MATH2349 Semester 1, 2020

Code

Assignment 2

Akshat Vijayvargia and s3826627

R-PUBS LINK

https://rpubs.com/s3826627/620701 (https://rpubs.com/s3826627/620701)

Required packages

List of required packages to reproduce the report.

Hide

```
library(knitr)
library(ggplot2)
library(readr)
library(tidyr)
library(deductive)
library(validate)
library(Hmisc)
library(stringr)
library(lubridate)
library(outliers)
library(MVN)
library(MASS)
library(caret)
library(dplyr)
```

EXECUTIVE SUMMARY

I am making this report to preprocess the data for further analysis. I will provide 5 step framework to give you essence of t what I did in this report.

1) COLLECTION OF DATA-> I have collected both the datasets from kaggle and all the datasets are opensource and authentic. To import the dataset, I have used read_csv under library(readr) and to merge the two dataset, I have used left_join() function.

2) UNDERSTANDING THE DATASET-> This dataset provides insights to India's situation fighting COVID-19. To check the datatypes and correct the datatype, I have used str() function, as.date() and as.factor().

3) TIDY AND MANIPULATE-> This dataset was bit untidy, so I have used gather() function to make data much more presentable. And wanted to create a new variable through mutate() function for better understanding.

4) SCANNING-> Every data set has some outliers and NAs, so to deal with all these I have used sum(is.na(x)), impute(), z-score() and winsoring technique.

5) TRANSFORMING-> To transform the data and make it closer to normal distribution, I have use hist() to check and the Boxcox() transformation to transform the dataset.

DATA

I have picked two datasets, which are related to the prevailing situation of COVID-19 in India. I downloaded the datasets from the website kaggle, which is a subsidiary of Google that functions as a community for data scientists and developers and is an opensource platform for all datas. The link to the datasets->https://www.kaggle.com/sudalairajkumar/covid19-in-india/tasks?taskId=631 (https://www.kaggle.com/sudalairajkumar/covid19-in-india/tasks?taskId=631)

DATASET 1-> STATE-WISE TESTING DETAILS This dataset contains the informaton of number of positive, negative and total cases from different states of India. The description of variables- **a)Date**->The date of tests recorded **b)States**->List of states fighting COVID-19. **c)Total Samples**->Numbers of recorded test samples in every states. **d)Negative**->Numbers of people, who had Corona -ve result. **e)Positive**->Number of people having Corona desiese. **f)Zones**->Zones define the category of the pandemic.

Now, let’s import the dataset.

Hide

```
# IMPORTING DATASET 1( STATEWISE TESTING DETAILS)
library(readr)
TEST <- read_csv("StatewiseTestingDetails.csv")
```

```
Parsed with column specification:
cols(
  Date = �[31mcol_character()�[39m,
  State = �[31mcol_character()�[39m,
  TotalSamples = �[32mcol_double()�[39m,
  Negative = �[32mcol_double()�[39m,
  Positive = �[32mcol_double()�[39m,
  Zones = �[32mcol_double()�[39m
)
```

Hide

```
head(TEST)
```

Date <chr>	State <chr>	TotalSamples <dbl>	Negative <dbl>	Positive <dbl>
17/04/2020	Andaman and Nicobar Islands	1403	1210	12
24/04/2020	Andaman and Nicobar Islands	2679	NA	27
27/04/2020	Andaman and Nicobar Islands	2848	NA	33
1/05/2020	Andaman and Nicobar Islands	3754	NA	33

Date <chr>	State <chr>	TotalSamples <dbl>	Negative <dbl>	Positive <dbl>	Zo <dl
16/05/2020	Andaman and Nicobar Islands	6677	NA	33	
2/04/2020	Andhra Pradesh	1800	1175	132	

6 rows

DATASET 2-> HOSPITAL BEDS INDIA This dataset contains the information of number of hospital beds in India including Government hospitals and Private hospitals. The description of variables- **a)States**->List of states fighting COVID-19. **b)NumPrimaryHealthCenters_HMIS**->Number of beds in Primary health centers. **c)NumCommunityHealthCenters_HMIS**->Number of beds in Community centers. **d)NumSubDistrictHospitals_HMIS**->Number of beds in Sub-district hospitals. **e)NumDistrictHospitals_HMIS**->Number of beds in District hospitals. **f)TotalPublicHealthFacilities_HMIS**->Number of total beds in public health facilities. **g)NumPublicBeds_HMIS**->Number of public beds. **h)NumRuralHospitals_NHP18**->Number of beds of Rural hospitals. **i)NumRuralBeds_NHP18**->Number of beds in Rural. **j)NumUrbanHospitals_NHP18**->Number of beds in urban hospitals. **k)NumUrbanBeds_NHP18**->Number of beds in Urban. **L)Total_Beds**->Number of beds in total.

