Wale Akinfaderin Follow
Physicist | Insight Data Science Fellow
Sep 11, 2017 · 8 min read

Missing Data Conundrum: Exploration and Imputation Techniques

Why Missing Data

Missing data is a common and exciting problem in statistical analysis and machine learning. They are necessary for evaluating data quality and can have different sources such as users not responding to questions in a recommender system, death of patients on treatment or non-compliance, errors in a database that describes the maintenance information of plant equipment, and so on.

Missing Data Mechanism

For us to fully understand the importance of missing data, we need to comprehensively identify the reasons for missing data occurrence. The first step is to understand your data and more importantly, the data collection process. This can lead to the possibility of reducing data collection errors. The nature or mechanism of missing data can be categorized into four classes. These categories are based on the degree of relationship between the nature of the missing data and observed values. Understanding the mechanism is very useful in understanding the appropriate analysis to use. I will explain these mechanisms briefly:

1) Missing Completely at Random (MCAR): This means that the nature of the missing data is not related to any of the variables, whether missing or observed. In this case, the missingness on the variable is completely unsystematic. For example, let's look at a study that involves determining the reason for obesity among K12 children. MCAR is when the parents forgot to take their kids to the clinic for the study.

2) Missing at Random (MAR): This means that the nature of the missing data is related to the observed data but not the missing data. Using the above K12 study, missing data in this case is due to parents moving to a different city and hence, the children had to leave the study—missingness has nothing to do with the study.

3) Missing Not at Random (MNAR): This is also known as non-ignorable because the missingness mechanism cannot be ignored. They

exist when the missing values are neither MCAR or MAR. The missing values on the variable are related to that of both the observed and unobserved variables. An example of MNAR is that the parents are offended by the nature of the study and do not want their children to be bullied, so they withdrew kids from the study. The difficulty with MNAR data is intrinsically associated with the issue of identifiability.

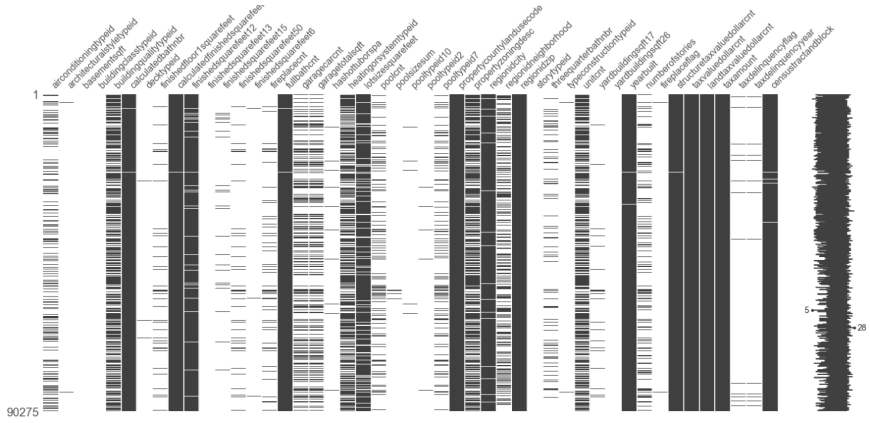
The easiest way to assume a missing data mechanism from data is understanding the data collection process and use substantive scientific knowledge (critical in determining randomness in a missing data). The second method to understand the type of missing data mechanism is statistical testing. This method is mostly used when trying to figure out if the mechanism is either MAR or MCAR.

Exploring Data Missingness

Here, I used the datasets in the ongoing Zillow's Home Value Prediction Competition ($1.2million prize) on Kaggle and I used a python package called missingno. This package is a very flexible missing data visualization tool built with matplotlib and it takes any pandas DataFrame thrown at it. The Kaggle/Zillow data has a training set and a properties dataset that describes the properties of all the homes. I merged both dataset and presented a plot of the missing value matrix.

