations, and may result in a biased sub-sample of the original data if the missing values are not randomly distributed.

Moreover, as is the case for the DiOGenes clinical data, the removal of observations with missing values may not even be a possibility. Even at CID1, which is the time step that has the least missing values, there is no observation for which we have complete data.

Therefore, different missing data imputation methods need to be analysed. However, before deciding which methods to explore, it is important to identify first the mechanism behind the missing data.

4.1.1. Types of missing data

In order to determine which imputation methods can be used in a particular data set, it is necessary to understand the underlying structure and relationships of the missing values. After all, the probability of an item to be missing may depend both on other observed or non-observed variables, as well as on its own value[1]. Little and Rubin (2002) defined three *missing data mechanisms* .

Following the notation of Todorov (2012), let us denote the complete data as $X = \{X_o, X_m\}$ where X_o and X_m are the observed and missing parts, respectively. Let R be the indicator matrix encoding the fact that the observation is or is not missing. Each element of R is equal to 1 if the corresponding element of X is missing, and 0 otherwise. The probability of R conditional on the values of the observed and missing values of X,

$$Pr(R|X_o, X_m)$$

is what we will define as the *missingness mechanism.* Therefore, missing data can be defined as:

1. **Missing at random (MAR)** implies that missing pattern of X may depend on the observed part of X, but it does not depend on the unobserved part itself.

[1] For instance, some people were excluded from the experiment after CID2 because they did not lose enough weight

$$Pr(R|X) = Pr(R|X_o)$$

The missing data mechanism is said to be *ignorable* if the data are MAR and the parameters governing the missing-data mechanism are distinct from the parameters in the model to be estimated.

2. **Missing completely at random (MCAR)** is a special case of MAR, which occurs when the distribution does not depend on X_o either. This implies that the pattern of missing values is completely random and does not depend on any variable.

   $$Pr(R|X) = Pr(R)$$

   It is therefore only in this case that complete-case analysis may be used since it will not lead to a biased sub-sample. In general, it is very rare to find data that have a MCAR missingness structure.

   One example of MCAR is when patients, by accident, forget to check a specific item in a questionnaire.

3. **Missing not at random (MNAR)** implies that the distribution of the missing data depends of X_m. In other words, there is an unknown process which is generating the missing values. MNAR can appear in one of the two following versions (or a combination thereof):

   - Missingness that depends on the unobserved predictors.
   - Missingness that depends on the missing value itself.

   An example of data that may be of the type MNAR is that of revenues, where wealthier individuals avoid reporting their income.

Therefore, the exploration of missing values is crucial for determining their dependencies to other variables.

4.1.2. The structure of the missing data

The DiOGenes clinical data has large number of missing values, which are missing mainly because of two reasons. First, there are subjects that stopped participating in the trial after a certain amount of time (see Table 4.1). This means that the data has, to some extent, a *monotone* missing data pattern. The reason behind the missingness may not be random in this case: it could be that the patients who dropped out were those that were the least successful at losing weight, the ones who were put in a particular diet, etc. If this is the case, then the missing value mechanism at these later stages would be MNAR.

The second reason behind the missingness appears to come from the difficulty of administering certain tests. An example of this can be observed for the variables related to the *Basal Metabolic Rate Test (BMR)*, have more than 80 percent missing cases. In this particular case, we assume the data to be of type MAR because the missingness comes

	CID1	CID2	CID3
Number of Observations	614	614	614
Number of Variables	79	67	88
Percentage of missing values	16.56	24.47	40.42

Table 4.1.: Characteristics of main time steps

from the fact that this test in neither a confortable or easy procedure for patients. Moreover, there is no reason to believe that the missing cases depend on the other unobserved predictors, or in the missing value of the BMR test itself.

As can be seen in Figure 4.1, there are strong dependencies between the missing values for certain categories of variables. At the final stage of the analysis, I used these "combinations" of missing values to recreate the missingness structure at CID1(see section 4.3.2).

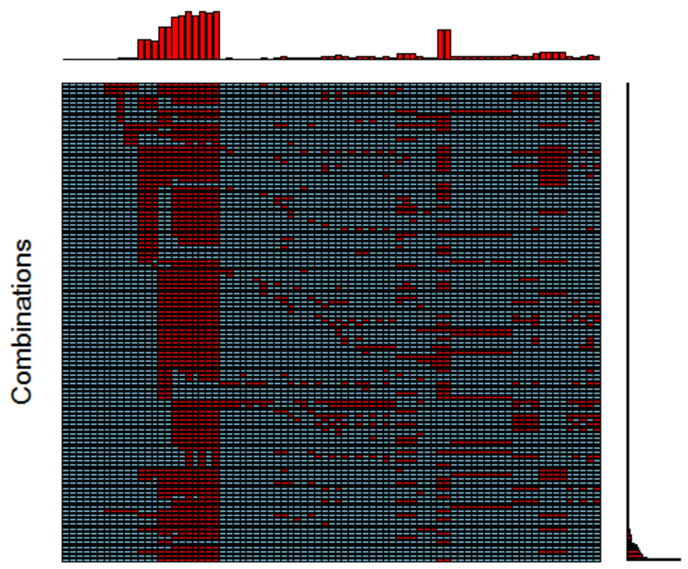


Figure 4.1.: Aggregation plot for CID1

In order to explore these relationships further, I performed Hierarchical Clustering on the shadow matrix R defined in section 4.1.1. The resulting dendrogram shows clearly the categories or clusters of variables that are jointly missing, and gives valuable supplementary information to both the aggreation plot and the missingness map.

For instance, it is very interesting to see how the missing values were divided into two main clusters with the 11 variables having the highest percentage of missing entries located in the right branch of the tree. These 11 variables are related to the *BMR test*. Three other variables that have also a very high percentage of missing values can be seen on the far left branch and are related to the *bioimpedance analysis*.

This graphical representation of "missingnes clusters" will be particularly useful to researchers that do not have a medical background or are unfamiliar with the protocol.

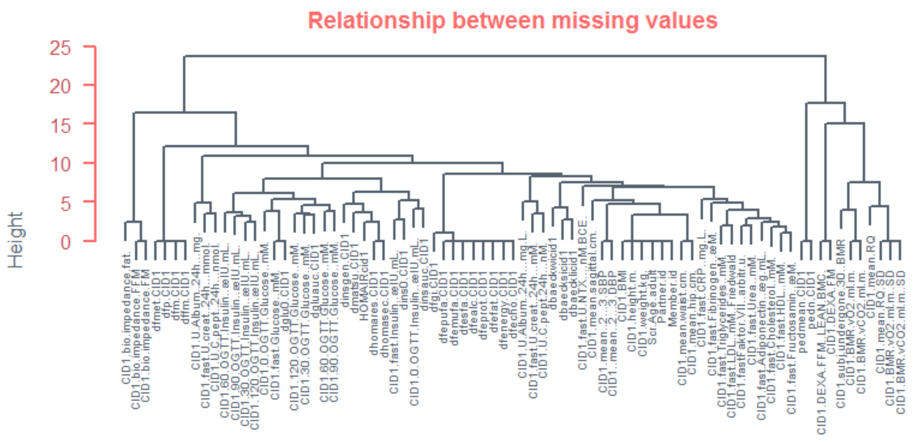


Figure 4.2.: Hierarchical clustering of missing entries

4.1.3. Imputation methods

The choice of imputation method depends greatly on the missingness mechanism of the data. In particular, if the data has a MNAR structure, it is necessary to specify a separate model for the missingness, and then introduce it into a more complex model for the imputation of values.

I explored four different imputation methods for the missing data at CID1. These were:

1. **Mean imputation:** Missing values are replaced by the column mean. One of the downsides of this method is that it is ignores possible relations between variables. Moreover, it reduces the variability of the data and the information that can be drawn from it.

2. ***k*-nearest neighbor imputation:** It is a distance-based, multivariate imputation method. The k-NN imputation searches for the k nearest observations (respective to the case which has to be imputed) and replaces the missing value with either the mean, median, or the most frequent value (for categorical variables) of the found k neighbors.

   I implemented this method with two different functions:

   - `impute.knn` from the package **Bioconductor**, which was designed for the imputation of missing gene expression data. The algorithm only works for numerical data, and selects the k neighbors by an euclidean distance. Missing values are imputed by the mean of its neighbors.

   - `kNN` of **VIM** handles mixed-type data, and chooses the neighbors by a variation of the Gower distance. Numerical missing values are imputed by the k-neighbors median, and categorical variables by the most frequent observation.

3. **Non-linear iterative partial least squares (NIPALS) algorithm:** The algorithm performs a principal component analysis using an iterative procedure on incomplete data, and can provide estimates for the missing cases. I implement this

method with the function `nipals` from **mixOmics**.

4. **`missForest`**: It is a non-parametric, iterative imputation method based on a random forest. The method treats the variable of the missing value as predictor and borrows information from other variables by the resampling-based classification and regression trees to grow a random forest for the final prediction. The method is repeated until the imputed values reach convergence (Liao et al., 2014). This method handles mixed-type data.

4.2. Principal Component Analysis

Principal component analysis (PCA) is a statistical method used to generate a low-dimensional representation of the variation in an original multivariate data set. In simpler words, PCA is a tool for reducing the number of variables of a data set while retaining as much as possible of the underlying information (measured in terms of variability) in the data.

This reduced representation comes in the form of principal components, which are "artificial variables" constructed by a weighted linear combination of the original variables in the data. These components account for maximal variance (or information), while remaining orthogonal to each other. They are obtained by an eigenvalue decomposition of the empirical covariance matrix $\frac{1}{n}X^TX$.

The number of principal components to be kept can be determined through different criteria. One commonly used method is the interpretation of the scree plot of the eigenvalues and consists in looking for an "elbow" in the downward-sloping line or barplot. This point represents a natural break between high and low eigenvalues and thus, the components right above the kink should be kept in the analysis.

Finally, one of the main features of PCA is that it facilitates the graphical representation of the main information contained in the data. This is done by plotting the projections of the variables and individuals on the axes of the principal components. An example of these graphical representations are the individuals' factor map and the variables' factor map (see Figure 4.4 and Figure 4.5, respectively).

In R, I performed PCA using the package **FactoMineR**, which allows for mixed-type data and automatically standardizes the variables. I also used the package **factoextra** for some visualizations of the output of PCA.

4.2.1. PCA on the *clean* data set

The first step in the analysis was to perform a standard PCA on a subset of the original data with no missing values. This data set was constructed by removing the 14 variables that had more than 200 missing values, and then keeping only the observations with complete cases. In the end, the reduced data set contained 298 observations and 65 variables, which corresponds to 48.5 percent of the individuals and 82.3 percent of the original variables at CID1. Henceforth, I will refer to this data set as the *clean* data set.

From looking at the scree plot, I chose to keep the first four principal components, which jointly explain 44.16 percent of the variance in the data.

