# How to Handle Missing Data

towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4

Alvira Swalin 31 January 2018

Alvira Swalin
Jan 31

*"The idea of imputation is both seductive and dangerous" (R.J.A Little & D.B. Rubin)*

One of the most common problems I have faced in Data Cleaning/Exploratory Analysis is handling the missing values. Firstly, understand that there is NO good way to deal with missing data. I have come across different solutions for data imputation depending on the kind of problem—Time series Analysis, ML, Regression etc. and it is difficult to provide a general solution. In this blog, I am attempting to summarize the most commonly used methods and trying to find a structural solution.

## Imputation vs Removing Data

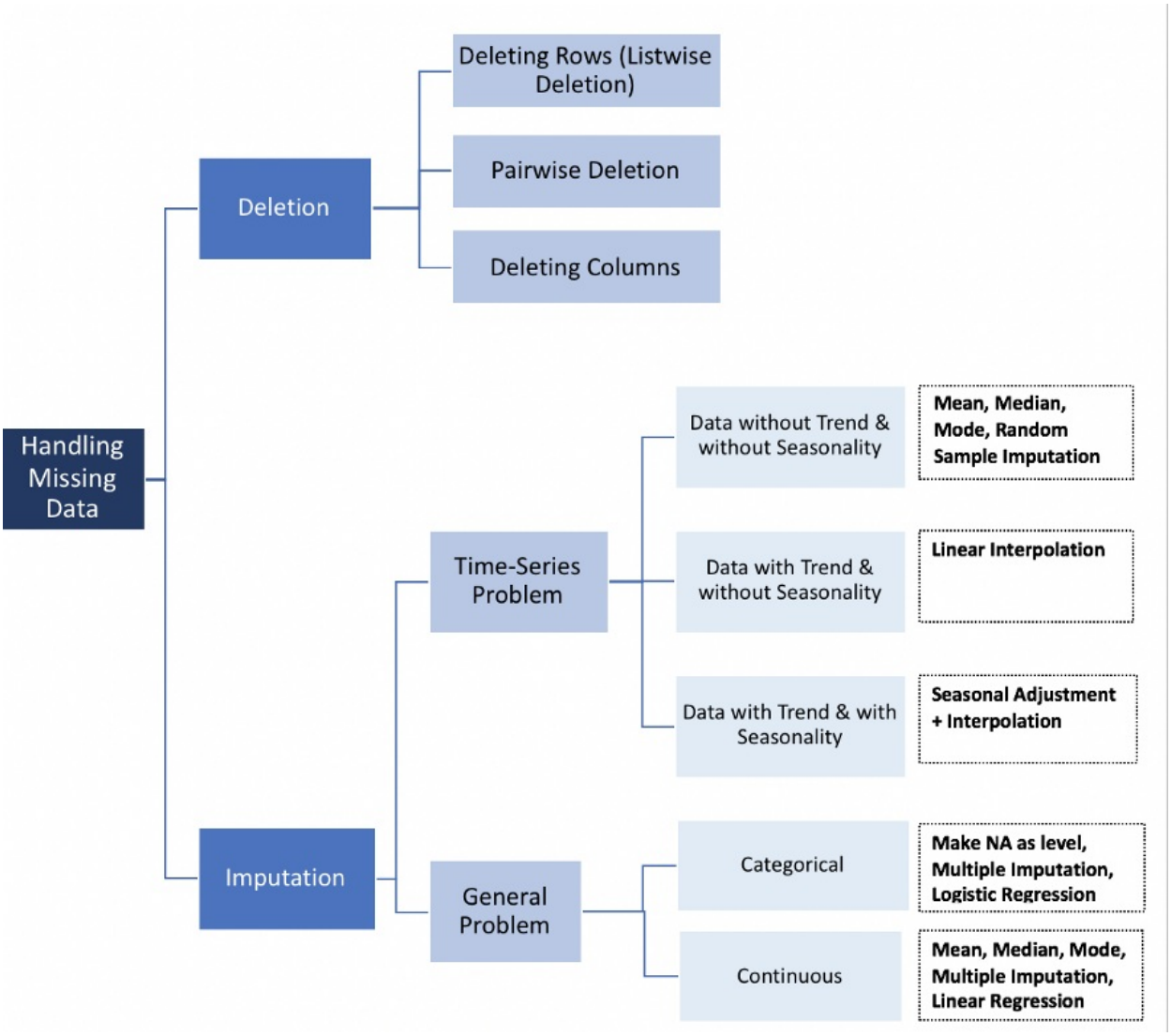
Before jumping to the methods of data imputation, we have to understand the reason why data goes missing.