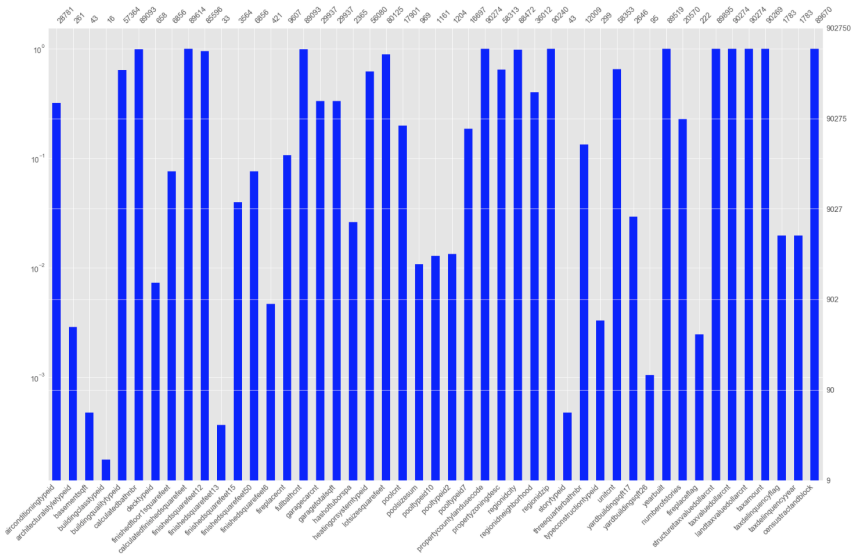
```
import numpy as np
import pandas as pd
import matplotlib
import missingno as msno
%matplotlib inline


train_df = pd.read_csv('train_2016_v2.csv', parse_dates=
["transactiondate"])
properties_df = pd.read_csv('properties_2016.csv')
merged_df = pd.merge(train_df,properties_df)
missingdata_df =
merged_df.columns[merged_df.isnull().any()].tolist()
msno.matrix(merged_df[missingdata_df])
```

The nullity matrix gives you a data-dense display which lets you quickly visually pick out the missing data patterns in the dataset. Also, the *sparkline* on the right gives you a summary of the general shape of the data completeness and an indicator of the rows with maximum and minimum rows.



```
msno.bar(merged_df[missingdata_df], color="blue", log=True,
figsize=(30,18))
```

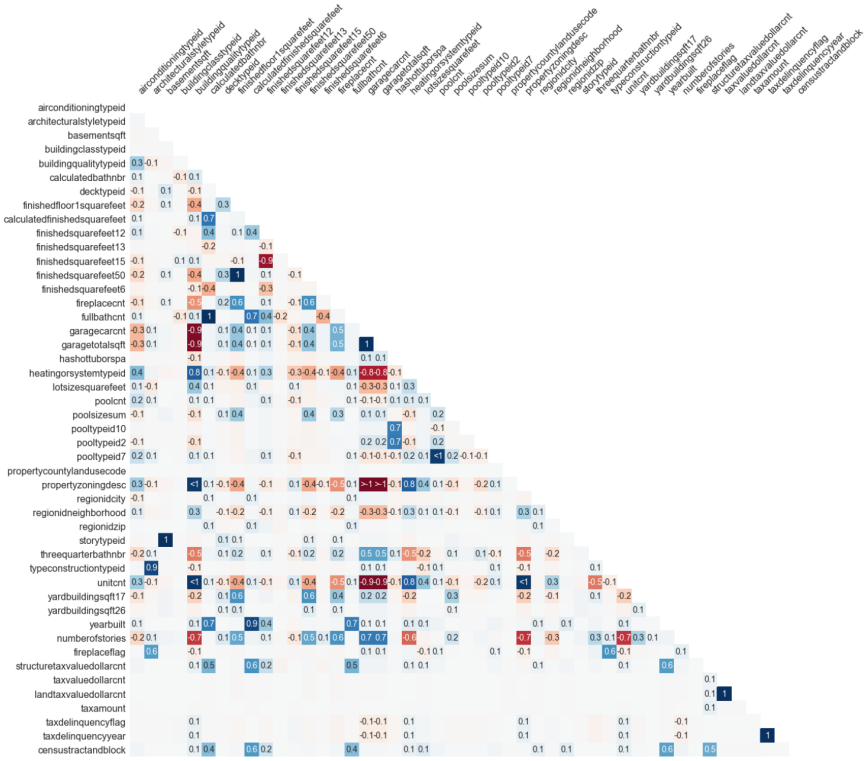
The *missingno* bar chart is a visualization of the data nullity. We log transformed the data on the y-axis to better visualize features with very large missing values.



