R tutorial: imputation of missing data using random forests

About imputation

This article shows how to implement a random forest imputation method proposed by *Daniel J. Stekhoven and Peter Bühlmann* via the software R. This article is (very) far from being exhaustive on the subject of missing values (which we will regularly call NA here) and omits a lot of prerequisites.

A very common problem when doing data analysis is missing data. For example, if you make questionnaires, it is rare that all the people interviewed answer all the questions. A (more or less) candid way of approaching this phenomenon would be to say that this is not necessarily a problem. Indeed, not answering a question (for whatever reason) can be considered as a modality in itself. However, it will quickly become apparent that this solution is difficult to consider for continuous variables.

Among the best-known methods in imputation, we find in particular the replacement of missing continuous data by the mean (more generally the median) of the other individuals in the sample and by the mode of the variable in question for qualitative variables. This approach is **very simple to understand and to implement**, but it can be a source of **considerable bias**. Indeed, replacing each NA with the average of the rest of the group will artificially decrease the variance within the sample, in addition to possibly accentuating an already present bias. The problem is quite similar for imputation by mode of the variable.

Why are NaN a problem?

Missing values are a problem mainly because they are a **lack of information**. Unfortunately, this is far from the only problem. As we will see next, missing data are rarely determined purely at random and their absence can give us a biased data set.

Moreover, some statistical methods or algorithms cannot operate when NA is present (or can only operate up to a certain limit). However, I would like to point out that many of them work despite the presence of NA, **depending on the statistical software used**. In R, some functions have as default argument "na.omit = FALSE", which means that the analysis is not performed (you just have to replace it with TRUE to get around this problem). Likewise, some functions will just not be performed in case of NA in the data. For more information on how R handles missing values, see *this article*.

The different types of NaN

Little R. et Rubin D.. propose a 3-modality classification:

- **MCAR (missing completely at random)**. This concerns NA whose absence is determined purely at random and does not depend on the individuals. This is a rather rare situation.
- **MAR (missing at random)**. This is the case where the probability of absence is related to one or more other observed variables.
- **MNAR (missing not at random)**. These missing data are not determined at random. It can be the case when a person is suffering from a disease and does not want to disclose it or the refusal to give the amount of his income when it is high.

Depending on the situation, different solutions (if any) will be considered. For more information, see *Statistical analysis with missing data*.

Brief presentation of decision trees

Decision trees are the **graphical representation** of a sequence of choices. Each node is subdivided into at least 2 following nodes, until reaching the leaf (at the end of the tree), which corresponds to the final decision.

They are widely used in statistical learning since they allow to easily discriminate the important parameters in a large data set and are globally efficient.

Random forests

Random forests designate a type of machine learning algorithm that will rely on decision trees. These forests will in fact **aggregate the results obtained by the decision trees** by choosing, for example, the majority predicted result.

What does this have to do with missing data?

The connection is in fact simple: since random forests allow us to handle mixed data sets as well as to predict variables with complex relationships, it is perfectly feasible to predict our missing values via our observed values! This is precisely what Daniel J. Stekhoven and Peter Bühlmann propose.

The algorithm is not only an implementation of a random forest algorithm but **is slightly more complex**. For a clear and thorough description, I recommend *this article* from the Kaggle platform (a huge data science community).

The advantages of this type of algorithm are, as said before, that it applies to mixed datasets, does not require any particular distribution of variables (because it's non-parametric), fits perfectly to low dimensional data and can propose complex (i.e. non-linear) relationships.

Warning: it is possible that the use of this imputation method is not justified for the treatment of high dimensional data (do some research!).

Transition to practice

Let's see how to set up the algorithm in R. For this, you will need to install the missForest package.

An interesting first step can be to measure the proportion of NA in your dataset. There is, to my knowledge, no pre-defined function in R for this. However, it is not fundamentally complex and it will suffice to use the code below. If you have any, remember to remove the parameters that are not strictly used for data analysis such as merge feature so as not to distort the calculation of the NA proportion in the dataset.

```r
#clean environment
rm(list=ls())

#load a dataset and add 120 (~15%) NaN randomly
data("iris")
df = data.frame(iris)
for (i in 1:120){
  index_row = round(runif(1, 1, nrow(df)))
  index_col = round(runif(1, 1, ncol(df)))
  df[index_row, index_col] = NA
}

vec = is.na(df)
count = sum(vec)
```

```r
prop_nan = count/(ncol(df)*nrow(df))*100
cat('In total there is', round(prop_nan, digits = 2), '% of missing values in the dataset')
```

```
## In total there is 14.4 % of missing values in the dataset
```

Finally, implementing the algorithm is quite simple:

```r
library(missForest)
imputation = missForest(df)
df_imputed = imputation$ximp
vec = is.na(df_imputed)
count = sum(vec)
prop_nan = count/(ncol(df_imputed)*nrow(df_imputed))*100
cat('In total there is', round(prop_nan, digits = 2), '% of missing values in the (imputed) dataset')
```

```
## In total there is 0 % of missing values in the (imputed) dataset
```

Finally, the model also allows us to calculate the error rate of the imputation via the "OOBerror" element. For continuous variables, it is the **mean square error**. For categorical variables, it is the **proportion of false classifications (between 0 and 1)**. The lower these values are, the better the imputation will be considered. When writing a dissertation/thesis or a scientific paper, it is recommended to give these values.

```r
cat("Error measures:\n")
```

```
## Error measures:
```

```r
cat("MSE =", imputation$OOBerror[1], "\n")
```

```
## MSE = 0.13768
```

```r
cat("PFC =", imputation$OOBerror[2])
```

```
## PFC = 0.03597122
```