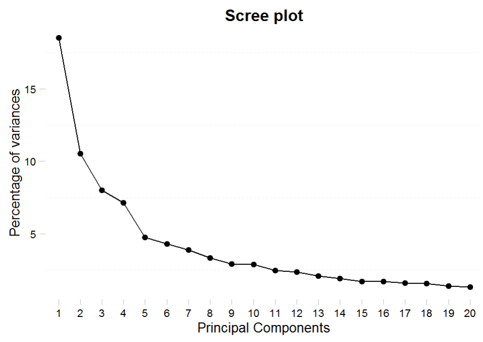


Figure 4.3.: Scree plot of PCA for *clean* data set

For the DiOGenes data at CID1, the reason to perform PCA was to determine which were the variables that provide the most information in the data, and to identify possible outliers. As can be seen in Figure Figure 4.4, there are some possible outliers, coming from different centers. Upon further examination of these individuals and discussion with my supervisors, we came to realize that the outliers were in fact individuals whose resistance to insulin was particularly high. They were not removed from the data since they may bring very valuable information for future analyses.

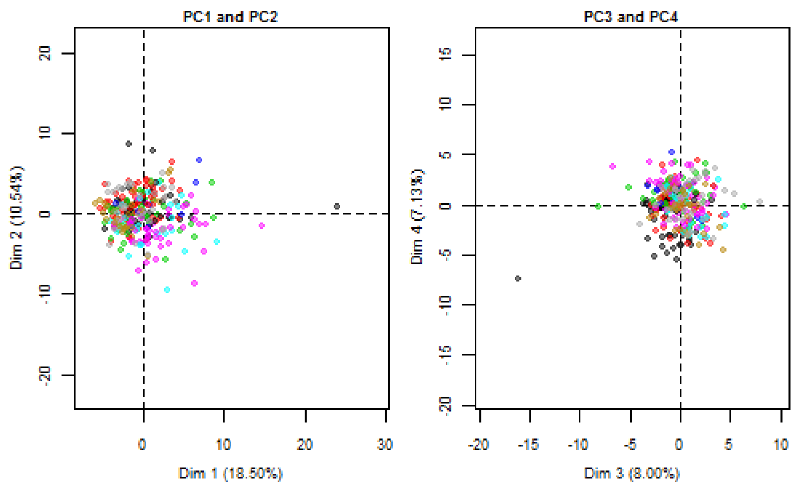


Figure 4.4.: Individuals' factor map for PCA in *clean* data set

4.2.2. PCA for different imputation methods

PCA was then performed on the entire data set at CID1, after imputation of the missing values through different methods. The main objective of this exercise was to see if there were any major changes in the composition of the principal components and the projections of the variables and individuals in these axes. We take the PCA results from the *clean* data set as baseline for the different comparisons.

The imputation methods explored were:

- Mean imputation: It is the default imputation method used by the function `PCA()` of **FactoMineR** when the data matrix contains missing values.
- k-nearest neighbor imputation: performed initially using the package **Bioconductor**.
- NIPALS algorithm: Implemented using the package **mixOmics**.

As can be seen in Figure 4.5, there is a significant change in the projections of the variables on the principal components — this is particularly evident in the case of components three and four. Therefore, it is clear that for the clinical data the method of imputation chosen will have a non negligible impact on the relationships of the variables and on any downstream statistical analysis.

One of the reasons for these drastic changes in the projections of the variables on the principal components comes from the fact that these imputation methods modify the variability of the variables in different magnitudes. For instance, variables with a larger percentage of missing values will have their missing elements replaced by the column mean in the case of mean and k-NN imputation. The remaining variables — specifically those with less than 50% missing cases — will be imputed either by the column mean or by the column mean of its nearest neighbors.

Another important remark to make is that only some of the imputation methods used work with mixed-type data, therefore, the number of variables used to construct the principal components is slightly different in each case.

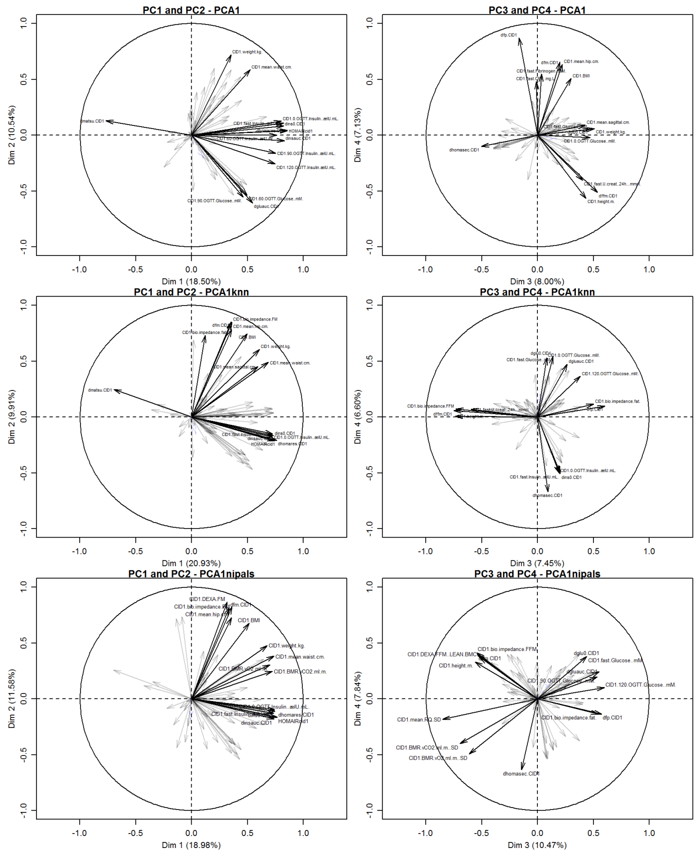


Figure 4.5.: Comparison of variables' factor map

4.3. Comparison of imputation methods

The different imputation methods were compared in two ways. The first was to compare the methods with respect to the variables' projections on the principal components. In order to do this, PCA was performed for various imputed data sets of CID1.

The second, more traditional approach, was to calculate the imputation error for different methods. In order to do this, I recreated the missingness structure of the data at CID1 in the *clean* data set, and then imputed these with through different methods, and compared the error.

The methods studied were:

- mean imputation
- NIPALS algorithm
- `missForest`
- k-NN imputation
  - `impute.knn` of **Bioconductor**
  - `kNN` of **VIM**

4.3.1. Distance between principal components

The main conclusion that could be drawn from the preliminary PCA analysis is that the choice of imputation method has an important effect on the relationships between variables and the information that they provide. However, it is impossible to determine which imputation methods yield the most similar interpretation in the principal components merely by looking at the variables' factor map.

One way of comparing these different methods is by calculating the two-by-two euclidean distances between the principal components for the different imputation methods. These distances are then used to construct a (symmetric) distance matrix, with which we can perform Hierarchical Clustering and plot the resulting dendogram.

This consists in calculating the pairwise distances between the variables' coordinates on the principal components for two different methods. I will demonstrate with a simple example.

Let us define two matrices, a^k and $a^{k'}$, that contain the variables' coordinates on p principal components (in columns), for n variables (in rows) after a PCA performed on the imputed (or the *clean* data set). The euclidean distance between these matrices is defined as:

$$d_{k,k'} := d(a^k, a^{k'}) = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{p} \left(a_{ij}^{k} - a_{ij}^{k'}\right)^2} \tag{4.1}$$

where

$$a^k = \begin{pmatrix} a_{11}^k & \cdots & a_{1p}^k \\ \vdots & \ddots & \vdots \\ a_{n1}^k & \cdots & a_{np}^k \end{pmatrix}$$

for $k = 1, ..., K$ different imputation methods.

The resulting distances can thus be used to construct a $K \times K$ euclidean distance matrix $\mathbf{M}$, where the rows and columns represent the different imputation methods.

$$\mathbf{M} = \begin{pmatrix} 0 & d_{1,2} & \cdots & d_{1,K} \\ & 0 & \cdots & d_{2,K} \\ & & \ddots & \vdots \\ & & & 0 \end{pmatrix}$$

To have a clearer visualization of the difference between methods, I performed hierarchical clustering on this matrix, and plotted the resulted dendogram.

Figure 4.6 gave us a better understanding of the imputation methods used. One particularly valuable thing that we learned is that imputation by k nearest neighbors yields very different results depending on the package choice. This is because the function `impute.knn` from the package **Bioconductor**, originally designed for microarray data, selects the neighbors by looking at the similarities between the variables and not the individuals. This method thus implicitly supposes that the some subsets of variables have a similar distribution.

It is also interesting to note that imputation by the mean, which is the least sophisticated method that we explored, is very similar to the non-parametric method `missForest` and k-NN imputation of the package **VIM**.

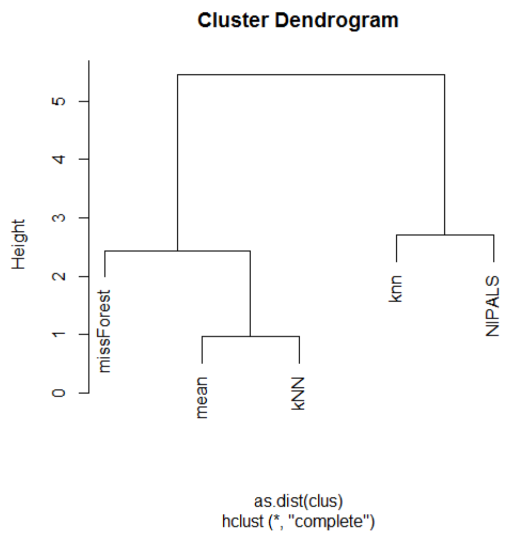


Figure 4.6.: Hierarchical clustering for imputation methods

4.3.2. Imputation error

The final step of the analysis was to compare the accuracy of the different methods by calculating the normalized root mean squared error (NRMSE) of the imputed data sets. In order to do this, I tried to recreate the missingness pattern that I observed in the data at CID1. This meant “randomly” inserting artificial missing values to the *clean* data set. However, the process is not entirely random, since I make some restrictions on the missingness of certain variables.

The artificial missing entries were introduced in two steps:

1. *Combinations of missing cases:* I identified the most common combinations of missing values with the use of the aggregation plot of **VIM**, and calculated their frequencies. I then recreated these combinations in the *clean* data set.

2. *Proportion of missing by column:* I recreated some of the MCAR structure of the data by introducing supplementary missing cases by column, respecting the proportion of missing values per variable.

To better understand the structure of the data at CID1, I first performed a visual analysis of the missing values of the data without the 14 variables with the highest rate of missingness. Next, I tried to reproduce the observed patterns on the *clean* data set, for which all cases had complete information. Figure 4.7 shows this result.