Finally, a simple correlation *heatmap* is shown below. This map describes the degree of nullity relationship between the different features. The range of this nullity correlation is from -1 to 1 ($-1 \leq R \leq 1$). Features with no missing value are excluded in the *heatmap*. If the nullity correlation is very close to zero ($-0.05 < R < 0.05$), no value will be displayed. Also, a perfect positive nullity correlation ($R=1$) indicates when the first feature and the second feature both have corresponding missing values while a perfect negative nullity

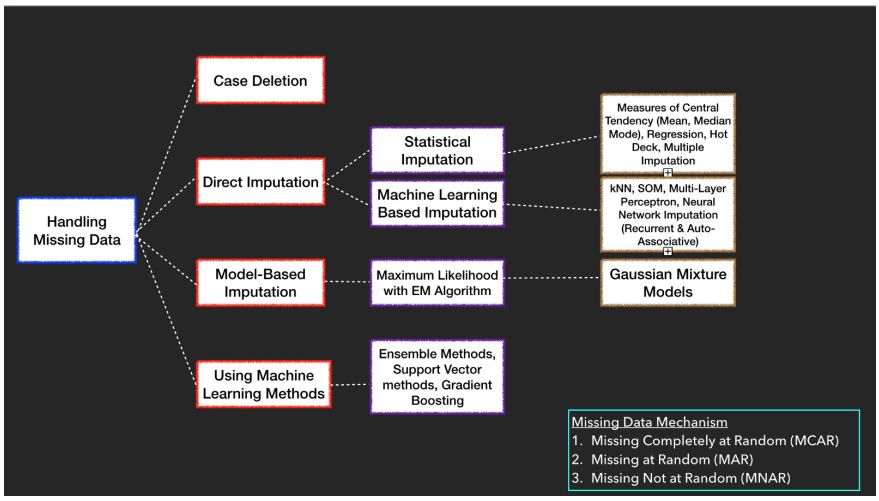
correlation (R=-1) means that one of the features is missing and the second is not missing.

```
msno.heatmap(merged_df[missingdata_df], figsize=(20,20))
```



Handling Missing Data

There are several methods used for treating missing data in literature, textbooks and standard courses. A summary of the methods is shown in Figure 1. Some of these methods started gaining a resurgence in the last decade because of their importance in clinical trials and biomedical studies. In addition, there are certain drawbacks associated with each of these methods when used for data mining and one needs to be careful to avoid bias or the under- or over-estimation of variability. I will explain case deletion and imputation using some fantastic python packages like `pandas` , `sklearn-Imputer` and `fancyimpute` . The underlying principles of model-based imputation methods and machine learning methods (this is different from machine learning imputation methods) is beyond the scope of this article.

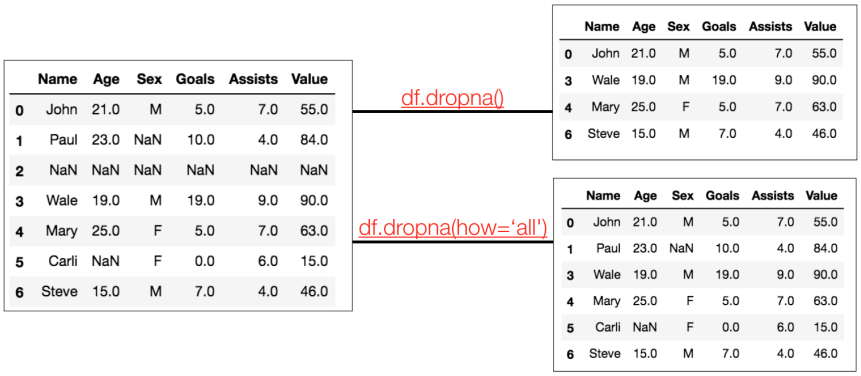


Handling Missing Data and the Different Data Mechanism (Adapted from [1])

Case Deletion

There are two types of case deletion methods. The first one is known as the list deletion (also known as complete case analysis) and the second method is the pair deletion. The case deletion removes all the instances with missing values while in pair deletion, you remove the missing cases from your dataset on an analysis-by-analysis basis. Let's create a dummy dataset with some missing values using pandas dataframe. From the figure below, we can see that `df.dropna()` removes all the missing value and `df.dropna(how ='all')` removes just the rows with missing values. We can also specify removing a column using `df.dropna(axis=1, how='all')` , create a column with missing values using df['New']=np.nan and create a threshold for the number of observations using `df.dropna(thresh=x)` .

```
import pandas as pd
import numpy as np
import fancyimpute
from sklearn.preprocessing import Imputer
data = {'Name': ['John','Paul', np.NaN, 'Wale', 'Mary',
'Carli', 'Steve'], 'Age':
[21,23,np.nan,19,25,np.nan,15],'Sex':
['M',np.nan,np.nan,'M','F','F','M'],'Goals':
[5,10,np.nan,19,5,0,7],'Assists':
[7,4,np.nan,9,7,6,4],'Value': [55,84,np.nan,90,63,15,46]}
df=pd.DataFrame(data, columns =['Name','Age','Sex','Goals',
'Assists', 'Value'])
```