Hide

```
# IMPORTING DATASET 2( HOSPITAL BEDS INDIA)
library(readr)
BEDS <- read_csv("HospitalBedsIndia.csv")
```

```
Parsed with column specification:
cols(
  State = [31mcol_character()[39m,
  NumPrimaryHealthCenters_HMIS = [32mcol_number()[39m,
  NumCommunityHealthCenters_HMIS = [32mcol_double()[39m,
  NumSubDistrictHospitals_HMIS = [32mcol_double()[39m,
  NumDistrictHospitals_HMIS = [32mcol_double()[39m,
  TotalPublicHealthFacilities_HMIS = [32mcol_double()[39m,
  NumPublicBeds_HMIS = [32mcol_double()[39m,
  NumRuralHospitals_NHP18 = [32mcol_double()[39m,
  NumRuralBeds_NHP18 = [32mcol_double()[39m,
  NumUrbanHospitals_NHP18 = [32mcol_double()[39m,
  NumUrbanBeds_NHP18 = [32mcol_double()[39m,
  Total_Beds = [32mcol_double()[39m
)
```

Hide

```
head(BEDS)
```

State <chr>	NumPrimaryHealthCenters_HMIS <dbl>	NumCommuni

State <chr>	NumPrimaryHealthCenters_HMIS <dbl>	NumCommunityH
Andaman & Nicobar Islands	27	
Andhra Pradesh	1417	
Arunachal Pradesh	122	
Assam	1007	
Bihar	2007	
Chandigarh	40	

6 rows | 1-3 of 12 columns

After importing both the datasets, now we will merge them through left_join() function.

Hide

```
# MERGING TWO DATASETS
COMB<-left_join(BEDS,TEST,by="State")
head(COMB)
```

State <chr>	NumPrimaryHealthCenters_HMIS <dbl>	NumCommunityH
Andaman & Nicobar Islands	27	
Andhra Pradesh	1417	
Andhra Pradesh	1417	
Andhra Pradesh	1417	
Andhra Pradesh	1417	
Andhra Pradesh	1417	

6 rows | 1-3 of 17 columns

UNDERSTAND

This is the step, where we will check the datatypes of all the variables and attributes, if there are variables with wrong or different datatype then will change them to thier designated one. First, we will look the datatypes of all variables of the table COMB using str() function.

Hide

```
# STRUCTURE OF THE MERGED TABLE
str(COMB)
```

```
tibble [1,170 x 17] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ State                           : chr [1:1170] "Andaman & Nicobar Islands" "Andh
ra Pradesh" "Andhra Pradesh" "Andhra Pradesh" ...
 $ NumPrimaryHealthCenters_HMIS    : num [1:1170] 27 1417 1417 1417 1417 ...
 $ NumCommunityHealthCenters_HMIS  : num [1:1170] 4 198 198 198 198 198 198 198 198
198 ...
 $ NumSubDistrictHospitals_HMIS    : num [1:1170] NA 31 31 31 31 31 31 31 31 31 ...
 $ NumDistrictHospitals_HMIS       : num [1:1170] 3 20 20 20 20 20 20 20 20 20 ...
 $ TotalPublicHealthFacilities_HMIS: num [1:1170] 34 1666 1666 1666 1666 ...
 $ NumPublicBeds_HMIS              : num [1:1170] 1246 60799 60799 60799 60799 ...
 $ NumRuralHospitals_NHP18         : num [1:1170] 27 193 193 193 193 193 193 193 19
3 193 ...
 $ NumRuralBeds_NHP18              : num [1:1170] 575 6480 6480 6480 6480 6480 6480
6480 6480 6480 ...
 $ NumUrbanHospitals_NHP18         : num [1:1170] 3 65 65 65 65 65 65 65 65 65 ...
 $ NumUrbanBeds_NHP18              : num [1:1170] 500 16658 16658 16658 16658 ...
 $ Total_Beds                      : num [1:1170] 2419 87527 87527 87527 87527 ...
 $ Date                            : chr [1:1170] NA "2/04/2020" "10/04/2020" "11/0
4/2020" ...
 $ TotalSamples                    : num [1:1170] NA 1800 6374 6958 6958 ...
 $ Negative                        : num [1:1170] NA 1175 6009 6577 6553 ...
 $ Positive                        : num [1:1170] NA 132 365 381 405 432 473 525 53
4 572 ...
 $ Zones                           : num [1:1170] NA 1 2 2 2 2 2 2 2 2 ...
```

If we look at the result of str(COMB), then we can see the datatype of most of the variables are fine except the Date, which should be in date format rather than character and Zones, which should be in factor rather than numeric. So, now let's convert them right away.

Hide

```
# CONVERTING DATATYPE OF DATE
COMB$Date<-as.Date(COMB$Date, "%d/%m/%Y %H:%M:%S")

# CONVERTING DATATYPE OF ZONES
COMB$Zones<-as.factor(COMB$Zones)
```

We have converted the datatypes, but lets conform that they have changed, by using str() function.