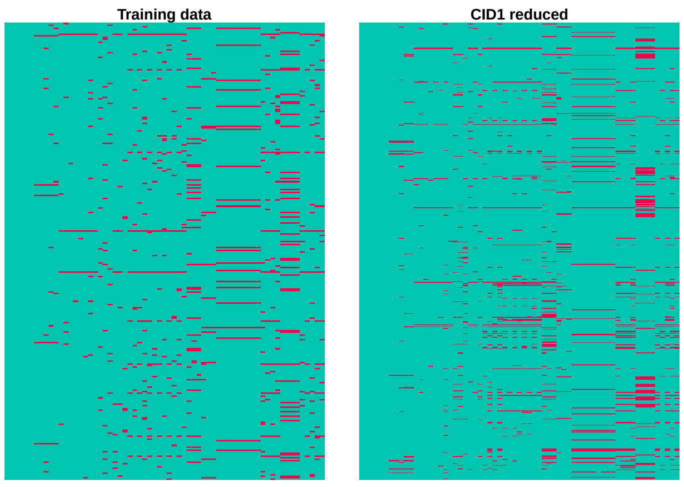


Figure 4.7.: Comparison of the training data and CID1 reduced

In order to compare all of the imputation methods, I generated several training data sets with a similar missingness structure, and imputed the missing values using four methods: `kNN`, `missForest`, NIPALS and mean imputation.

Finally, the imputation error was calculated with the function `mixError` of **missForest**, which is defined as:

$$NRMSE = \frac{\sqrt{mean\left[(y_{imp} - y_{true})^2\right]}}{variance(y_{true})}$$

where the mean and the variance are calculated over missing entries in the whole matrix (Oba et al., 2003). We know y_{true} because the missing entries are artificial.

As can be seen in Figure 4.8, the method that yields the smallest imputation error is that of `missForest`. Surprisingly, the second best imputation method is simple mean imputation, outperforming the commonly used k-NN method. NIPALS imputation has very highly variability: for one of the iterations, it gave an incredibly high NRMSE (which does not appear on the box plot).

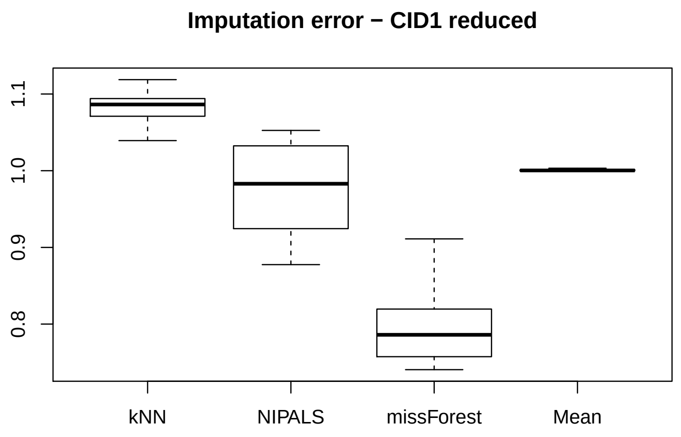


Figure 4.8.: Imputation error for the different methods

4.4. Conclusion and possible extensions

There are some interesting results that came from this preliminary analysis of imputation methods. First, the imputation method that best fits the data is the non-parametric, multiple imputation algorithm implemented by the function `missForest`, outperforming commonly used methods as k-nearest neighbor, or imputation based on principal components (NIPALS). Moreover, the choice of imputation method can lead to major changes in the underlying relationships between the data, as evidenced in the imputation analysis through PCA.

The method that appears to be the least reliable is imputation through NIPALS algorithm. On average, NIPALS is more accurate than k-NN imputation, but it can produce very large errors for some imputations, making it thus less reliable. Another fact that is very disappointing is that k-NN imputation appears to be less accurate than simple replacement by the mean. The next step would be to try an implement the method by changing the criteria that is used to choose the neighbors.

Finally, the good results shown by `missForest` suggest that it may be worthwhile to further explore iterative imputation methods.

Bibliography

Liao, S. G., Lin, Y., Kang, D. D., Chandra, D., Bon, J., Kaminski, N., Sciurba, F. C., and Tseng, G. C. (2014). Missing value imputation in high-dimensional phenomic data: imputable or not, and how? *BMC Bioinformatics*, 15.

Little, R. J. A. and Rubin, D. B. (2002). *Statistical Analysis with Missing Data*. Wiley, second edition.

Oba, S., Sato, M.-a., Takemasa, I., Monden, M., Matsubara, K.-i., and Ishii, S. (2003). A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096.

Stekhoven, D. J. (2013). *missForest: Nonparametric Missing Value Imputation using Random Forest*. R package version 1.4.

Templ, M., Kowarik, A., and Filzmoser, P. (2011). Iterative stepwise regression imputation using standard and robust methods. *Computational Statistics & Data Analysis*, 55(10):2793 – 2806.

Todorov, V. (2012). R in the statistical office. Development policy, statistics and research branch working paper, United Nations Industrial Development Organization.

Viguerie, N., Montastier, E., Maoret, J.-J., Roussel, B., Combes, M., Valle, C., Villa-Vialaneix, N., Iacovoni, J. S., Martinez, J. A., Holst, C., Astrup, A., Vidal, H., Clément, K., Hager, J., Saris, W. H. M., and Langin, D. (2012). Determinants of human adipose tissue gene expression: Impact of diet, sex, metabolic status, and cis genetic regulation. *PLoS Genet*, 8(9):e1002959.

A. R Markdown scripts

The following are extracts of the reports I created using R Markdown. The complete `.Rmd`, `.pdf` and `.html` versions are available upon request.

A.1. clinicAnalysis

Extract of the file `clinicAnalysis`.

A.2. Imputation

Extract of the file `Imputation`.

Clinic data analysis

Maria Paula Caldas

Monday, April 6, 2015

Last update of this file is: **07 juin 2015**. This file must be compiled with the following packages:

- FactoMineR
- knitr
- impute (of Bioconductor)
- mixOmics
- ggplot2
- ggthemes
- reshape
- gridExtra
- factoextra (Github)
- RColorBrewer

Preliminary steps

Clinical data can be imported using:

```
clinic <- read.csv("../data/export_diogenes.csv", sep=";", dec=",",
                   fileEncoding="latin1")
```

The data contain the measure of 444 variables for 614 individuals.

Tidying the data

This section shows the preliminary steps taken in order to tidy the data, namely:

- Renaming of variables.
- Encoding categorical variables as factors.
- Encoding numerical variables as numeric.

Code available in complete version

The data will also be separated into different data sets for each time step: CID1, CID2 and CID3. Center, age and sex will be included in each one of the data sets. The resulting data sets have the following characteristics:

	CID1	CID2	CID3
Number of observations	614	614	614
Number of variables	79	67	67

Missing values in the data

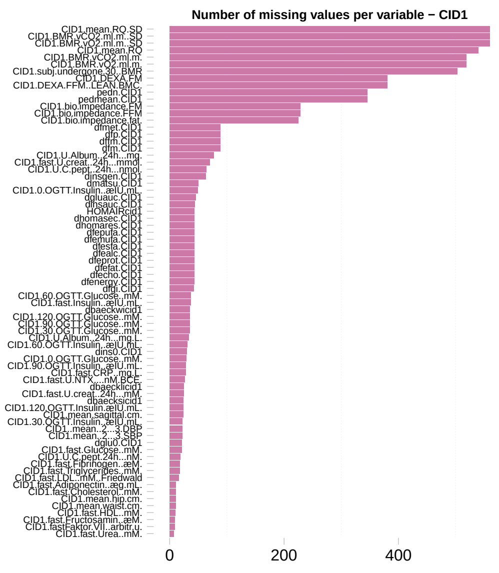
This section will check:

- the number of missing values per variable and per individual;
- global and local analysis of the missing data.

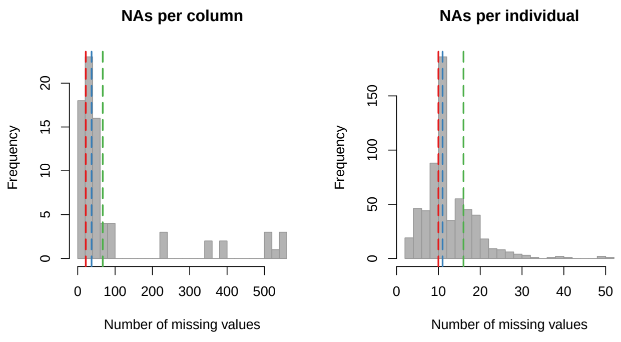
One important remark about the clinical data is that there is no individual for which we have complete information for all variables. The overall number of missing values also increases at each time step: it is 16.56% at CID1, 24.47% at CID2, and 40.42% at CID3.

Exploratory analysis of variabels in CID1

The largest number of missing values at CID1 is comes from the variables measured in the Basal Metabolic Rate (BMR) test.



The figure below shows the distribution of missing values per variable and per individual, where the dashed lines represent the quartiles of each distribution.



Reduced data set with no missing values

The first step in the analysis will be to perform a standard PCA on a subset of the original data that has no missing values. This data set is constructed by removing the 14 variables that have more than 200 missing values. These are:

```
##  [1] "CID1.bio.impedance.FFM"      "CID1.bio.impedance.FM"
##  [3] "CID1.bio.impedance.fat."     "CID1.DEXA.FFM..LEAN.BMC."
##  [5] "CID1.DEXA.FM"                "CID1.subj.undergone.30..BMR"
##  [7] "CID1.BMR.vO2.ml.m."          "CID1.BMR.vO2.ml.m..SD"
##  [9] "CID1.BMR.vCO2.ml.m."         "CID1.BMR.vCO2.ml.m..SD"
## [11] "CID1.mean.RQ"                "CID1.mean.RQ.SD"
## [13] "pedmean.CID1"                "pedn.CID1"
```

After removing these columns, only the observations that have no missing values will be kept.

```
CID1noNA <- CID1[, which(as.numeric(naByCol) < 200)]
CID1noNA <- subset(CID1noNA,complete.cases(CID1noNA)==T)
```

By doing this, we further restrict our analysis to a total of 298 observations, which correspond to 48.5 percent of the individuals.