1. **Missing at Random (MAR):** Missing at random means that the propensity for a data

point to be missing is not related to the missing data, but it is related to some of the observed data

2. **Missing Completely at Random (MCAR):** The fact that a certain value is missing has nothing to do with its hypothetical value and with the values of other variables.
3. **Missing not at Random (MNAR):** Two possible reasons are that the missing value depends on the hypothetical value (e.g. People with high salaries generally do not want to reveal their incomes in surveys) or missing value is dependent on some other variable's value (e.g. Let's assume that females generally don't want to reveal their ages! Here the missing value in age variable is impacted by gender variable)

In the first two cases, it is safe to remove the data with missing values depending upon their occurrences, while in the third case removing observations with missing values can produce a bias in the model. So we have to be really careful before removing observations. Note that imputation does not necessarily give better results.




## Deletion

**Listwise**

Listwise deletion (complete-case analysis) removes all data for an observation that has one or more missing values. Particularly if the missing data is limited to a small number of observations, you may just opt to eliminate those cases from the analysis. However in most cases, it is often disadvantageous to use listwise deletion. This is

because the assumptions of MCAR (Missing Completely at Random) are typically rare to support. As a result, listwise deletion methods produce biased parameters and estimates.

```
newdata <- na.omit(mydata)

# In python
mydata.dropna(inplace=True)
```

**Pairwise**
pairwise deletion analyses all cases in which the variables of interest are present and thus maximizes all data available by an analysis basis. A strength to this technique is that it increases power in your analysis but it has many disadvantages. It assumes that the missing data are MCAR. If you delete pairwise then you'll end up with different numbers of observations contributing to different parts of your model, which can make interpretation difficult.

```
     ageNA DV1 DV2
[1,]    18  NA   9
[2,]    NA   1   4
[3,]    27   5   2
[4,]    22  -3   7
```

Cases 3 and 4 will be used to find covariance between ageNa & DV1

Cases 2,3 and 4 will be used to find covariance between DV1 and DV2

```
#Pairwise Deletion
ncovMatrix <- cov(mydata, use="pairwise.complete.obs")

#Listwise Deletion
ncovMatrix <- cov(mydata, use="complete.obs")
```

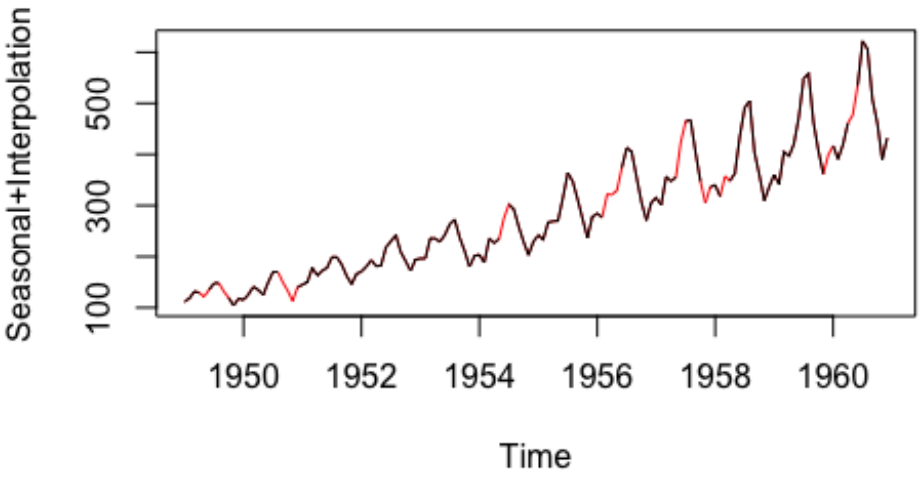
**Dropping Variables**
In my opinion, it is always better to keep data than to discard it. Sometimes you can drop variables if the data is missing for more than 60% observations but only if that variable is insignificant. Having said that, imputation is always a preferred choice over dropping variables