Hide

```
# CHECKING THE DATATYPE OF VARIABLES AFTER CONVERTING
str(COMB)
```

```
tibble [1,170 x 17] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ State                           : chr [1:1170] "Andaman & Nicobar Islands" "Andh
ra Pradesh" "Andhra Pradesh" "Andhra Pradesh" ...
 $ NumPrimaryHealthCenters_HMIS    : num [1:1170] 27 1417 1417 1417 1417 ...
 $ NumCommunityHealthCenters_HMIS  : num [1:1170] 4 198 198 198 198 198 198 198 198
198 ...
 $ NumSubDistrictHospitals_HMIS    : num [1:1170] NA 31 31 31 31 31 31 31 31 31 ...
 $ NumDistrictHospitals_HMIS       : num [1:1170] 3 20 20 20 20 20 20 20 20 20 ...
 $ TotalPublicHealthFacilities_HMIS: num [1:1170] 34 1666 1666 1666 1666 ...
 $ NumPublicBeds_HMIS              : num [1:1170] 1246 60799 60799 60799 60799 ...
 $ NumRuralHospitals_NHP18         : num [1:1170] 27 193 193 193 193 193 193 193 19
3 193 ...
 $ NumRuralBeds_NHP18              : num [1:1170] 575 6480 6480 6480 6480 6480 6480
6480 6480 6480 ...
 $ NumUrbanHospitals_NHP18         : num [1:1170] 3 65 65 65 65 65 65 65 65 65 ...
 $ NumUrbanBeds_NHP18              : num [1:1170] 500 16658 16658 16658 16658 ...
 $ Total_Beds                      : num [1:1170] 2419 87527 87527 87527 87527 ...
 $ Date                            : Date[1:1170], format: NA NA NA NA ...
 $ TotalSamples                    : num [1:1170] NA 1800 6374 6958 6958 ...
 $ Negative                        : num [1:1170] NA 1175 6009 6577 6553 ...
 $ Positive                        : num [1:1170] NA 132 365 381 405 432 473 525 53
4 572 ...
 $ Zones                           : Factor w/ 3 levels "1","2","3": NA 1 2 2 2 2 2
2 2 2 ...
```

Now, the last thing is to label and order the factor variable i.e. Zones. We can do that using as.factor() function.

Hide

```
# LABELLING AND ORDERING ZONES FACTOR
COMB$Zones<-factor(COMB$Zones,levels=c("1","2","3"),labels=c("Green Zone","Orange Z
one","Red Zone"),ordered=TRUE)

# THE FINAL CHECK,BEFORE MOVING TO TIDYING UP
lapply(COMB,class)
```

```
$State
[1] "character"

$NumPrimaryHealthCenters_HMIS
[1] "numeric"

$NumCommunityHealthCenters_HMIS
[1] "numeric"

$NumSubDistrictHospitals_HMIS
[1] "numeric"

$NumDistrictHospitals_HMIS
[1] "numeric"

$TotalPublicHealthFacilities_HMIS
[1] "numeric"

$NumPublicBeds_HMIS
[1] "numeric"

$NumRuralHospitals_NHP18
[1] "numeric"

$NumRuralBeds_NHP18
[1] "numeric"

$NumUrbanHospitals_NHP18
[1] "numeric"

$NumUrbanBeds_NHP18
[1] "numeric"

$Total_Beds
[1] "numeric"

$Date
[1] "Date"

$TotalSamples
[1] "numeric"

$Negative
[1] "numeric"

$Positive
[1] "numeric"

$Zones
[1] "ordered" "factor"
```

TIDY AND MANIPULATE 1

In this step, we will try to tidy the data up. But first let’s understand how tidy dataset looks- In brief, there are three interrelated rules which make a dataset tidy (Hadley Wickham and Grolemund (2016)). In tidy data: **1)**Each variable must have its own column. **2)**Each observation must have its own row. **3)**Each value must have its own cell.

Our dataset COMB2 seems untidy as it is wide formatted, so we use gather() function to make it long formatted. In our dataset, we have different category of hospital beds namely NumPrimaryHealthCenters_HMIS,NumCommunityHealthCenters_HMIS,NumSubDistrictHospitals_HM
district
hospitals,NumDistrictHospitals_HMIS,TotalPublicHealthFacilities_HMIS,NumPublicBeds_HMIS,NumRu
and NumUrbanBeds_NHP18. So we will gather all these variables and make a new variable “NO.OF BEDS” and will select all columns through 2:11.

Hide

```
COMB2<-gather(COMB,key="TYPES OF BEDS",value="Total_Beds", c(2:11))
head(COMB2)
```

State <chr>	D... <date>	TotalSamples <dbl>	Negative <dbl>	Positive <dbl>	Zones <ord>
Andaman & Nicobar Islands	<NA>	NA	NA	NA	NA
Andhra Pradesh	<NA>	1800	1175	132	Green Zone
Andhra Pradesh	<NA>	6374	6009	365	Orange Zone
Andhra Pradesh	<NA>	6958	6577	381	Orange Zone
Andhra Pradesh	<NA>	6958	6553	405	Orange Zone
Andhra Pradesh	<NA>	8755	8323	432	Orange Zone

6 rows | 1-6 of 8 columns

Now, our dataset looks much better, but wait lets remove some irrelevant variables by subsetting. After subsetting, our dataset will have all important columns and will be more presentable.

