```
getwd()

setwd("C:\\Users\\Swapnil bandekar\\Downloads\\Swapnil\\Data Analytics\\My
Work\\R\\Datasets")


Goodbad <- read.csv("GOODBAD - Copy.csv" , stringsAsFactors = TRUE )

### Description of the Dataset

## 1. Check_Account_Status

## A11 = ...< 0
## A12 = 0 <= ... < 200
## A13 = ... >200
## A14 = no checking account

## 2. Duration : in months

## 3. Credit History

## A30 = NO credits taken / all credits paid back duly
## A31 = all credits at this bank paid back duly
## A32 = existing credits paid back duly till now
## A33 = delay in paying off in the past
## A34 = critical account / other credits existing ( not at this bank )


## 4. Purpose

## A40 = Car (new)
## A41 = Car (used)
## A42 = furniture / equipment
## A43 = radio / television
## A44 = domestic appliances
## A45 = repairs
## A46 = education
## A47 = (vacation - does not exist?)
## A48 = retraining
## A49 = business
## A410 = others

## 5. Amount : Credit Amount

## 6. Age

## 7. GoodBad ( Target Variable )

## 1 : Good
## 0 : Bad
```

```
View(Goodbad)

summary(Goodbad)

# Duration col : Mean > Median => positively skewed


colSums( is.na(Goodbad))

dim(Goodbad)

boxplot(Goodbad$Duration)

## Outlier : value that is significantly different from the rest of the data


quantile(Goodbad$Duration)

hist(Goodbad$Duration)

plot(density(Goodbad$Duration))

new = quantile(Goodbad$Duration , p = c(1:100)/100)

new = quantile(Goodbad$Duration , seq(0.99,1,0.001))

new

Goodbad[Goodbad$Duration > 60, ]


### Identifying an outlier ( Numeric )

## > 99% or < 1% as an outlier
## mean +/- SD


### Treating the outlier

## Delete
## mean (take average of col and replace it with average) (this is global average)
## local mean (take average category wise as per other criteria's in outlier row )
(filter the data , take average and replace it)
## client (client will tell the value)
## capping (replacing the outlier with the value at 99% or 1%) (capping is good as
it maintains the extremities of the data)
## Regression
```

```
### Outlier Treatment Dependencies

## Size of the data
## Priority of variables
## Time
## Trial and error



summary(Goodbad$Amount)

boxplot(Goodbad$Amount)

quantile(Goodbad$Amount)

hist(Goodbad$Amount, labels = TRUE)

plot(density(Goodbad$Amount))

new = quantile(Goodbad$Amount , p = c(1:100)/100)

new = quantile(Goodbad$Amount , seq(0.99,1,0.001))

new

Goodbad[Goodbad$Amount > 15000,]

nrow(Goodbad[Goodbad$Amount > 15000,])


## Cross Tabulation ( like excel pivot table )

table( Goodbad$GoodBad , Goodbad$CreditHistory )

names(Goodbad)

table( Goodbad$GoodBad , Goodbad$Check_Account_Status) / nrow(Goodbad)


library(Hmisc)

describe(Goodbad)

# gives the summary statistics


### Data Preparation
```

```
dim(Goodbad)

index <- which(Goodbad$Duration>61)

index

length(index)

Goodbad$Duration[index]

Goodbad <- Goodbad[-index,]

# -index : deleting the outlier row

dim(Goodbad)

X = boxplot(Goodbad$Amount)

str(X)

list <- X$out

list

# out is created after running the boxplot

length(list)

index1 <- which(Goodbad$Amount %in% list)

index1

length(index1)

# %in% gives row index of the observations stored in list from Goodbad$Amount col

## Shortlist the outliers from the dataset and replace

Goodbad$Amount[index1]

summary(Goodbad$Amount)

## na.rm = TRUE => making sure missing values are removed before calculating the
mean

mean_sw <- mean(Goodbad$Amount , na.rm = TRUE)
```

```
Goodbad$Amount[index1] <- mean_sw

# replacing the outlier with the mean


colSums(is.na(Goodbad))


Goodbad$Age[is.na(Goodbad$Age)] <- mean(Goodbad$Age , na.rm = TRUE)

# replacing the missing value with the mean


## na.omit => to omit the missing value


## Using R package for dealing missing values


install.packages("randomForest")