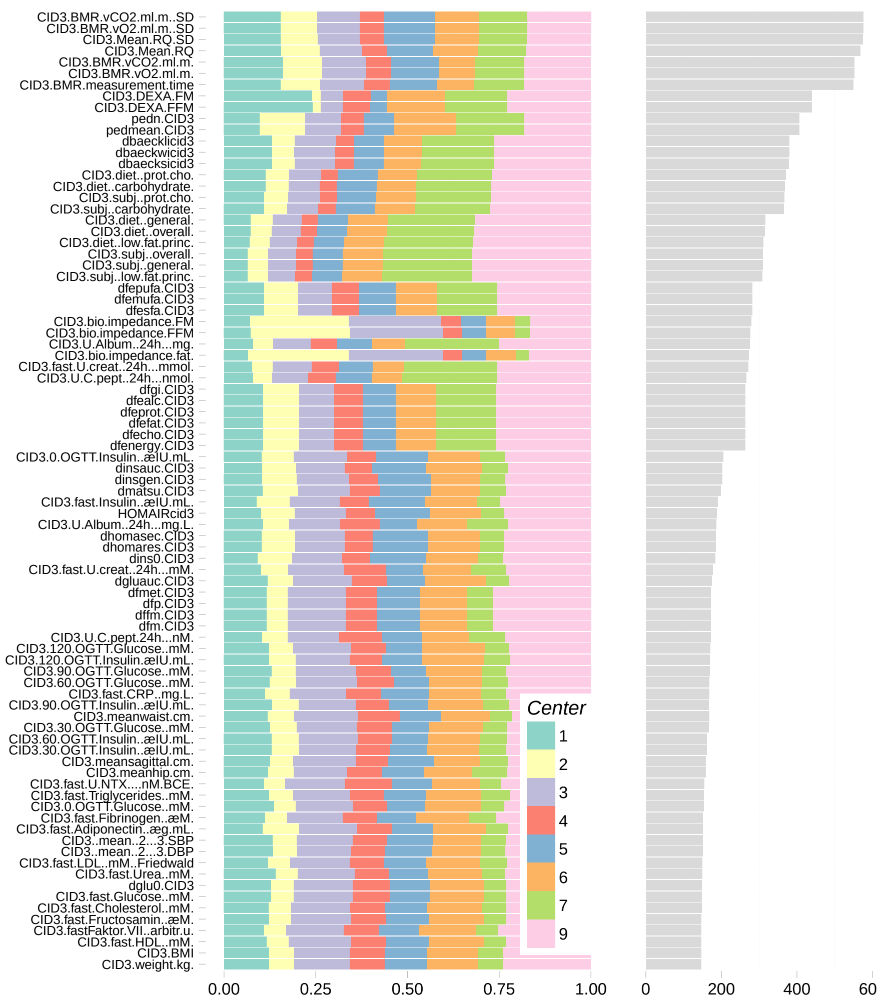
Exploratory analysis of variabels in CID2

Not in extract. Available in complete version

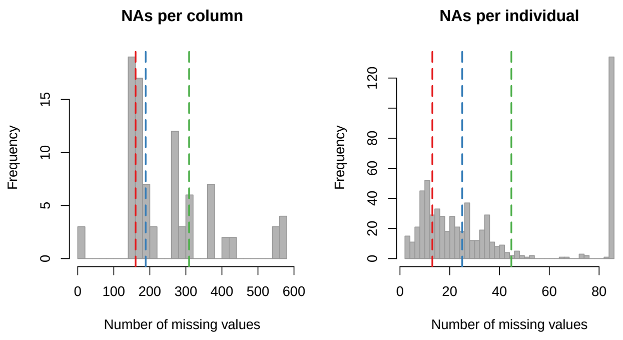
Exploratory analysis of variables in CID3

The following figure gives a more detailed example of the composition of missing values at CID3. To the right is an inverted bar plot showing the number of missing values per variable, and to the left, a spineplot showing the proportion of missing values per center. The variables are ordered, going from the one that has the most missing values (top), to the one that has the least (bottom).

First, we should note that there is no variable at CID3 that has less than 100 missing values per variable. Moreover, it is interesting to see that the proportion of missing values remains more or less constant for all centers. Therefore, we can rule out the possibility that the large number of N.A for one particular variable was due to an error in measurement from one particular center.



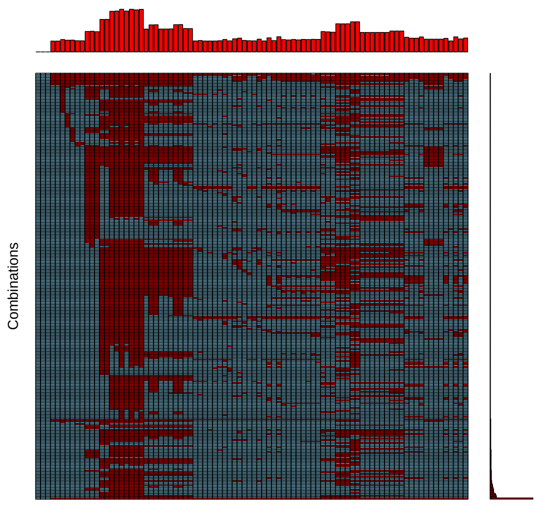
It is important to note that there are now more than 120 individuals that have more than 80 missing values at CID3 (90.91% of the variables at this time step).



There are 134 individuals for which we have no information at CID3.

Example using package VIM: The following plot (called aggregation plot) presents a quick overview of the missing values in the complete data set. It shows for the same variables (horizontal axis) all different combinations that are present in the observations with missing and non-missing values (vertical axis). The missing values are presented in red and the complete cases in blue.

The columns have the same order as the original data set. The bar plot on the right represents the frequencies of observations to the corresponding combinations.



Reduced data set with no missing values

Not in extract. Available in complete version

Pricipal Component Analysis on CID1

A PCA is performed in order to determine which are the variables that provide the most information in the data, and to analyse the behavior of outliers. It is performed first on a reduced version of the data set containing no missing values, and on the complete data set where the values have been imputed using different imputation techniques. These are:

- Imputation by the mean (default treatment of missing values in package **FactoMineR**);
- k-nearest neighbor imputation using the function `impute.knn` of the **Bioconductor** package;
- Imputation using the NIPALS algorithm, implemented using the **mixOmics** package.

Analysis of the clinical data at CID1

Standard PCA

PCA is performed using the following command lines:

```
PCA1 <- PCA(CID1noNA, quali.sup=c(1:2,60), quanti.sup=3, graph=F)
```

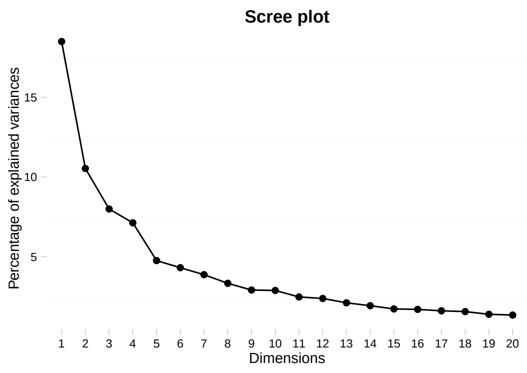
For the selection of the principal components to keep, there are different criteria that can be used, namely:

- Variance explained criterion: Determining a minimum threshold for the explained cumulative percentage of variance (generally 70% or 80%). We would need to keep 12 components in our analysis if we were to explain, for instance, at least 70% of the variance in the model.
- Kaiser criterion: Consists in keeping the all of the principal components that have eigenvalues greater than 1, which corresponds to 17 components for the current data.
- Catell Scree Test: Consists in taking the second differences in the eigenvalues until the sign changes. As we can observe in the table below, this method points to choosing only the first two components.

```
diff(diff(PCA1$eig[,1]))[1:10]
```

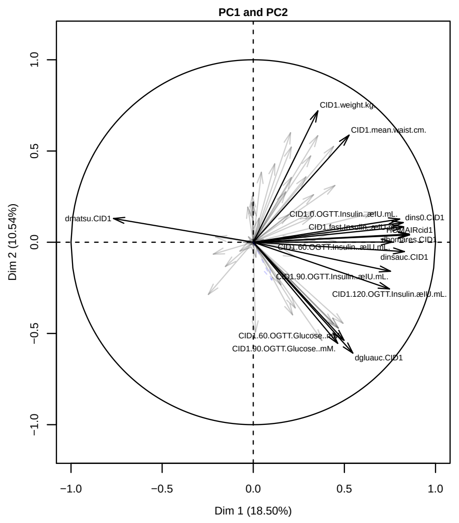
```
##  [1]  3.305896661  1.022774667 -0.917631401  1.173416127  0.005990917
##  [6] -0.064583850  0.075119940  0.238299678 -0.230310493  0.189641047
```

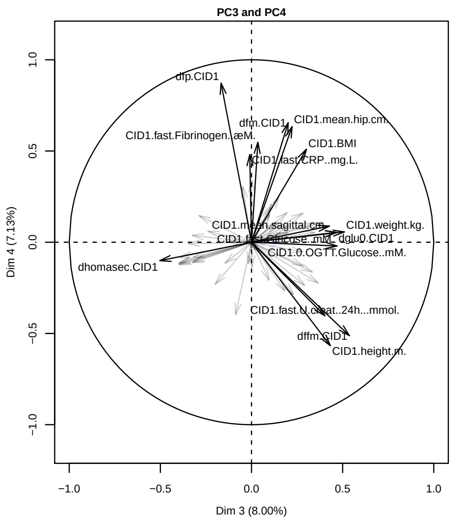
- Scree plot: We select the number of components strictly below a transition point (or "elbow"). In this case, the scree plot has two "elbows", one at the third component and another at the fifth component. This suggests a natural break between high and low eigenvalues for the second or the fourth component.



The first two criteria suggest using a large number of components in order to capture the most information from the original data set; however, they have the disadvantage of making the interpretation more difficult. The goal of using PCA at this stage is to identify the most relevant variables, and thus, keeping a large number of components would be counterproductive. Therefore, we will analyse only the first four principal components, which account for 44.16% of the variance in our data.

Variables' Factor Map The variables' factor map shows the relationship between the variables and in the space of the principal components. Given the large number of variables, only the 15 elements that have the highest contribution on the two dimensions of the plot are drawn. The supplementary categorical variable (age), does not contribute much to the construction of the first four components, which is why it does not appear in the figure.

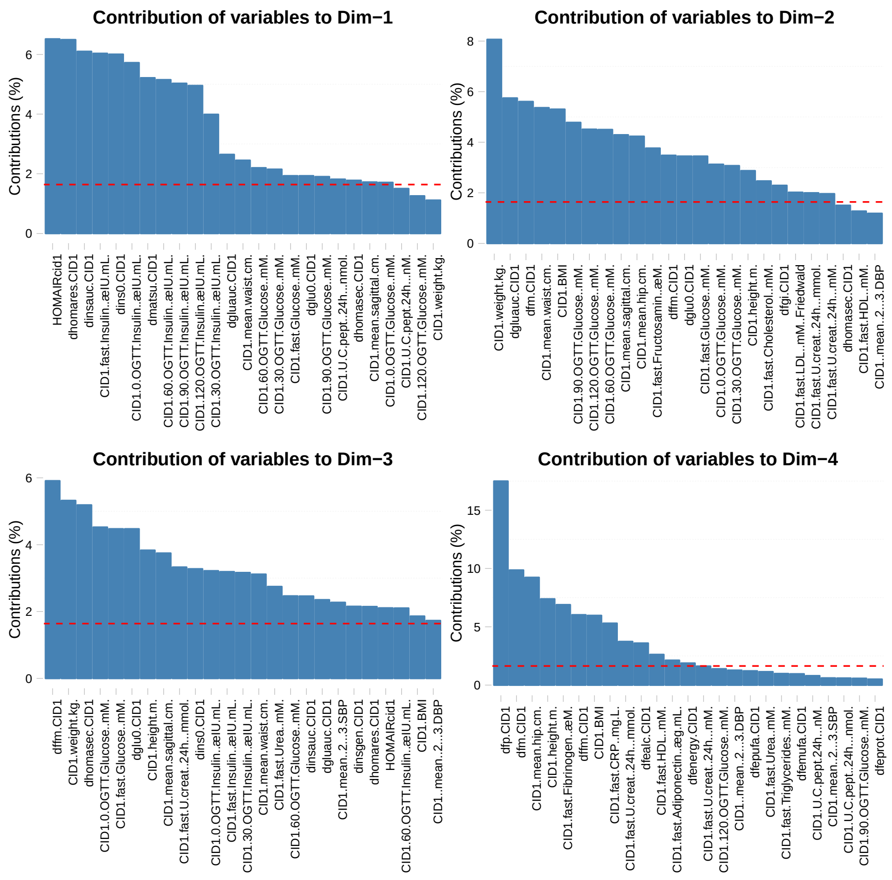




The variables that contribute most to the construction of the **first principal component** are those related to the insulin response of patients. These variables include the insulin concentration measures during the oral glucose tolerance test (OGTT), the HOMA-IR and *dhomares* measures, *dinsauc.CID1* and *dins0.CID1*. Moreover, the variable *dmatsu.CID1* appears to be negatively correlated with the insulin variables. The **second principal component** is highly correlated with variables related to the weight and waist measurement of the patients at CID1, *dgluauc.CID1* and *CID1.90.OGTT.Glucose..mM.*.