Hide

```
COMB2<-COMB2[ ,-c(2,6)]

head(COMB2)
```

State <chr>	TotalSamples <dbl>	Negative <dbl>	Positive <dbl>	TYPES OF BEDS <chr>
Andaman & Nicobar Islands	NA	NA	NA	NumPrimaryHealt
Andhra Pradesh	1800	1175	132	NumPrimaryHealt

State	TotalSamples	Negative	Positive	TYPES OF BEDS
<chr>	<dbl>	<dbl>	<dbl>	<chr>
Andhra Pradesh	6374	6009	365	NumPrimaryHealthC
Andhra Pradesh	6958	6577	381	NumPrimaryHealthC
Andhra Pradesh	6958	6553	405	NumPrimaryHealthC
Andhra Pradesh	8755	8323	432	NumPrimaryHealthC

6 rows

TIDY AND MANIPULATE 2

This is the step, where we will create a new variable using mutate() function under package dplyr. The best thing about the mutate() function is that it keeps the existing columns intact and add the new column. So, here we will create the column Ratio, by dividing Total_samples and Total_beds.This new created column will tell us about the number of beds for a person.

Hide

```
COMB2<-mutate(COMB2,Ratio=COMB2$TotalSamples/COMB2$Total_Beds)

df<-COMB2[ ,c(2,6,7)]

head(df)
```

TotalSamples	Total_Beds	Ratio
<dbl>	<dbl>	<dbl>
NA	27	NA
1800	1417	1.270289
6374	1417	4.498236
6958	1417	4.910374
6958	1417	4.910374
8755	1417	6.178546

6 rows

SCAN 1

Now comes the part, where we will see if we have NAs in our dataset or not. For that, first, we will use (sum(is.na())) to find total number of missing values and will also find the columns containing NAs.Secondly, I am planning to impute the NAsb but before that we see the percentage of NAs and Total observations.

Hide

```
# SUM OF MISSING PLACES

sum(is.na(df))
```

```
[1] 403
```

Let us check the columns with missing values in it and lets subset them. Here, I have created a new function.na to check the missing values and then used lapply().

Hide

```
# LIST OF COLUMNS CONTAINING MISSING VALUES

function.na <-  function(x) sum(is.na(x))

lapply(df, function.na)
```

```
$TotalSamples
[1] 60

$Total_Beds
[1] 143

$Ratio
[1] 200
```

After looking above columns, I think, we should impute the NAs of TotalSamples and Total_Beds. But before imputing lets check the proportion of NAs to number of observations.

Hide

```
# CHECKING THE PROPORTIONS

sum(is.na(df)) / nrow(df) *100
```

```
[1] 3.44
```

If we eye on the result of the above coding, we can see the percentage is 3.4, which is very insignificant to disturb our intrepretation.So, lets just change all the NAs to zero(0).

Hide

```
# IMPUTING ZEROS FOR NAs

df <- impute(df, 0)

"Why not check once again sum(is.na(df))."
```

```
[1] "Why not check once again sum(is.na(df))."
```

Hide

```
sum(is.na(df))
```

```
[1] 0
```

SCAN 2

This is the step, where we will detect the outliers in our df table. To detect all the outliers of each variable of df, we will use z scores under the library(outliers) and the boxplots.

a. OUTLIERS OF TOTALSAMPLES

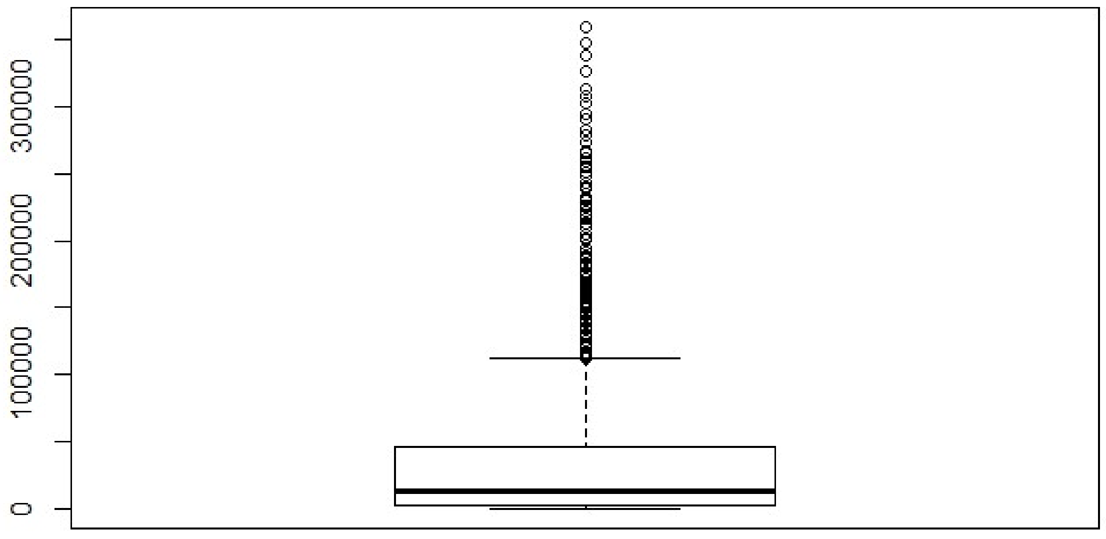
Hide

```
## DETECTING OUTLIERS OF TOTALSAMPLES
df_z<-df$TotalSamples %>% scores(type="z")
df_z %>% summary()
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  -0.66   -0.61   -0.43    0.00    0.13    5.46 
```