	Name	Age	Sex	Goals	Assists	Value
0	John	21.0	M	5.0	7.0	55.0
1	Paul	23.0	NaN	10.0	4.0	84.0
2	NaN	NaN	NaN	NaN	NaN	NaN
3	Wale	19.0	M	19.0	9.0	90.0
4	Mary	25.0	F	5.0	7.0	63.0
5	Carli	NaN	F	0.0	6.0	15.0
6	Steve	15.0	M	7.0	4.0	46.0

	Name	Age	Sex	Goals	Assists	Value
0	John	21.0	M	5.0	7.0	55.0
3	Wale	19.0	M	19.0	9.0	90.0
4	Mary	25.0	F	5.0	7.0	63.0
6	Steve	15.0	M	7.0	4.0	46.0

	Name	Age	Sex	Goals	Assists	Value
0	John	21.0	M	5.0	7.0	55.0
1	Paul	23.0	NaN	10.0	4.0	84.0
3	Wale	19.0	M	19.0	9.0	90.0
4	Mary	25.0	F	5.0	7.0	63.0
5	Carli	NaN	F	0.0	6.0	15.0
6	Steve	15.0	M	7.0	4.0	46.0

Mean, Median and Mode Imputation

Using the measures of central tendency involves substituting the missing values with the mean or median for numerical variables and the mode for categorical variables. The major limitation of using this method is that it leads to biased estimates of the variances and covariance. The standard errors and test statistics can also be underestimated and overestimated respectively. This imputation technique works well with when the values are missing completely at random. Scikit-learn comes with an imputed function in the form `sklearn.preprocessing.Imputer(missing_values='NaN', strategy='mean', axis=0, verbose=0, copy=True)` . Strategy is the imputation strategy and the default is the "mean" of the axis (0 for columns and 1 for rows). The other strategies are "median" and "most_frequent". Another API that can be used for this imputation is `fancyimpute.SimpleFill()` .

Imputation with Regression

This is an imputation technique that uses information from the observed data to replace the missing values with predicted values from a regression model. The major drawback of using this method is that it reduces variability and overestimates the model fit and correlation coefficient. Scikit-learn preprocessing Imputer function can be utilized for this imputation technique.

k-Neareast Neighbor (kNN) Imputation

For k-Nearest Neighbor imputation, the missing values are based on a kNN algorithm. These values are obtained by using similarity-based methods that rely on distance metrics (Euclidean distance, Jaccard similarity, Minkowski norm etc). They can be used to predict both discrete and continuous attributes. The main disadvantage of using kNN imputation is that it becomes time-consuming when analyzing

large datasets because it searches for similar instances through all the dataset. `fancyimpute.kNN(k=x).complete(data matrix)` can be used for kNN imputation. Choosing the correct value for the number of neighbors (k) is also an important factor to consider when using kNN imputation.

Multiple Imputation using MICE (Multiple Imputation by Chained Equations)

Multiple imputation is a process where the missing values are filled multiple times to create "complete" datasets. Multiple imputation has a lot of advantages over traditional single imputation methods. Multiple Imputation by Chained Equations (MICE) is an imputation method that works with the assumption that the missing data are Missing at Random (MAR). Recall that for MAR, the nature of the missing data is related to the observed data but not the missing data. The MICE algorithm works by running multiple regression models and each missing value is modeled conditionally depending on the observed (non-missing) values. A complete explanation of the MICE algorithm can be seen here. `fancyimpute.MICE().complete(data matrix)` can be used for MICE implementation.

Conclusion

This article highlights the importance of missing data in data science projects. It reviews exploration techniques and important imputation methods used for handling missing data. The other methods not described are model-based and machine learning based methods. Model-based methods assume a joint distribution of all the missing values in the model and estimate the model parameters describing the observed data. A wildly used model-based imputation method is a Pattern Mixture Model (PMM) trained with expectation-maximization (EM-) algorithm. Machine learning algorithms like eXtreme Gradient Boosting (xgboost) automatically learn the best imputation value for the missing data based on the training loss reduction.

References

[1] García Laencina P.J et al. *Pattern Classification with Missing Data: A Review*. Neural Comput Applied. 2009. 9(1): 1–12.

[2] Azur M. J et al. *Multiple Imputation by Chained Equations: what is it and how does it work?*. Int J Methods Psych Res. 2011. 20(1): 40–49.

The author would like to thank Busola Sanusi for the discussion on the mechanism of missing data.

This entry was originally published on IBM Data Science Experience webpage.