For the **third** and **fourth component**, we can see that *dhomasec.CID1* is negatively correlated with *dglu0.CID1*, *CID1.fast.Glucose..mM.* and the patient's weight.

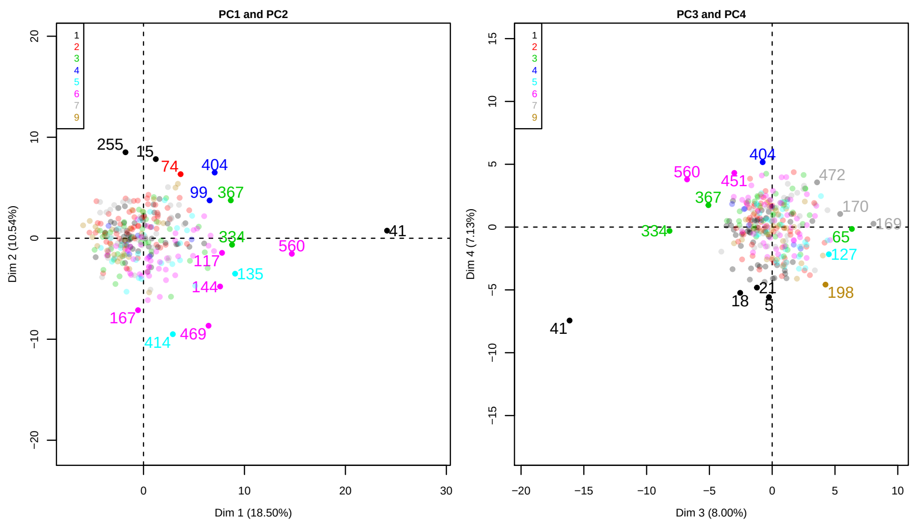
The following bar plots show the variables that contribute most (top 25) in the composition of the first two components. The red dashed line is located at the point 1/61, and it indicates the expected average contribution of each variable (if the contribution of variables were uniform).



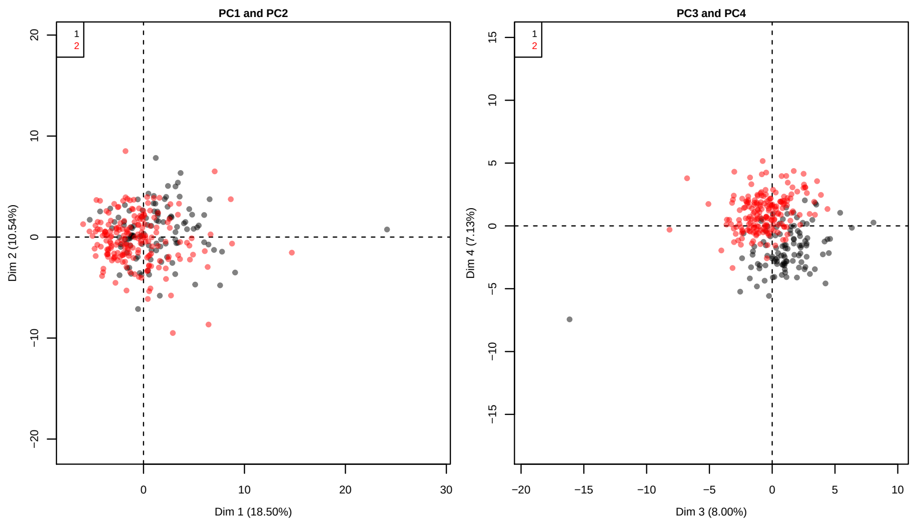
Individuals' Factor Map For the interpretation of the individuals' factor map, it is possible to plot each individual with respect to the center or his/her sex.

Center: The following two figures show the 10 individuals that contribute the most to the construction of each dimension. In particular, we can observe that individuals 41, 334, 367 and 560 contributed highly to the construction of the four principal components. Moreover, we have that individual 41, in particular, seems to be an outlier for all dimensions.

Upon further exploration of these individuals, we remark that they are all insulin resistant.



Sex: There is a clear differentiation between men and women in the principal components axes, particularly for the third and forth component.



PCA: Imputation by the mean

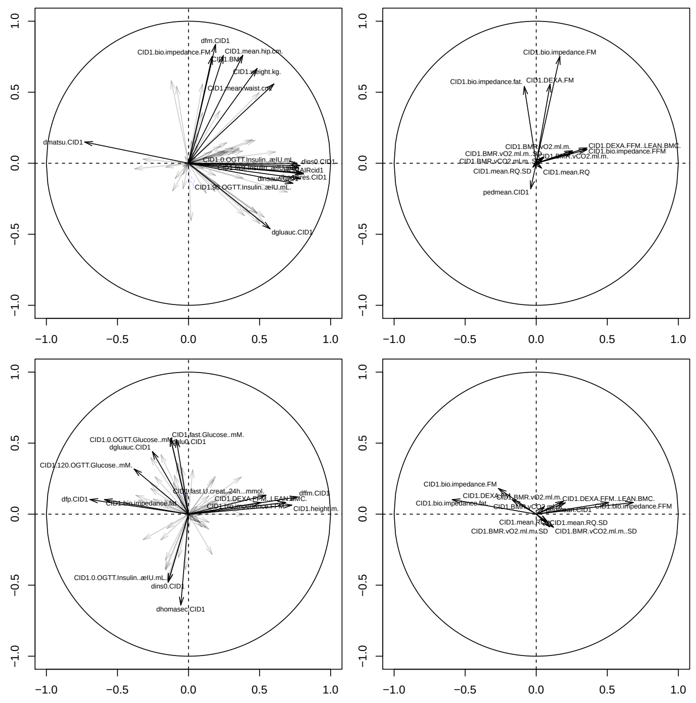
The function PCA in FactoMineR can also perform an analysis on data with missing values by imputing all the missing values by each variable's mean.

```r
PCA1mean <- PCA(CID1, quali.sup=c(1:2,17,57,74), quanti.sup=3, graph=F)
```

Once again, the scree plot suggests keeping only the first four principal components.

Not in extract. Available in complete version

Variables' factor map Comparing the variables' factor map for PC1 and PC2, the interpretation of these components remains the same. The first component seems to separate the individuals that are insulin-resistant from those that are not, while the second component is constructed mainly from those variables related to physical attributes (BMI, mean waist, etc).



In the figure above, to the right are the variables that were omitted in the first PCA, and, as before, to the left are the variables that contribute the most to the construction of the first two components. In particular, the newly included variables *CID1.bio.impedance.fat*, *CID1.bio.impedance.FM*, *CID1.bio.impedance.FFM* and *CID1.DEXA.FFM..LEAN.BMC.* are now contributing heavily in the construction of the axes.

Individuals' factor map *Not in extract. Available in complete version*

PCA: *k*-NN imputation

The function `impute.knn` imputes missing data of an expression matrix, where genes (variables) are located in rows and samples (observations) in columns. Therefore, we must work with a transposed version of our data of CID1.

In the case that an individual has more than 80% missing values, then the program halts and reports an error; which is not the case with the clinical data. If any given variable has more than 50% missing entries, then the missing values are replaced with the variable's mean. This is the case for:

```
##  [1] "CID1.DEXA.FFM..LEAN.BMC."    "CID1.DEXA.FM"
##  [3] "CID1.subj.undergone.30..BMR" "CID1.BMR.vO2.ml.m."
##  [5] "CID1.BMR.vO2.ml.m..SD"       "CID1.BMR.vCO2.ml.m."
##  [7] "CID1.BMR.vCO2.ml.m..SD"      "CID1.mean.RQ"
##  [9] "CID1.mean.RQ.SD"             "pedmean.CID1"
## [11] "pedn.CID1"
```

> Important remark: The *k*-NN algorithm requires the data to be normalized. Therefore, the data will be scaled to unit variance and zero mean.

```r
# Scaling only the numerical variables
CID1sc.matrix <- scale(as.matrix(CID1[,-col.fact]))

# Imputation
CID1knn.sc <- impute.knn(t(CID1sc.matrix), k=5)$data

# Unscale the numeric variables
CID1knn <- unscale(as.matrix(t(CID1knn.sc)),CID1sc.matrix)
CID1knn <- as.data.frame(CID1knn, row.names=1:nrow(CID1))

# Including back Member.Id and Partner.Id
CID1knn$Partner.id <- CID1$Partner.id
CID1knn$Member.id <- CID1$Member.id
```

PCA for the imputed data:

```r
# PCA
PCA1knn <- PCA(CID1knn, quali.sup=75:76, quanti.sup =1, graph=F)
```

The scree plot has a kink at the fifth component, therefore the analysis will be conducted keeping only the first four components, which account to 44.88% of the information in the data.

Figure not available in complete version

Variables' factor map *Not in extract. Available in complete version*

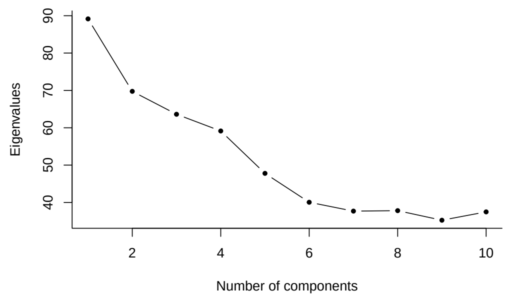
Individuals' factor map *Not in extract. Available in complete version*

PCA: NIPALS algorithm

The missing values are imputed using the NIPALS algorithm, which handles only numerical variables. Therefore, 5 factor variables will be excluded from the data set. The data also needs to be scaled to unit variance before running the nipals algorithm.