Hide

```
boxplot(df$TotalSamples)
```



Hide

```
Totalsample_outliers <- boxplot(df$TotalSamples, plot = FALSE)$out
length(Totalsample_outliers)
```

```
[1] 1260
```

b. OUTLIERS OF TOTAL_BEDS

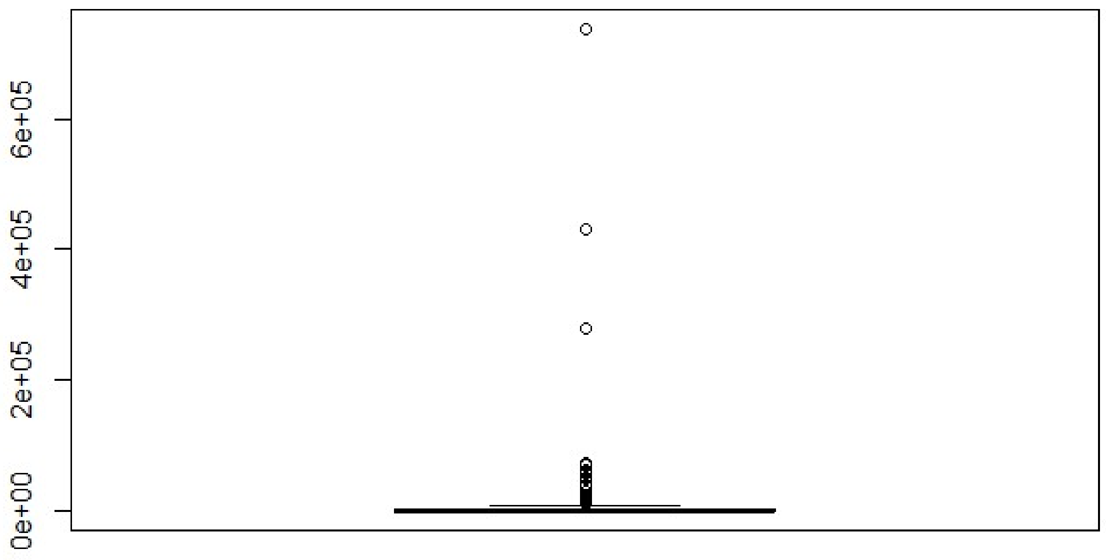
Hide

```
## DETECTING OUTLIERS OF TOTALBEDS
df_y<-df$Total_Beds %>% scores(type="z")
df_y %>% summary()
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   -0.4    -0.4    -0.3     0.0    -0.2    48.2 
```

Hide

```
boxplot(df$Total_Beds)
```



Hide

```
TotalBeds_outliers <- boxplot(df$Total_Beds, plot = FALSE)$out
length(TotalBeds_outliers)
```

```
[1] 1808
```

c. OUTLIERS OF RATIO

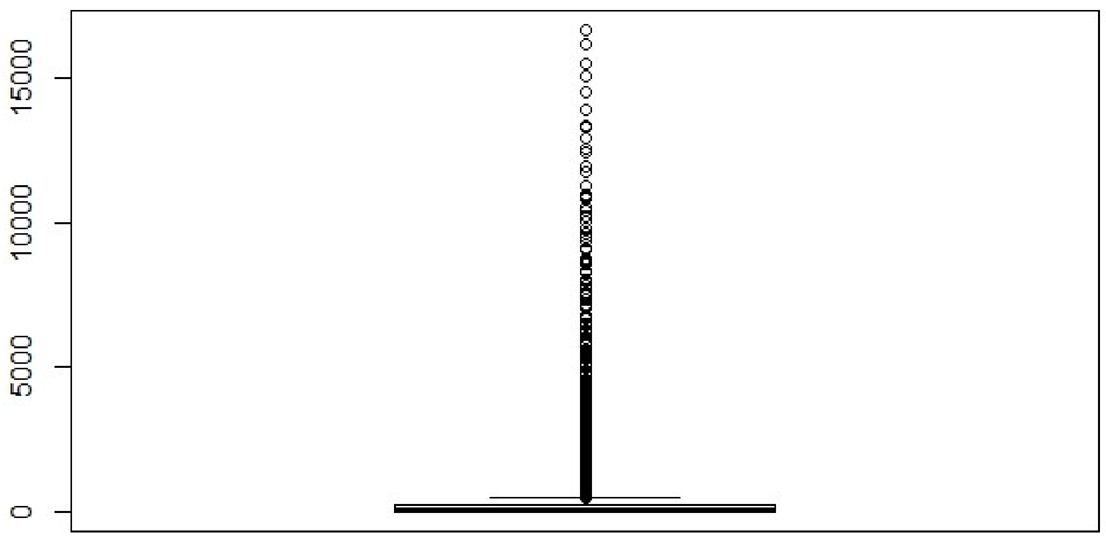
Hide

```
## DETECTING OUTLIERS OF RATIO
df_x<-df$Ratio %>% scores(type="z")
df_x %>% summary()
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
     NA      NA      NA     NaN      NA      NA   11700 
```

