Missing Data Analysis and Imputation

Ananth Subramanian, Vitaly Briker, Jackson Au & Richard Farrow

November 03 2019

1. Introduction

As missing data is common in a dataset, this case study explores different factors that contribute to missing data and avenues to address missing data.

With various reasons why data are missing, missing data are classified as follows:

- **Missing completely at random:** In a chemical laboratory when measurements of certain parameters (example pH values) are done and during one of the measurements, the test tube breaks, data cannot be collected for that sample. This kind of missing data are classified under "Missing completely at random"
- **Missing at random:** Collection of blood pressure data of young and older patients. When the dataset contains age, presence of cardiovascular disease and blood pressure and if blood pressure values are missing for few people then it can be considered as data missing at random. The missing values can be filled by certain assumptions like younger people with no cardiovascular disease will have normal blood pressure.
- **Missing not at random:** When data are collected between "Age" and "IQ" and if the dataset misses IQ values for people with low IQ then it can be considered "Missing not at random".

This case-study covers measurement of performance of dataset with missing values after imputing them with at least one method. There are several methods of imputation and few are:

- Listwise Deletion
- Pairwise Deletion
- Single Imputation
- Multiple Imputation

2. Methods

The dataset used for this case-study is the "Boston Housing Data" set and it has around 14 variables.

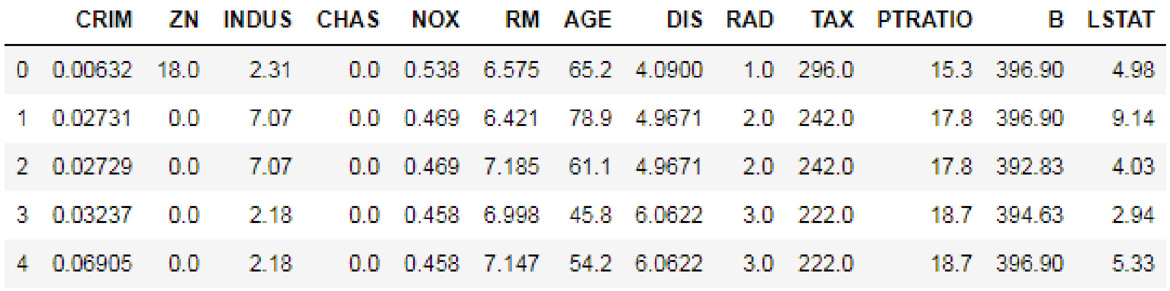
CRIM: Per capita crime rate by town
ZN: Proportion of residential land zoned for lots over 25,000 sq. ft
INDUS: Proportion of non-retail business acres per town
CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX: Nitric oxide concentration (parts per 10 million)
RM: Average number of rooms per dwelling
AGE: Proportion of owner-occupied units built prior to 1940
DIS: Weighted distances to five Boston employment centers
RAD: Index of accessibility to radial highways
TAX: Full-value property tax rate per $10,000
PTRATIO: Pupil-teacher ratio by town
B: 1000(Bk — 0.63)^2, where Bk is the proportion of [people of African American descent] by town

LSTAT: Percentage of lower status of the population
MEDV: Median value of owner-occupied homes in $1000s

MEDV is chosen as the response or target variable and the remaining variables are feature variables. The values are removed to replicate the different types of missing data (**Missing Completely at Random**, **Missing at Random**, and **Missing Not at Random**) and the performance is calculated and compared with the baseline. The different scenarios and the outcome are discussed in the "Results" and "Conclusion" sections.

3. Results

- Step 1 – Using Sklearn we downloaded the Boston Housing dataset.



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

Table 1 – Sample of the data set with explanatory variables

- Step 2 – Create a baseline results by fitting linear regression model using original data set and calculating RMSE and R^2 for further comparison.

RMSE	R^2
4.67919	0.74064

Table 2 – Original data set, Baseline Statistics

- Step 3 – Create missing data, in **Completely at Random** pattern of feature "RM" (Average number of rooms per dwelling) by removing 1, 5 10, 20, 33, and 50% of the data and impute missing values by Mean value.