The reconstitution of the missing values is done using a determined number of principal components, chosen by analysing the scree plot.

```r
# NIPALS to choose the number of components
nipals.tune <- nipals(CID1sc.matrix, reconst = TRUE,
                      ncomp=10, max.iter = 1000)$eig

plot(1:10,nipals.tune[1:10],type="b", pch=20, bty = "l",
     ylab="Eigenvalues", xlab="Number of components")
```



There is a break at component 6, therefore, the missing values will be imputed using 5 components.

```r
# NIPALS with the chosen number of components
nipals.X <- nipals(CID1sc.matrix, reconst = TRUE, ncomp = 5)$rec

# Only replace the imputation for the missing values
id.na <- is.na(CID1sc.matrix)
nipals.X[!id.na] <- CID1sc.matrix[!id.na]

# Unscaling the variables
nipals.X <- unscale(as.matrix(nipals.X), CID1sc.matrix)

# Data with imputed missing values
CID1nipals <- as.data.frame(nipals.X, row.names=1:nrow(CID1))
CID1nipals$Partner.id <- CID1$Partner.id
CID1nipals$Member.id <- CID1$Member.id
```

Performing PCA on the data with imputed values we get once again that we should keep only the first four principal components.

```r
PCA1nipals <- PCA(CID1nipals, quali.sup=75:76, quanti.sup=1, graph=F)
```

Variables factor map *Not in extract. Available in complete version*

Individuals factor map *Not in extract. Available in complete version*

Comparison of the PCA results

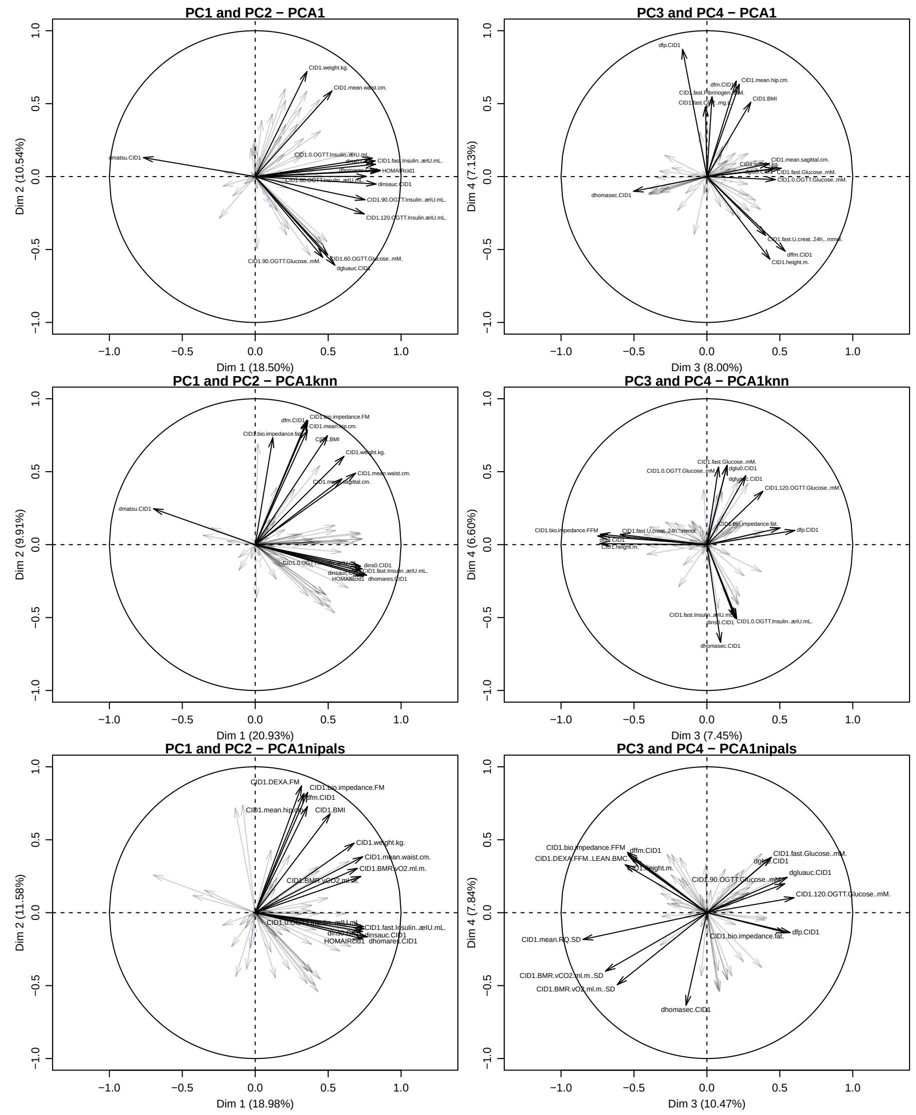
The following table summarizes the main features of the analysis.

Table 2: Summary of PCA results for CID1

	PCA1	PCA1mean	PCA1knn	PCA1nipals
Observations	298.00	614.00	614.00	614.00
Variables	65.00	79.00	76.00	76.00
% imputed values	0.00	16.56	16.56	15.20
Num. of components selected	4.00	4.00	4.00	4.00
Cum. % of variance explained	44.16	38.56	44.88	48.87

Imputation by the mean seems to be the least effective method to use for this data, as it reduces the information (variance), given by the principal components.

As mentioned before, the interpretation of the first and second principal component remains the same regardless of the imputation method used. However, the biggest changes are observed in the third and fourth principal component. The figure below shows the variables' factor map for the three methods explored: mean imputation, *k*-NN, and NIPALS algorithm.



Session info

```
## R version 3.2.0 (2015-04-16)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## locale:
## [1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252
## [3] LC_MONETARY=French_France.1252 LC_NUMERIC=C
## [5] LC_TIME=French_France.1252
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] RColorBrewer_1.1-2  factoextra_1.0.2    gridExtra_0.9.1
##  [4] reshape_0.8.5       ggthemes_2.1.2      ggplot2_1.0.1
##  [7] VIM_4.1.0           data.table_1.9.4    colorspace_1.2-6
## [10] Amelia_1.7.3        Rcpp_0.11.6         missForest_1.4
## [13] itertools_0.1-3     iterators_1.0.7     foreach_1.4.2
## [16] randomForest_4.6-10 mixOmics_5.0-4      MASS_7.3-40
## [19] impute_1.42.0       FactoMineR_1.29     DMwR_0.4.1
## [22] lattice_0.20-31     knitr_1.10.5
##
## loaded via a namespace (and not attached):
##  [1] vcd_1.3-2           rgl_0.95.1247       class_7.3-12
##  [4] zoo_1.7-12          gtools_3.5.0        digest_0.6.8
##  [7] plyr_1.8.2          chron_2.3-45        evaluate_0.7
## [10] e1071_1.6-4         highr_0.5           gplots_2.17.0
## [13] minqa_1.2.4         gdata_2.16.1        SparseM_1.6
## [16] car_2.0-25          TTR_0.22-0          nloptr_1.0.4
## [19] rpart_4.1-9         Matrix_1.2-1        rmarkdown_0.6.1
## [22] labeling_0.3        proto_0.3-10        splines_3.2.0
## [25] lme4_1.1-7          stringr_1.0.0       foreign_0.8-63
## [28] igraph_0.7.1        pheatmap_1.0.2      munsell_0.4.2
## [31] RGCCA_2.0           mgcv_1.8-6          htmltools_0.2.6
## [34] nnet_7.3-9          flashClust_1.01-2   codetools_0.2-11
## [37] bitops_1.0-6        leaps_2.9           nlme_3.1-120
## [40] gtable_0.1.2        magrittr_1.5        formatR_1.2
## [43] scales_0.2.4        KernSmooth_2.23-14  quantmod_0.4-4
## [46] stringi_0.4-1       ROCR_1.0-7          reshape2_1.4.1
## [49] sp_1.1-0            scatterplot3d_0.3-35 robustbase_0.92-3
## [52] xts_0.9-7           tools_3.2.0         DEoptimR_1.0-2
## [55] abind_1.4-3         parallel_3.2.0      pbkrtest_0.4-2
## [58] yaml_2.1.13         cluster_2.0.1       caTools_1.17.1
## [61] quantreg_5.11
```

“‘

Imputation at CID1

Maria Paula Caldas

Wednesday, April 29, 2015

Data&Packages

Last update of this file is: **07 juin 2015**. Data importation and loading of packages is the same as `clinicAnalysis.Rmd` file.

Imputation at CID1

Fist, the variables will be centered and scaled to unit variance.

Imputation methods

The purpose of this section is to compare different imputation methods by performing hierarchical clustering on the distances of the principal components.

The steps to be taken are:

1. Perform PCA on each of the imputed data sets.

- Determine a number of principal components to keep. This number must be the same for all different methods.
- Extract the coordinates of the active variables on the principal components. The **active variables** must be the same for all the different imputation methods.

2. Calculate the two-by-two euclidean distance between the principal components for the different imputation methods. This means, for example, calculating the distance between PC1 under missForest imputation, and PC1 under k-NN imputation. This is done for the first four components. The same procedure is carried out for each different pair of methods. Therefore, the 4 imputation methods explored will lead to 6 entries in an euclidean distance matrix.
3. Perform Hierarchical Clustering on the resulting distance vectors.

Heterogeneous data

There are some imputation methods that are designed to handle heterogeneous data; that is, both categorical and numerical variables.

`missForest` Imputation is performed with the parameters of `missForest` set to default.

```r
set.seed(19920403)

# Data frame of scaled variables - includes missing values
CID1scaled <- cbind(as.data.frame(CID1sc.matrix), CID1[col.fact])
CID1scaled <- CID1scaled[,colnames(CID1)]  #So variables have the same order

# Imputation
CID1.missForest.sc.df <- missForest(CID1scaled, variablewise = TRUE)$ximp

# Unscaling the numerical variables - imputed data
CID1.missForest.sc.matrix <-
  unscale(as.matrix(CID1.missForest.sc.df[,-col.fact]),CID1sc.matrix)

# Re-inserting the factor variables.
CID1.missForest.scaled <- as.data.frame(CID1.missForest.sc.matrix)
CID1.missForest.scaled <- cbind(CID1.missForest.scaled,
                                CID1.missForest.sc.df[,col.fact])
CID1.missForest.scaled <- CID1.missForest.scaled[,colnames(CID1)]

# PCA
PCA1.missForest <- PCA(CID1.missForest.scaled, quali.sup=col.fact,
                      quanti.sup =3, graph=F)
```

kNN (package VIM) The function kNN has the default numFun = median, which means that the median is used to aggregate the *k*-nearest neighbors in case of a numerical variable. The default for the categorical variable is to take the level with the most occurrences and random if the maximum is not unique, i.e catFun=maxCat.

```r
# Complete data
CID1.kNN.sc.df <- VIM::kNN(CID1scaled, k=5, imp_var=FALSE)

# Unscale the numeric variables
CID1.kNN.sc.matrix <- unscale(as.matrix(CID1.kNN.sc.df[,-col.fact]),CID1sc.matrix)

# Re-insert the factor variables.
CID1.kNN <- as.data.frame(CID1.kNN.sc.matrix)
CID1.kNN <- cbind(CID1.kNN, CID1.kNN.sc.df[,col.fact])
CID1.kNN <- CID1.kNN[,colnames(CID1)]

# PCA
PCA1.kNN <- PCA(CID1.kNN, quali.sup=col.fact,
                      quanti.sup = 3, graph=F)
```

Homogeneous data

There are other imputation methods that work only on numerical data, these will be used excluding the categorical variables at CID1.