Hide

```
boxplot(df$Ratio)
```



Hide

```


Ratio_outliers <- boxplot(df$Ratio, plot = FALSE)$out
length(Ratio_outliers)
```

```
[1] 1796
```

We can notice that our all three numerical variables have alot of outliers and just removing the outliers would not be smart way. So, to handle all outliers, I planning to do winsorising technique or capping technique, which will convert all outliers to thier nearest neighbours that are not a oulier. And to progress with the planning, I created the new function named capped to implement the handling process.

Hide

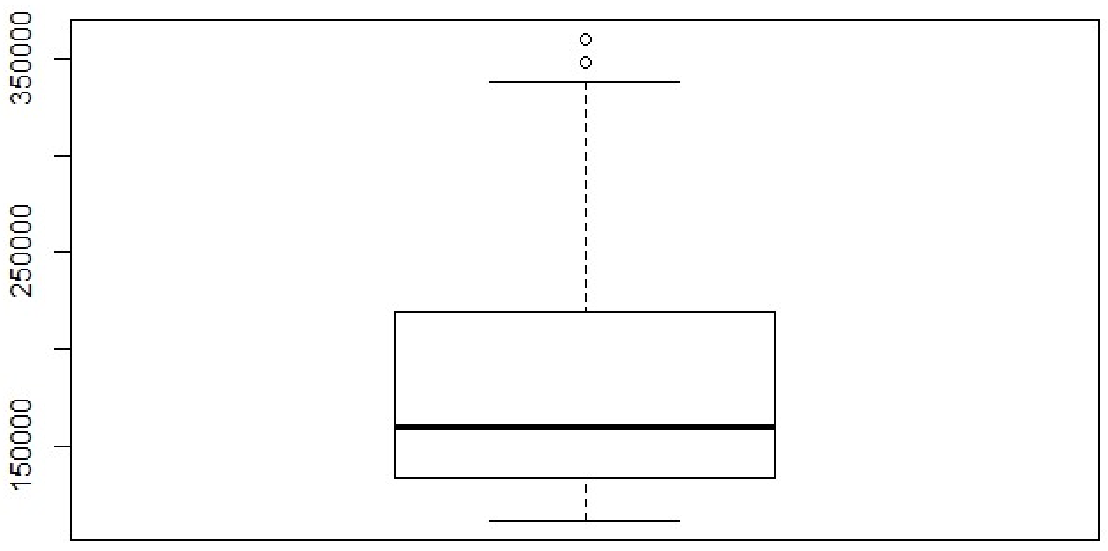
```
capped<- function(x){
  quantiles <- quantile( x, c(.05, 0.25, 0.75, .95 ) )
  x[ x < quantiles[2] - 1.5*IQR(x) ] <- quantiles[1]
  x[ x > quantiles[3] + 1.5*IQR(x) ] <- quantiles[4]
  x
}
```

"APPROACH FOR ALL THE OUTLIERS"

a. Handling outliers of TOTALSAMPLES

Hide

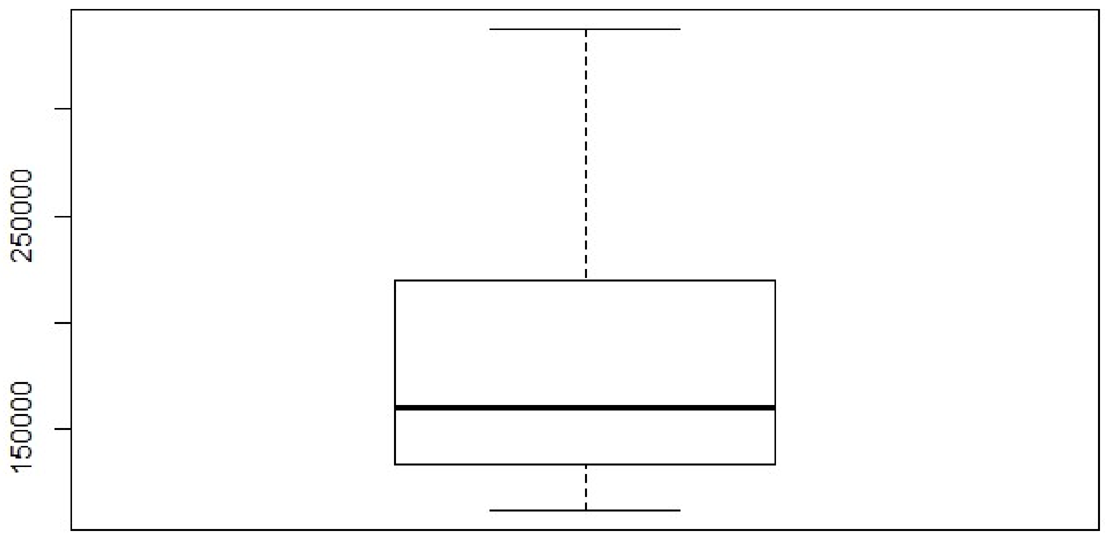
```
TotalBeds_outliers  <- capped(Totalsample_outliers)
boxplot(Totalsample_outliers)
```