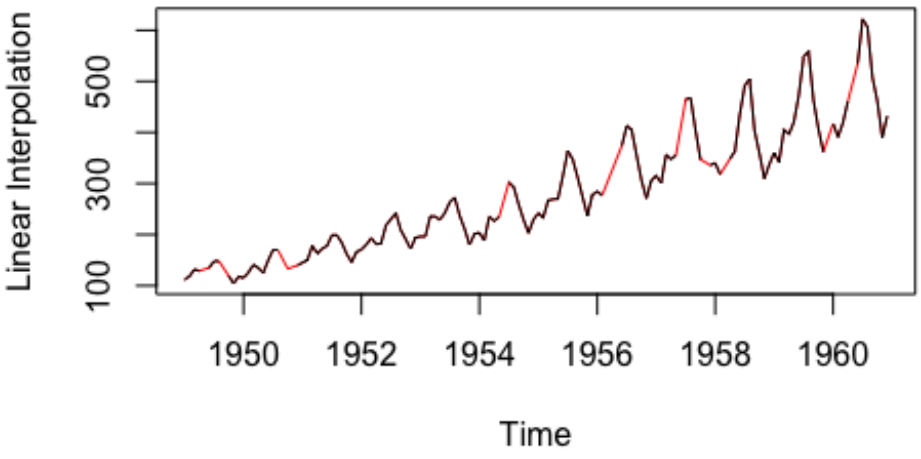
```
df <- subset(mydata, select = -c(x,z) )
df <- mydata[ -c(1,3:4) ]

In python
del mydata.column_name
mydata.drop('column_name', axis=1, inplace=True)
```