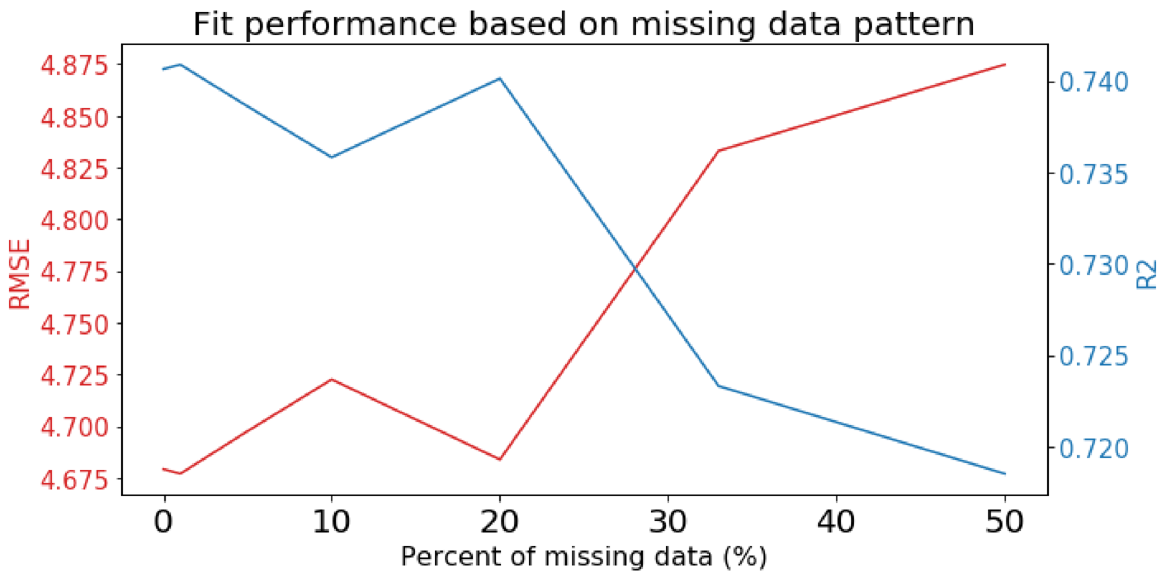


FIGURE 1 – Model statistics trend of **Completely at Random** pattern

- Step 4 – Create missing data, in **Random** pattern when controlled for a third variable. The controlled feature is “ZM” (Proportion of residential land zoned for lots over 25,000 sq. ft) has approximately 73.5% of zero value. The manipulated feature is “RM” and we remove gradually 10, 20, and 30% of the data and impute missing values by Mean value.

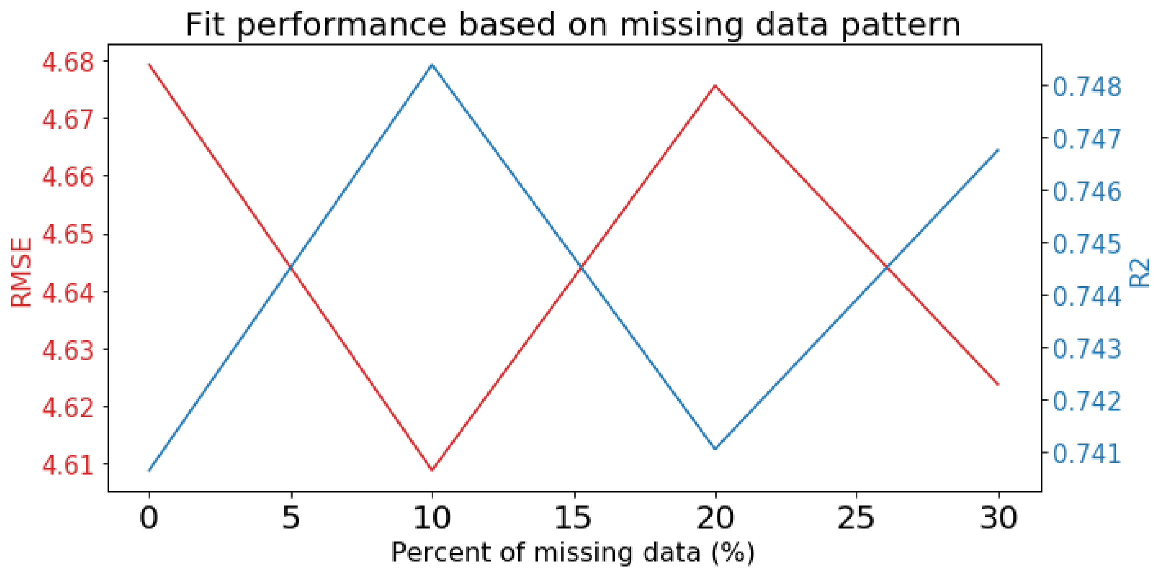


FIGURE 2 – Model statistics trend of **at Random** pattern

- Step 5 – Create missing data, in **Not at Random** pattern. The selected feature in this case is “PTRATIO” (Pupil-teacher ratio by town) that has approximately 27.7% of the same value. The whole subset was imputed by Mean value.