Mean This method of imputation is applied by default with the function PCA of the package FactoMineR.

NIPALS algorithm Imputation using the NIPALS algorithm.

Code available in complete version. Same as that of `clinicAnalysis.Rmd`

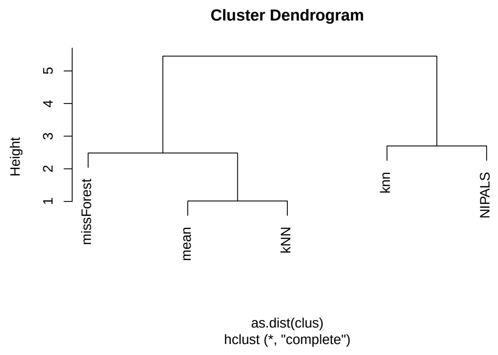
***k*-NN (package Bioconductor)** The function `impute.knn` from the package Bioconductor was created with the objective of imputing missing gene expression data, using nearest neighbor averaging. The default for the function is `rowmax = 0.5`, `colmax = 0.8` and `k=10`. In order to make the results comparable with those of the **VIM** package *k*-NN imputation, the number of neighbors used will be set to `k=5`.

> **Remark:** This function was designed to impute missing gene expression data from an expression matrix with genes in the rows and samples in the columns, i.e variables in the rows, and observations in columns. Therefore, the our data matrix needs to be transposed before running the `impute.knn` function.

Code available in complete version. Same as that of `clinicAnalysis.Rmd`

Comparison of the different methods

```r
# Variables that are common for all PCA analysis.
commonVar <- Reduce(intersect, list(
  rownames(PCA1mean$var$coord),
  rownames(PCA1.missForest$var$coord),
  rownames(PCA1.kNN$var$coord),
  rownames(PCA1knn$var$coord),
  rownames(PCA1nipals$var$coord)
  ))

# List with all the coordinates for each method
listClus <- list(
  PCA1mean$var$coord[commonVar,1:4],
  PCA1.missForest$var$coord[commonVar,1:4],
  PCA1.kNN$var$coord[commonVar,1:4],
  PCA1knn$var$coord[commonVar,1:4],
  PCA1nipals$var$coord[commonVar,1:4])

names(listClus) <- c("mean", "missForest", "kNN", "knn", "NIPALS")

# This function calculates the euclidean distance between two methods
distMethods <- function(x,y){sqrt(sum((x-y)**2))}

# Initializing the distance matrix
clus <- matrix(NA, length(listClus), length(listClus),
               dimnames = list(names(listClus), names(listClus)))

# Completeing the distance matrix
for(i in 1:ncol(clus)) {
  for(j in i:ncol(clus)) {
    # Filling the upper triangular matrix
    clus[i,j] <- distMethods(listClus[[i]], listClus[[j]])
    # Making the matrix symmetric
    clus[j,i] <- clus[i,j]
  }
}
# Dendogram
plot(hclust(as.dist(clus)))
```



Some observations:

- Imputation with the function `kNN` from the **VIM** is very close to simple mean imputation. This is may be due to the fact that default for the function for aggregating the k-nearest neighbors is to replace by the median.
- k-NN imputation varies greatly depending on the package used. The function `impute.knn` is designed for microarray data, and imputes missing elements by *avaraging* those non-missing elements of its neighbors. It selects the k-nearest neighbors for imputation calculating the *euclidean distance*. The function `kNN`, as mentioned before, replaces the missing case by the *median* of its neighbors, and uses a variation of the *Gower distance*.

Fit of different imputation methods

Objective

The objective of this section is to construct a function that recreates the missingness structure of the data at CID1 in a reduced, complete case, data set. Afterwards, the data with the artificial missing values will be imputed with the different methods shown above, and the error of each method will be calculated. This process will be repeated several times to see the difference in the distribution of the errors of each method.

The reduced *clean* data set was already examined in the `clinicAnalysis.Rmd` file. It was constructed by removing the variables that had more than 200 missing values, and then keeping only the individuals with complete information.

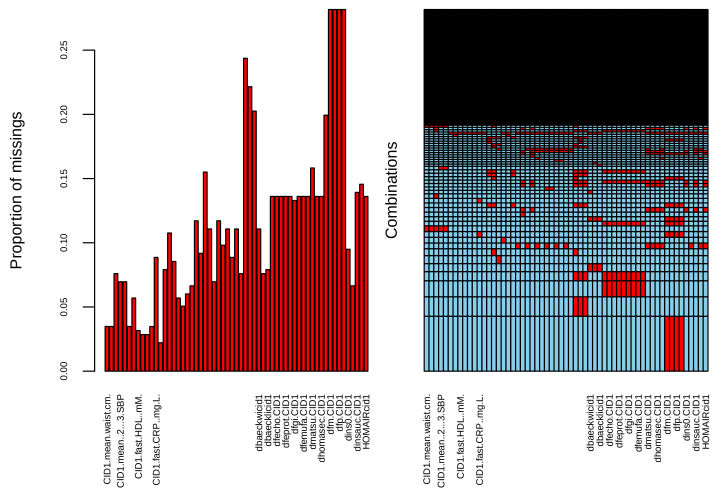
Analysis of missingness structure

In order to recreate the model of missingness in the reduced data set, it is interesting to have a look at the missingness structure of the data at CID1 from different perspectives:

- Look at the structure of the missing values at the whole data in CID1, paying close attention to the structure of the variables with more than 200 missing values.
- The missingness structure at CID1 after the removal of these 14 variables. To facilitate a graphical interpretation, the variables that have complete cases will be taken out for the aggregation plots and missingness maps..

Reduced CID1

I use the aggregation plot (package **VIM**) to see which are the most common combinations of missing variables.



I then create a data set `combR`, that contains combinations of missing values with their percentages.

```
# aggrR: list of class "aggr" containing the following components:
  # x: the data used.
  # tabcomb: indicator matrix for the combinations of variables
  # percent: the percentage of these combinations.
# combR: data frame of the indicator matrix and the percentages.

aggrR <- aggr(CID1[which(as.numeric(naByCol) < 200)], plot=FALSE)
combR <- as.data.frame(aggrR$tabcomb)
colnames(combR) <- colnames(CID1noNA)
combR$percent <- aggrR$percent
combR = combR[order(combR$percent, decreasing=TRUE),]
combR <- combR[-1,] # Remove the most frequent combination (complete cases)
```

There are some combinations of the missing values that are worth looking into:

- The most common combination is that of *dfm.CID1, dffm.CID1, dfp.CID1, dfmet.CID1*, which is present in at least 7.82% of the observations in the reduced data.
- The second most common combination is that of *CID1.U.Album..24h...mg., CID1.fast.U.creat..24h...mmol., CID1.U.C.pept..24h...nmol.*, present in at least 2.77% of the observations in the reduced data.
- The third combination is *dfenergy.CID1, dfecho.CID1, dfefat.CID1, dfeprot.CID1, dfealc.CID1, dfgi.CID1, dfesfa.CID1, dfemufa.CID1, dfepufa.CID1*, corresponding to at least 2.28% of the observations in the reduced data.
- The forth combination is the addition of combination 2 and 3.
- The fifth combination is that of *dbaeckwicid1, dbaecksicid1, dbaecklicid1*.

I will focus in the combinations 1,2,3 and 5. First, I wish to know the total percentage of observations in the *reduced* data set that have this joint structure of missing values.

```r
combs <- c(1:3,5,8,11)
real.percent <- NULL

for (i in combs){
  # Get the columns that have missing values
  cols <-  which(combR[i,]==1)
  # I set a condition, a vector of ones
  cond <- combR[i,cols]
  # I get the rows that have the same pattern
  rows <- which(apply(combR[,cols], 1, function(x)all(x==cond)))
  # Get the total percentage of observations that have the same pattern
  real.percent[i] <- sum(combR$percent[rows])
}

# Comparing the percentages
real.percent[combs]
combR$percent[combs]
```

```
## [1] 14.495114  9.934853  6.840391  3.420195  2.280130  1.465798
## [1] 7.8175896 2.7687296 2.2801303 1.1400651 0.8143322 0.8143322
```

It is clear that only using the percentages from `aggr()` leads to an underestimation of the frequency of certain patterns. Therefore, I need to adjust the percentages accordingly when performing my recreation.

There are other patterns that are apparent in both the aggregation plot and the missingness map, but since they differ slightly, they are not gathered under one simple combination in the `combR` matrix. I identify one other combination of 31 variables.

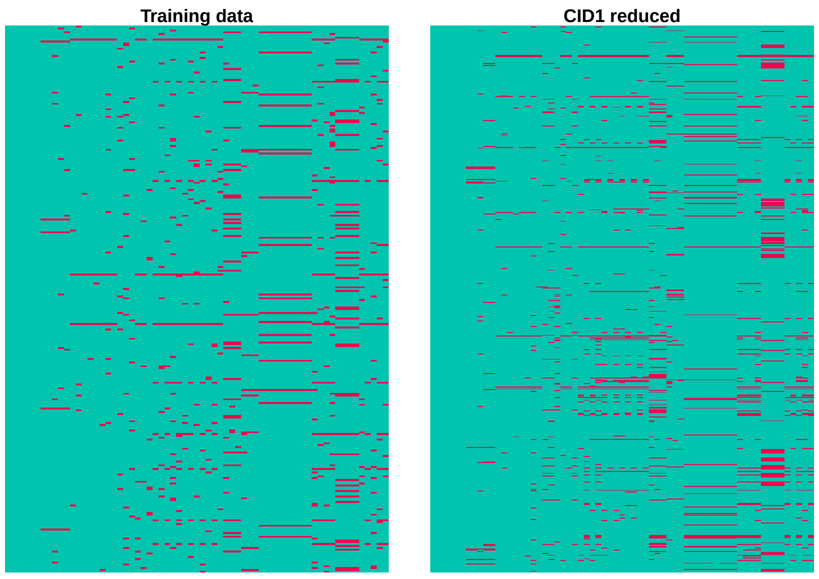
More details in complete version

Simulation in *clean* data set Given the long iteration time of the loop and the restrictions of the computer, I perform the iteration only for 10 repetitions. A loop with more iterations should be run in the future to guarantee robust results.

```r
# Data frames
complete <- CID1[, which(as.numeric(naByCol) < 200)]
complete <- subset(CID1noNA,complete.cases(CID1noNA)==T)
CID1reduced <- CID1[, which(as.numeric(naByCol) < 200)]

comp.var <- which(colSums(is.na(CID1))==0)
index.comb <- combs # Combinations that I wish to recreate
col.factor <- which(sapply(complete, is.factor)==T)

# Initializing the error matrices
err.NIPALS <- NULL
err.kNN <- NULL
err.missForest <- NULL
err.mean <- NULL

set.seed(19920304)
```