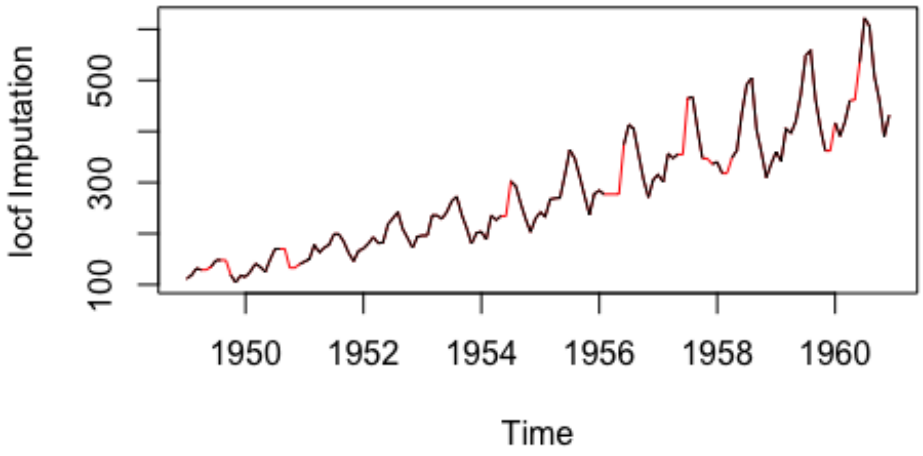
## Time-Series Specific Methods

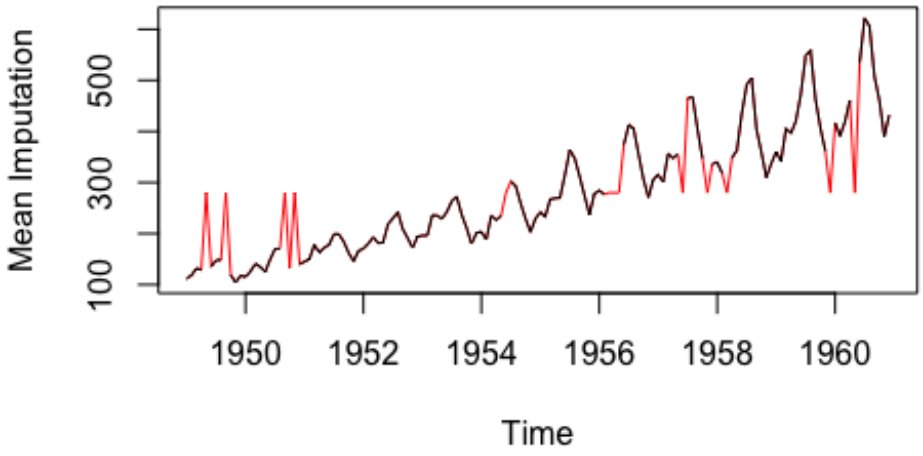
- **Last Observation Carried Forward (LOCF) & Next Observation Carried Backward (NOCB)**
  This is a common statistical approach to the analysis of longitudinal repeated measures data where some follow-up observations may be missing. Longitudinal data track the same sample at different points in time. Both these methods can introduce bias in analysis and perform poorly when data has a visible trend

- **Linear Interpolation**
  This method works well for a time series with some trend but is not suitable for seasonal data
- **Seasonal Adjustment + Linear Interpolation**
  This method works well for data with both trend and seasonality






Data: tsAirgap form library(imputeTS), Interpolated Data in Red

```
library(imputeTS)

na.random(mydata)                 # Random Imputation
na.locf(mydata, option = "locf")   # Last Obs. Carried Forward
na.locf(mydata, option = "nocb")   # Next Obs. Carried Backward
na.interpolation(mydata)          # Linear Interpolation
na.seadec(mydata, algorithm = "interpolation") # Seasonal Adjustment then Linear
Interpolation
```

## Mean, Median and Mode

Computing the overall mean, median or mode is a very basic imputation method, it is the only tested function that takes no advantage of the time series characteristics or relationship between the variables. It is very fast, but has clear disadvantages. One disadvantage is that mean imputation reduces variance in the dataset.

```
library(imputeTS)
```

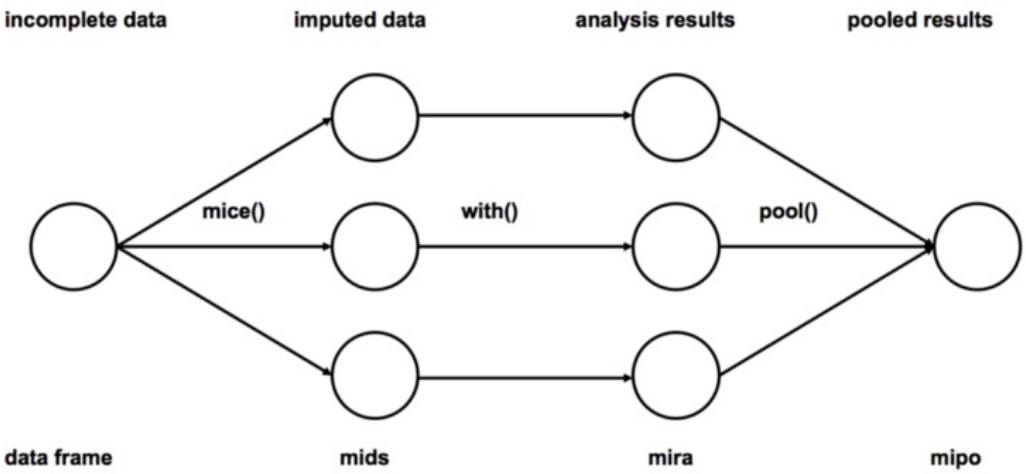
```
na.mean(mydata, option = "mean")   # Mean Imputation
na.mean(mydata, option = "median") # Median Imputation
na.mean(mydata, option = "mode")   # Mode Imputation

In Python
from sklearn.preprocessing import Imputer
values = mydata.values
imputer = Imputer(missing_values='NaN', strategy='mean')
transformed_values = imputer.fit_transform(values)

# strategy can be changed to "median" and "most_frequent"
```

## Linear Regression

To begin, several predictors of the variable with missing values are identified using a correlation matrix. The best predictors are selected and used as independent variables in a regression equation. The variable with missing data is used as the dependent variable. Cases with complete data for the predictor variables are used to generate the regression equation; the equation is then used to predict missing values for incomplete cases. In an iterative process, values for the missing variable are inserted and then all cases are used to predict the dependent variable. These steps are repeated until there is little difference between the predicted values from one step to the next, that is they converge.
It "theoretically" provides good estimates for missing values. However, there are several disadvantages of this model which tend to outweigh the advantages. First, because the replaced values were predicted from other variables they tend to fit together "too well" and so standard error is deflated. One must also assume that there is a linear relationship between the variables used in the regression equation when there may not be one.