Case	RMSE	R^2
Baseline	4.67919	0.74064
Not at Random pattern	4.70222	0.73808

Table 3 – Model statistics of Baseline Statistics vs **Not at Random** pattern

- Step 6 - Create missing data, in **Not at Random** pattern. The selected feature in this case is “PTRATIO” (Pupil-teacher ratio by town) that has approximately 27.7% of the same value. Using MCMC method we imputed the data used SAS software.

Case	RMSE	R^2
Baseline	4.67919	0.74064
Not at Random pattern	4.70222	0.73808
MCMC - Not at Random pattern	4.57735	0.7521

Table 4 – Model statistics of Baseline Statistics vs **Not at Random** pattern imputed by MCMC algorithm

4. Conclusion

By comparing the **Completely at Random** pattern and imputing the results with the baseline regression model parameters we can see that RMSE is increasing and R^2 is trending down as expected. In our case it is approximately 1% change after imputing the data using Mean value in **Completely at Random** pattern scenario.

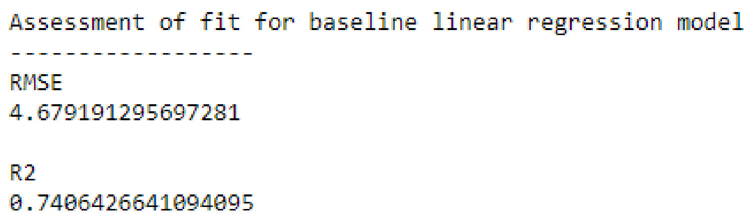
In case of missing values in **Random** pattern when controlled for a third variable we see up to 30% of imputed data, regression model parameters are fluctuating without a trend compared to previous case.
In case of **Not at Random** pattern imputation the regression model parameters are slightly worse using Mean compare to baseline results but using MCMC imputation the results are further from the baseline.

In our case using simple mean imputation is preferable on MCMC method.

Imputation of data can definitely create a higher accuracy model and in many cases the choice to impute data is a desired and necessary step. However, it is imperative to consider why the data is missing as it can have a potential of creating biases into a model.