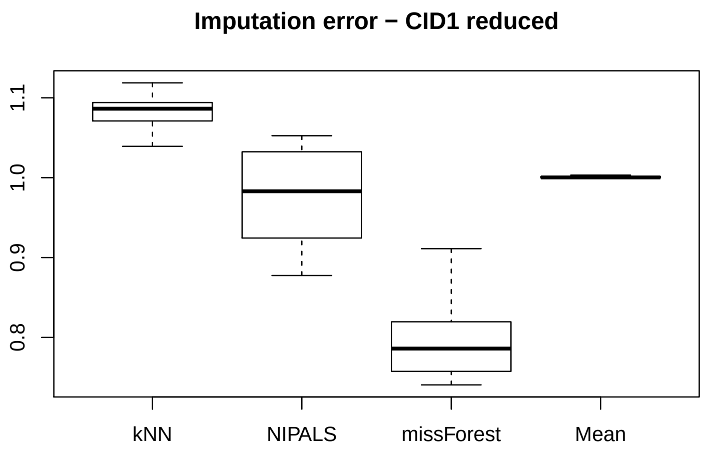
```r
for (n in 1:10){

  train <- complete

  # Combinations from combR
  for(i in index.comb){
    train[sample(nrow(train), ceiling(nrow(train) * real.percent[i]/100)),
          which(combR[i,]==1)] <- NA

  } # closing combR loop

  # Combinations observed
  percent.suppcomb <- sum(combR$percent[xrows])/100
  train[sample(nrow(train), ceiling(nrow(train) * percent.suppcomb)),
        cols.suppcomb] <- NA

  # Percentage of missing values per variable
  prop.miss.var <- apply(CID1reduced, 2, function(x){ sum(is.na(x))/614 })

  # MCAR missingness
  for (j in 1:ncol(train)){

    # Rate of MCAR values
    MCAR.col.rate <- prop.miss.var[j] - colSums(is.na(train))[j]/nrow(train)

    if (MCAR.col.rate>0){

      # Cases that don't have NAs yet
      IND <- which(!is.na(train[,j]),arr.ind=TRUE)
      ntest <- floor(nrow(train)*MCAR.col.rate)

      # Introducing some random NAs
      ind.test <- IND[sample(1:length(IND),ntest)]
      train[ind.test,j] <- NA

    } else
      train[,j] <- train[,j]

  } # closing MCAR loop

  # Scale
  train.sc.mx <- scale(as.matrix(train[,-col.factor]))
  train.scaled <- cbind(as.data.frame(train.sc.mx), train[,col.factor])
  train.scaled <- train.scaled[,colnames(train)] # To leave in original order

  complete.scaled <- scale(as.matrix(complete[,-col.factor]))
  complete.scaled <- cbind(as.data.frame(complete.scaled), train[,col.factor])
  complete.scaled <- complete.scaled[,colnames(complete)]

  # Mean
  train.mean <- apply(train.sc.mx, 2, function(x)ifelse(is.na(x), mean(x, na.rm=TRUE), x))
  train.mean <- as.data.frame(train.mean)
  # NIPALS

  train.nipals <- nipals(train.sc.mx, reconst = TRUE, ncomp = 5)$rec
  id.na <- is.na(train.sc.mx)
  train.nipals[!id.na] <- train.sc.mx[!id.na]

  # missForest
  train.missForest<- missForest(train.scaled, variablewise = TRUE)$ximp

  # VIM::kNN
  train.kNN <- VIM::kNN(train.scaled, k=5, imp_var=FALSE)

  ## Error

  err.kNN <- c(err.kNN,
               mixError(train.kNN, train.scaled, complete.scaled)[1])
  err.missForest <- c(err.missForest,
                      mixError(train.missForest, train.scaled,
                               complete.scaled)[1])
  err.NIPALS <- c(err.NIPALS,
               mixError(train.nipals, train.scaled[,-col.factor],
                       complete.scaled[,-col.factor])[1])

  err.mean <- c(err.mean,
                mixError(train.mean, train.scaled[,-col.factor],
                       complete.scaled[,-col.factor])[1])


} # closing the original loop
```

Checking to see if the training data (the last one) has a somewhat similar missingness structure than the *reduced* data at CID1.



Finally, the resulting boxplot for the imputation errors.

```r
boxplot(cbind(err.kNN, err.NIPALS, err.missForest, err.mean), outline = FALSE,
        main = "Imputation error - CID1 reduced",
        names = c("kNN", "NIPALS", "missForest", "Mean"))
```



Remark The boxplot has the option `outline = FALSE` because for one iteration, the NIPALS algorithm results is a very large imputation error, therefore, it is best to leave it out.

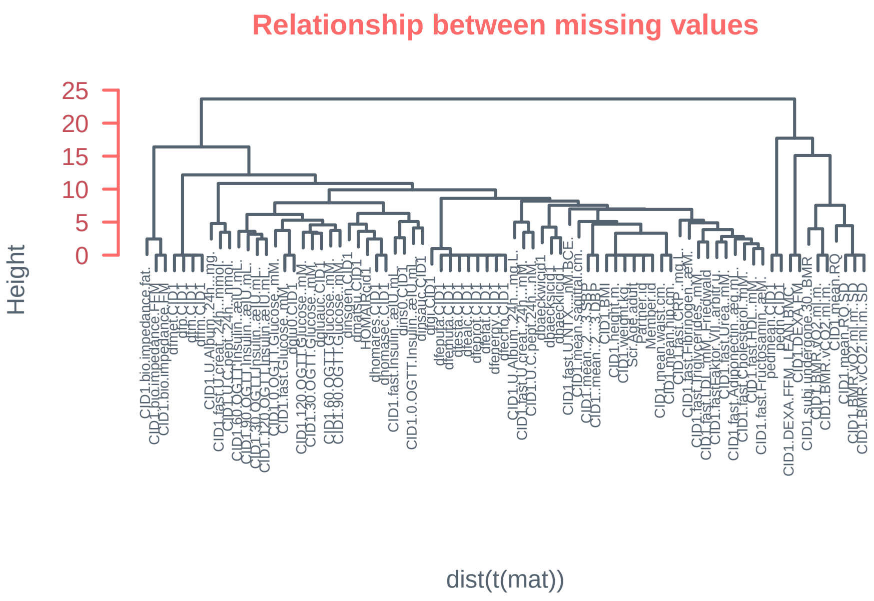
```
##      NRMSE      NRMSE      NRMSE      NRMSE      NRMSE      NRMSE
##  0.9911044  0.9244304  1.0007677  0.9748392 67.0697904  1.0525049
##      NRMSE      NRMSE      NRMSE      NRMSE
##  1.0324168  0.8775216  0.9735896  0.8799608
```

Therefore, the best method appears to be non-parametric iterative imputation by `missForest`. NIPALS algorithm has a very high variability, to the point of generating one imputation with an extremely high error. Finally, it appears that simple mean imputation outperforms k-NN imputation with the package **VIM**.

CID1

Relationship between the variables with missing values:

```r
# Shadow matrix
mat <- matrix(, nrow = nrow(CID1), ncol = ncol(CID1))
for(column in 1:ncol(CID1)){
  mat[, column] <- sapply(CID1[,column], is.na)
}
mat <- ifelse(mat==TRUE, 1, 0)
colnames(mat) <- colnames(CID1)

# Hierarchical clustering
hc <- hclust(dist(t(mat)))
```



In CID1, there are 14 variables that have more than 200 missing values, which corresponds to 17.72% of the variables in that time step. Out of these variables:

- Half may be missing together. This is to recreate the joint missingness of the variables *CID1.subj.undergone.30..BMR, CID1.BMR.vO2.ml.m., CID1.BMR.vO2.ml.m..SD, CID1.BMR.vCO2.ml.m., CID1.BMR.vCO2.ml.m..SD, CID1.mean.RQ, CID1.mean.RQ.SD* in the original data. These variables have a missingness rate of 81.92%.
- the remaining variables can be divided into three pairs, which have a different proportion of missing values. These are:
  - *CID1.bio.impedance.FFM, CID1.bio.impedance.FM, CID1.bio.impedance.fat.* with 36.48% of missing.
  - *CID1.DEXA.FFM..LEAN.BMC., CID1.DEXA.FM* with 62.05% of missing.
  - *pedmean.CID1, pedn.CID1* with 56.35% of missing.

The next step in the analysis would be to select 17.72% of the variables in the reduced data set, and introduce missing values with the rates of missingness specified above.

Session info

```
## R version 3.2.0 (2015-04-16)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## locale:
## [1] LC_COLLATE=French_France.1252  LC_CTYPE=French_France.1252
## [3] LC_MONETARY=French_France.1252 LC_NUMERIC=C
## [5] LC_TIME=French_France.1252
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] RColorBrewer_1.1-2  reshape_0.8.5       ggthemes_2.1.2
##  [4] ggplot2_1.0.1       VIM_4.1.0           data.table_1.9.4
##  [7] colorspace_1.2-6    Amelia_1.7.3        Rcpp_0.11.6
## [10] missForest_1.4      itertools_0.1-3     iterators_1.0.7
## [13] foreach_1.4.2       randomForest_4.6-10 mixOmics_5.0-4
## [16] MASS_7.3-40         impute_1.42.0       FactoMineR_1.29
## [19] DMwR_0.4.1          lattice_0.20-31     knitr_1.10.5
##
## loaded via a namespace (and not attached):
##  [1] vcd_1.3-2            rgl_0.95.1247        class_7.3-12
##  [4] zoo_1.7-12           gtools_3.5.0         digest_0.6.8
##  [7] plyr_1.8.2           chron_2.3-45         evaluate_0.7
## [10] e1071_1.6-4          gplots_2.17.0        minqa_1.2.4
## [13] gdata_2.16.1         SparseM_1.6          car_2.0-25
## [16] TTR_0.22-0           nloptr_1.0.4         rpart_4.1-9
## [19] Matrix_1.2-1         rmarkdown_0.6.1      proto_0.3-10
## [22] splines_3.2.0        lme4_1.1-7           stringr_1.0.0
## [25] foreign_0.8-63       igraph_0.7.1         pheatmap_1.0.2
## [28] munsell_0.4.2        RGCCA_2.0            mgcv_1.8-6
## [31] htmltools_0.2.6      nnet_7.3-9           flashClust_1.01-2
## [34] codetools_0.2-11     bitops_1.0-6         leaps_2.9
## [37] nlme_3.1-120         gtable_0.1.2         magrittr_1.5
## [40] formatR_1.2          scales_0.2.4         KernSmooth_2.23-14
## [43] quantmod_0.4-4       stringi_0.4-1        ROCR_1.0-7
## [46] reshape2_1.4.1       sp_1.1-0             scatterplot3d_0.3-35
## [49] robustbase_0.92-3    xts_0.9-7            tools_3.2.0
## [52] DEoptimR_1.0-2       abind_1.4-3          parallel_3.2.0
## [55] pbkrtest_0.4-2       yaml_2.1.13          cluster_2.0.1
## [58] caTools_1.17.1       quantreg_5.11
```