b. Handling outliers of TOTALBEDS

Hide

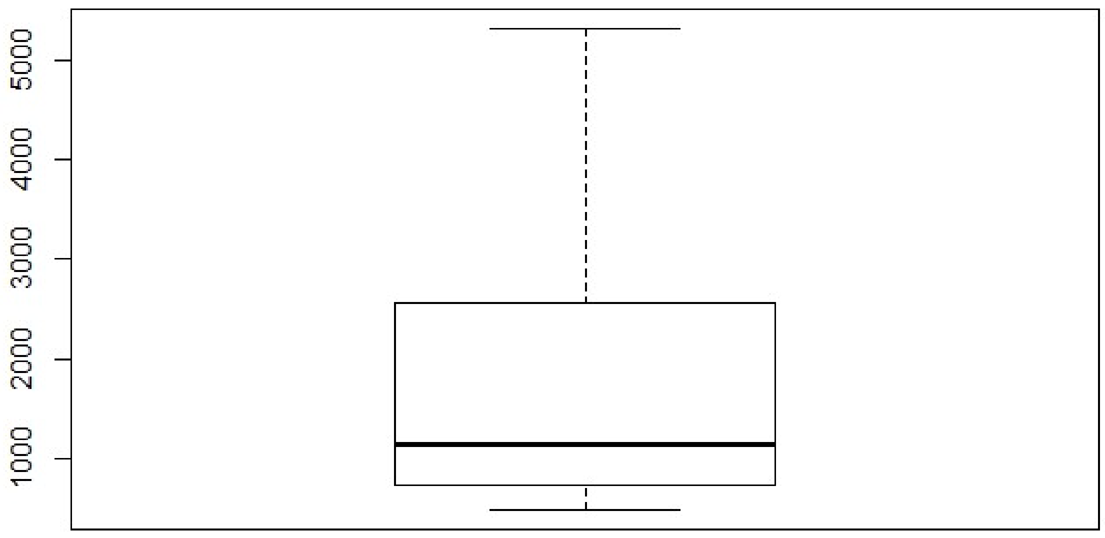
```
TotalBeds_outliers  <- capped(TotalBeds_outliers)
boxplot(TotalBeds_outliers)
```



c. Handling outliers of RATIO

Hide

```
Ratio_outliers  <- capped(Ratio_outliers)
boxplot(Ratio_outliers)
```

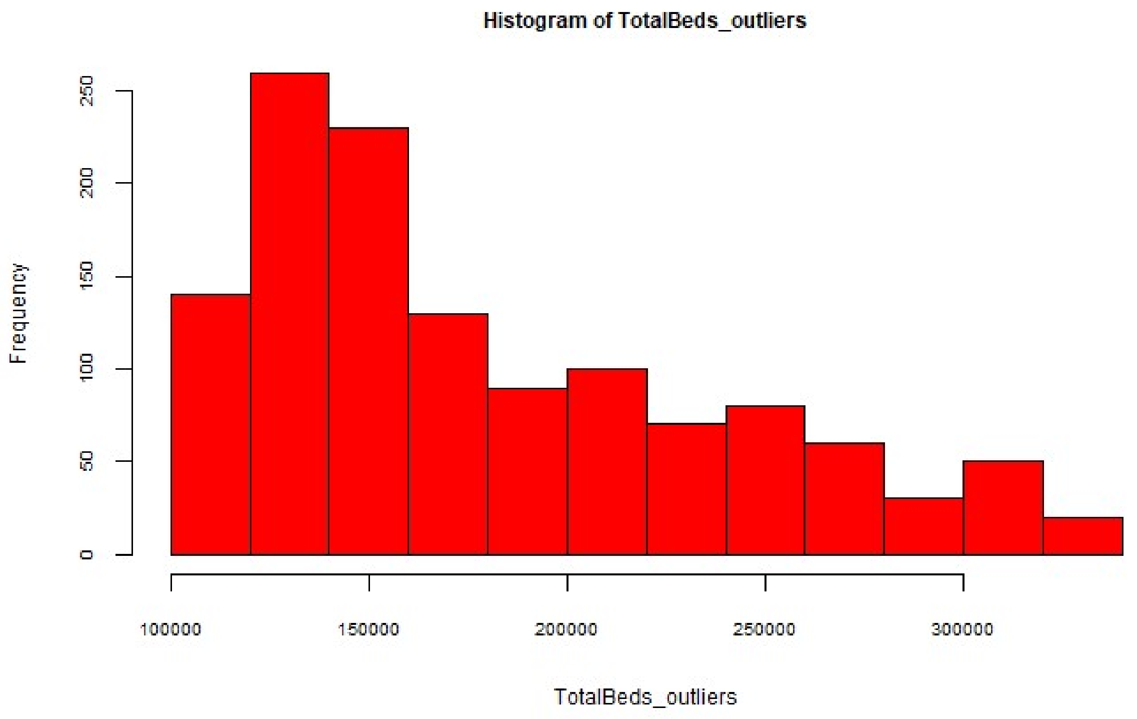


TRANSFORM

This is the last step, where we have to transform one of variable, I am choosing TotalBeds_outliers, which had outliers but are now converted to the nearest non-outlier value. Let's first check the histogram of this variable through hist() function.