### Multiple Imputation

1. **Imputation**: Impute the missing entries of the incomplete data sets *m* times (*m*=3 in the figure). Note that imputed values are drawn from a distribution. Simulating random draws doesn't include uncertainty in model parameters. Better approach is to use Markov Chain Monte Carlo (MCMC) simulation. This step results in m complete data sets.
2. **Analysis**: Analyze each of the *m* completed data sets.
3. **Pooling**: Integrate the *m* analysis results into a final result




Source: http://www.stefvanbuuren.nl/publications/mice%20in%20r%20-%20draft.pdf

```
# We will be using mice library in r
library(mice)
# Deterministic regression imputation via mice
imp <- mice(mydata, method = "norm.predict", m = 1)

# Store data
data_imp <- complete(imp)

# Multiple Imputation
imp <- mice(mydata, m = 5)

#build predictive model
fit <- with(data = imp, lm(y ~ x + z))

#combine results of all 5 models
combine <- pool(fit)
```

This is by far the most preferred method for imputation for the following reasons:
- Easy to use
- No biases (if imputation model is correct)

## Imputation of Categorical Variables

1. Mode imputation is one method but it will definitely introduce bias
2. Missing values can be treated as a separate category by itself. We can create another category for the missing values and use them as a different level. This is the simplest method.
3. Prediction models: Here, we create a predictive model to estimate values that will substitute the missing data. In this case, we divide our data set into two sets: One set with no missing values for the variable (training) and another one with missing values (test). We can use methods like logistic regression and ANOVA for prediction
4. Multiple Imputation

## KNN (K Nearest Neighbors)

There are other machine learning techniques like XGBoost and Random Forest for data imputation but we will be discussing KNN as it is widely used. In this method, k neighbors are chosen based on some distance measure and their average is used as an imputation estimate. The method requires the selection of the number of nearest neighbors, and a distance metric. KNN can predict both discrete attributes (the most frequent value among the k nearest neighbors) and continuous attributes (the mean among the k nearest neighbors)
The distance metric varies according to the type of data:
1. Continuous Data: The commonly used distance metrics for continuous data are Euclidean, Manhattan and Cosine
2. Categorical Data: Hamming distance is generally used in this case. It takes all the categorical attributes and for each, count one if the value is not the same between two points. The Hamming distance is then equal to the number of attributes for which the value was different.
One of the most attractive features of the KNN algorithm is that it is simple to understand and easy to implement. The non-parametric nature of KNN gives it an edge in certain settings where the data may be highly “unusual”.
One of the obvious drawbacks of the KNN algorithm is that it becomes time-consuming when analyzing large datasets because it searches for similar instances through the entire dataset. Furthermore, the accuracy of KNN can be severely degraded with high-dimensional data because there is little difference between the nearest and farthest neighbor.

```
library(DMwR)
knnOutput <- knnImputation(mydata)

In python
from fancyimpute import KNN

# Use 5 nearest rows which have a feature to fill in each row's missing features
knnOutput = KNN(k=5).complete(mydata)
```

Among all the methods discussed above, multiple imputation and KNN are widely used, and multiple imputation being simpler is generally preferred.

If you have any questions about this post, please ask in the comments and I will do my best to answer.

**Resources:**


*1. https://www.bu.edu/sph/files/2014/05/Marina-tech-report.pdf*
*2. https://arxiv.org/pdf/1710.01011.pdf*
*3. https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/*
*4. https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/*
*5. Time-Series Lecture at University of San Francisco by Nathaniel Stevens*