5. Appendix – Code

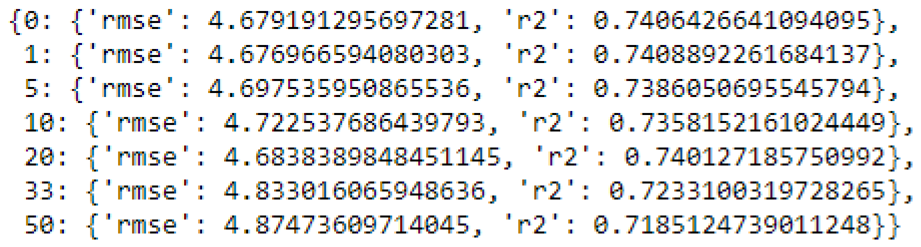
```
Python :
import pandas as pd
import numpy as np
from sklearn.datasets import load_boston
boston_dataset = load_boston()
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
np.random.seed(42)

print(boston_dataset.keys())
       dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
boston = pd.DataFrame(boston_dataset.data,
                   columns=boston_dataset.feature_names)
boston['MEDV'] = boston_dataset.target

#Result stats dictionary
sum_dict={}

x = boston.loc[:, boston.columns != 'MEDV']
y = boston['MEDV']

#create a model
baseline_lm = LinearRegression()
baseline_lm.fit(x, y)
y_predict = baseline_lm.predict(x)

#parameter calculation
rmse = (np.sqrt(mean_squared_error(y, y_predict)))
r2 = r2_score(y, y_predict)
print("Assessment of fit for baseline linear regression
model")
print("------------------")
print("RMSE")
print(rmse)
print("\nR2")
print(r2)
```



```
Assessment of fit for baseline linear regression model
------------------
RMSE
4.679191295697281

R2
0.7406426641094095
```

```
#function to remove and imputed data Missing Completely at
#Random
def runner_repeated (boston_samp,miss=0,col=['AGE']):
    if (miss !=0):
        onepercent_rows = boston_samp.sample(frac =
                        miss/100,random_state=42)
        for i in onepercent_rows.index:
            boston_samp[col][i] = np.NaN

    imputer=boston_samp[column].mean()
    boston_samp[col].fillna((imputer), inplace=True)
    x = boston_samp.loc[:, boston_samp.columns != 'MEDV']
    y = boston_samp['MEDV']
    baseline_lm = LinearRegression()
    baseline_lm.fit(x, y)
    y_predict = baseline_lm.predict(x)
    rmse = (np.sqrt(mean_squared_error(y, y_predict)))
    r2 = r2_score(y, y_predict)
    sum_dict[miss]={}
    sum_dict[miss]["rmse"]=rmse
    sum_dict[miss]["r2"]=r2

#function to convert dictionary result for ploting
def converttoplot (sum_dict):
    j=0
    data1=[]
    data2=[]
    for i in sum_dict:
        for j in sum_dict[i]:
            if j=='rmse': data1.append(sum_dict[i]['rmse'])
            if j=='r2': data2.append(sum_dict[i]['r2'])
    return data1,data2

#plot result data
def plotData(sampletuple):
    data1,data2=converttoplot (sum_dict)

    plt.figure(figsize=(10,5))

    ax1 = plt.subplot(111)

    plt.xticks( fontsize = 20)
    plt.title("Fit performance based on missing data
pattern", fontsize=20)

    color = 'tab:red'
    ax1.set_xlabel('Percent of missing data (%)',
fontsize=16)
    ax1.set_ylabel('RMSE', color=color, fontsize=16)
    ax1.plot(sampletuple, data1, color=color)
    ax1.tick_params(axis='y', labelcolor=color)
    plt.yticks( fontsize = 15)
    ax2 = ax1.twinx()

    color = 'tab:blue'
    ax2.set_ylabel('R2', color=color, fontsize=16)
    ax2.plot(sampletuple, data2, color=color)
    ax2.tick_params(axis='y', labelcolor=color,)

    plt.yticks( fontsize = 15)
    plt.tight_layout()
    plt.show()
```

```
#call function data Missing Completely at Random
column='RM'
sum_dict={}
imputer=0
sampletuple = (0,1,5,10,20,33,50)

for i in sampletuple:
    bb=boston.copy()
    runner_repeated(bb, miss=i,col=column)
plotData(sampletuple)
sum_dict
```



```
{0: {'rmse': 4.679191295697281, 'r2': 0.7406426641094095},
 1: {'rmse': 4.676966594080303, 'r2': 0.7408892261684137},
 5: {'rmse': 4.697535950865536, 'r2': 0.7386050695545794},
 10: {'rmse': 4.722537686439793, 'r2': 0.7358152161024449},
 20: {'rmse': 4.6838389848451145, 'r2': 0.740127185750992},
 33: {'rmse': 4.833016065948636, 'r2': 0.7233100319728265},
 50: {'rmse': 4.87473609714045, 'r2': 0.7185124739011248}}
```

```
#function to remove and imputed data Missing at Random
def runnerMissRandon (boston_samp, miss,col):

    ZNrows = boston_samp.loc[boston_samp.ZN == 0]

    if (miss !=0):
        ZNrows_10 = ZNrows.sample(frac = miss/100,
             random_state=42)

        for i in ZNrows_10.index:
            boston_samp[col][i] = np.NaN

    imputer=boston_samp[column].mean()
    boston_samp[col].fillna((imputer), inplace=True)

    x = boston_samp.loc[:, boston_samp.columns != 'MEDV']
    y = boston_samp['MEDV']

    baseline_lm = LinearRegression()
    baseline_lm.fit(x, y)
    y_predict = baseline_lm.predict(x)

    rmse = (np.sqrt(mean_squared_error(y, y_predict)))
    r2 = r2_score(y, y_predict)
    sum_dict[miss]={}
    sum_dict[miss]["rmse"]=rmse
    sum_dict[miss]["r2"]=r2

#call function data Missing at Random
column='RM'
sum_dict={}
imputer=0
sampletuple = (0,10,20,30)

for i in sampletuple:
    bb=boston.copy()
    runnerMissRandon(bb, miss=i,col=column)
plotData(sampletuple)
sum_dict
```

```
{0: {'rmse': 4.679191295697281, 'r2': 0.7406426641094095},
 10: {'rmse': 4.608835315300141, 'r2': 0.7483833849556861},
 20: {'rmse': 4.675543316759945, 'r2': 0.7410469054532929},
 30: {'rmse': 4.623758901290144, 'r2': 0.7467512582308403}}
```

```
# function data Missing not an Random
def runnerMissNotRandom (boston_samp, value, miss,col):

    rows_25 = boston_samp.loc[boston_samp['PTRATIO'] ==
             value]

    if (miss !=0):
        for i in rows_25.index:
            boston_samp['PTRATIO'][i] = np.NaN

    imputer=boston_samp['PTRATIO'].mean()
    boston_samp.to_csv('hrdata_modified.csv')
    boston_samp['PTRATIO'].fillna((imputer), inplace=True)

    x = boston_samp.loc[:, boston_samp.columns != 'MEDV']
    y = boston_samp['MEDV']
    np.random.seed(42)
    baseline_lm = LinearRegression()
    baseline_lm.fit(x, y)
    y_predict = baseline_lm.predict(x)

    rmse = (np.sqrt(mean_squared_error(y, y_predict)))
    r2 = r2_score(y, y_predict)
    sum_dict[miss]={}
    sum_dict[miss]["rmse"]=rmse
    sum_dict[miss]["r2"]=r2

#Find the feature with ~25% same value
for col in boston:
    print(col)
    print(boston[col].value_counts(normalize=True)*100)
    print(boston[col].describe())
    print("----------------------")

#call function data Missing not an Random
column=['PTRATIO']
sum_dict={}
imputer=0
sampletuple = (0,27)
val=20.2

for i in sampletuple:
    bb=boston.copy()
    runnerMissNotRandom(bb, val, miss=i,col=column)
plotData(sampletuple)
sum_dict
{0: {'rmse': 4.679191295697281, 'r2': 0.7406426641094095},
 27: {'rmse': 4.7022285847319685, 'r2': 0.7380825643981362}}
```