Hide

```
#HISTOGRAM OF TOTALBEDS(CONTAINING NO OUTLIERS)
hist(TotalBeds_outliers,border="black",col="red",cex.main=0.75,cex.axis=0.6,cex.lab
=0.75)
```



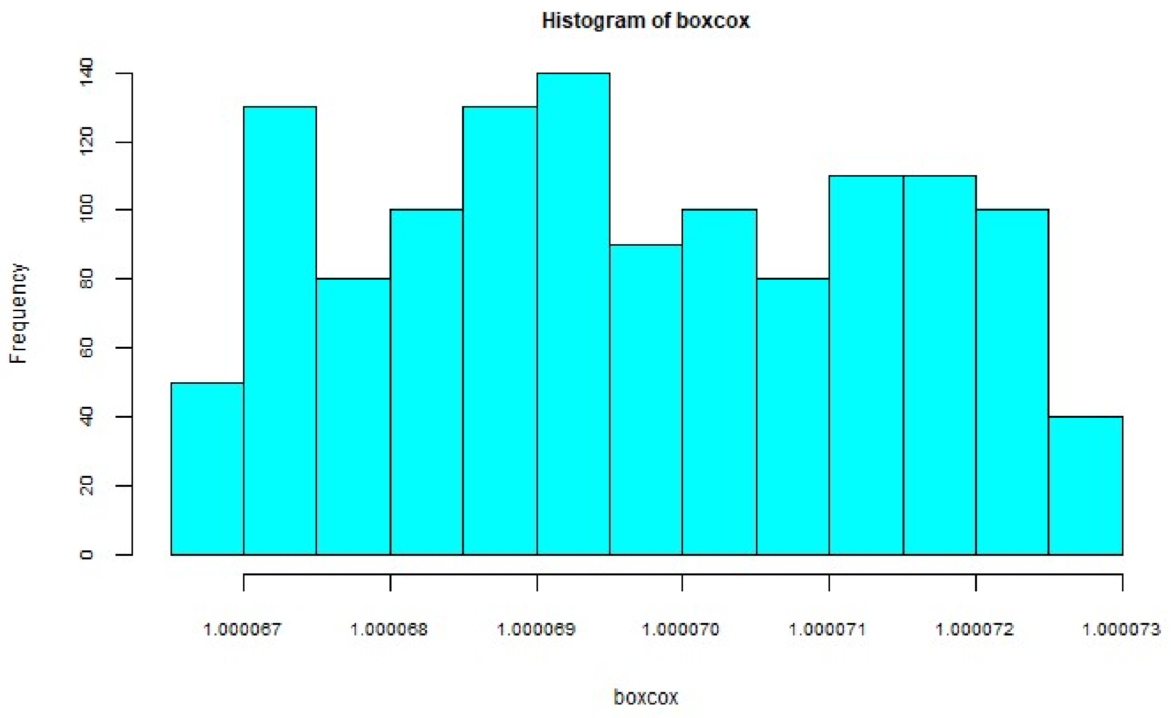
There are so many ways to transform the dataset, but out of all I prefer the Boxcox() function to transform the data.

Hide

```
# CODING FOR BOXCOX
boxcox<- BoxCox(TotalBeds_outliers,lambda = "auto")
```

Hide

```
# HISTOGRAM AFTER APPLYING BOXCOX()FUNCTION
boxcox<- BoxCox(TotalBeds_outliers,lambda = "auto")
hist(boxcox,border="black",col="cyan",cex.main=0.75,cex.axis=0.6,cex.lab=0.75)
```



Now, the histogram of Total beds looks much closer to the normal distribution after the transformation.

FINDINGS OF THE REPORT

If we look at the the mutated column Ratio, we can see that there is at least one bed for every patient, which is a very good situation. These data is was last updated in march 2020, that time the corona was not in its peak and India had 6000(approx) corona cases. But now, things are very unstable for India as they have crossed 2,00,000 cases and converting trains and hotels into a temporary hospitals.

REFERENCES

1)https://www.techopedia.com/definition/33757/kaggle (https://www.techopedia.com/definition/33757/kaggle) 2)http://rare-phoenix-161610.appspot.com/secured/Module_04.html (http://rare-phoenix-161610.appspot.com/secured/Module_04.html) 3)http://rare-phoenix-161610.appspot.com/secured/Module_07.html#box-cox_transformation (http://rare-phoenix-161610.appspot.com/secured/Module_07.html#box-cox_transformation) 4)https://www.worldometers.info/coronavirus/ (https://www.worldometers.info/coronavirus/)