SAS part :

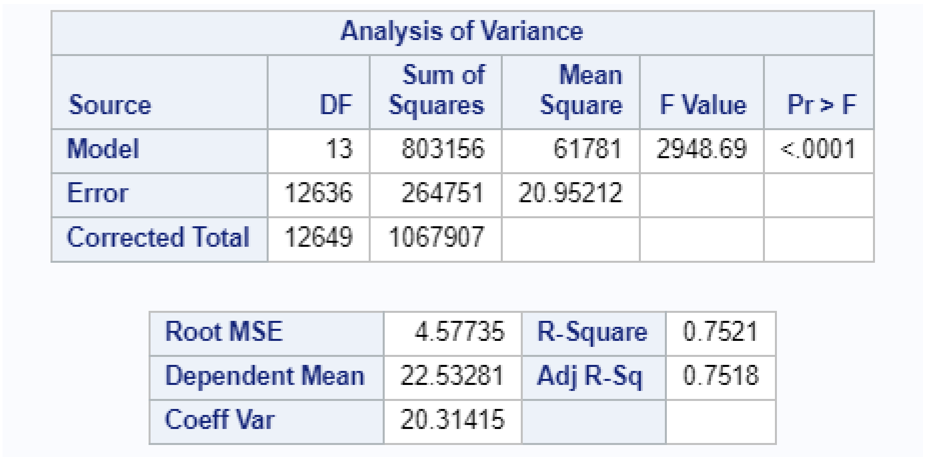
```
data boston_housing;
infile '/home/anantharams0/boston_housing.csv' dsd
truncover;

input crim zn indus chas nox rm age dis rad tax ptratio
b lstat medv;
run;

PROC PRINT DATA=boston_housing;RUN;

PROC REG DATA = boston_housing;
   MODEL medv = crim zn indus chas nox rm age dis rad
tax ptratio b lstat;
RUN;
```

The code below is for running linear regression in SAS after imputing the missing values using MCMC algorithm:
Imputation of missing values:

```
PROC MI DATA = boston_housing
      OUT = boston_housing_imputed  seed = 35399;
     VAR crim zn indus chas nox rm age dis rad tax
ptratio b lstat medv;
RUN;
```

The output is saved as "boston_housing_imputed" dataset. This dataset is then used for running linear regression:

```
PROC REG DATA = boston_housing_imputed;
   MODEL medv = crim zn indus chas nox rm age dis rad
tax ptratio b lstat;
RUN;
```



Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	13	803156	61781	2948.69	<.0001
Error	12636	264751	20.95212		
Corrected Total	12649	1067907			

Root MSE	4.57735	R-Square	0.7521
Dependent Mean	22.53281	Adj R-Sq	0.7518
Coeff Var	20.31415		