library(randomForest)


Goodbad$Age <- na.roughfix(Goodbad$Age)

Goodbad$Age

summary(Goodbad$Age)

summary(Goodbad)

# na.roughfix : customize replacement of missing value ; each value will be treated
differently and replaced with regression technique


Goodbad$GoodBad <- ifelse(Goodbad$GoodBad == -1 , 1 , Goodbad$GoodBad )

Goodbad$GoodBad <- ifelse(Goodbad$GoodBad ==1 , 1 , 0 )

# ifelse statement is same as if statement in excel
# ifelse ( condition , value if true , value if false )


## I can't use Qualitative ( charater ) values in my Statistical Model

## I have to transfrom them to Quantitative ( numeric ) values
```

```
## Qualitative to Quantitative Transformation


Goodbad$Check_Account_Status_new <- with( Goodbad , ifelse(
Goodbad$Check_Account_Status == "A11" , 1 ,
                                                        ifelse(
Goodbad$Check_Account_Status == "A12" , 2 ,
                                                                ifelse(
Goodbad$Check_Account_Status == "A13" , 3 , 4 ))))

head(Goodbad$Check_Account_Status_new)

str(Goodbad$Check_Account_Status_new)

summary(Goodbad$Check_Account_Status_new)

describe(Goodbad$Check_Account_Status_new)


## Creating Dummy Variables

Goodbad$Check_Account_Status_A11 <- ifelse( Goodbad$Check_Account_Status == "A11" ,
1 , 0 )

Goodbad$Check_Account_Status_A12 <- ifelse( Goodbad$Check_Account_Status == "A12" ,
1 , 0 )

Goodbad$Check_Account_Status_A13 <- ifelse( Goodbad$Check_Account_Status == "A13" ,
1 , 0 )

Goodbad$Check_Account_Status_A14 <- ifelse( Goodbad$Check_Account_Status == "A14" ,
1 , 0 )

unique( Goodbad$Check_Account_Status_A11 )

# If there are "N" variables than I have to create "N-1" dummy variables

# There are 100+ packages to create dummy variables


Goodbad1 <- Goodbad

View(Goodbad1)


## Using factor and model.matrix combination to create dummy variables


X <- factor(Goodbad1$Check_Account_Status)
```

```
class(Goodbad1$Check_Account_Status)

typeof(Goodbad1$Check_Account_Status)

Dummies <- model.matrix(~X)

View(Dummies)

class(Dummies)

Y <- data.frame(Dummies)

dim(Y)

Goodbad_New <- cbind(Goodbad , Y)

View(Goodbad_New)

install.packages("dummies")

library(dummies)

Goodbad2 <- dummy(Goodbad$CreditHistory)

View(Goodbad2)

Goodbad_New2 <- cbind(Goodbad , Goodbad2)

View(Goodbad_New2)


cor(Goodbad$Amount , Goodbad$Duration)

# cor : to find correlation between 2 continuous variables ( variables should be
numeric only )

## library for correlation

library(corrgram)


corrgram(Goodbad)

corrgram( Goodbad , order = TRUE , lower.panel = panel.shade , upper.panel =
panel.pie , main = "corrgram")

names(Goodbad)


library(Information)
```

```
res = create_infotables( data = Goodbad , y = "GoodBad")

res

library(InformationValue)

WOE( X = Goodbad$Check_Account_Status , Y = Goodbad$GoodBad )

options(scipen=999 , digits = 2)

WOETable(X = Goodbad$Check_Account_Status , Y = Goodbad$GoodBad)

options(scipen=999 , digits = 4)

IV(X = Goodbad$Check_Account_Status , Y = Goodbad$GoodBad)



#### ggplot Case Study


## Dataset : Presidential and Economy (Present in ggplot library)

library(ggplot2)

head(economics)
head(presidential)

presidential <- presidential[-c(1:3),]

head(presidential)

# Taking out the first three columns from the presidential dataset as the dates
don't match


## Doing Data Manipulation

Q <- ggplot( economics , aes( x = date , y = unemploy ))

Q

# Setting up the aesthetic map

Q + geom_line()

unemp = Q + geom_line() + xlab(" ") + ylab("No of unemployed(1000's)")

unemp
```

```
# making a partial plot

# redering a line from economice dataset using date and unemploy column

# labels on x-axis and y-axis using xlab and ylab commands and storing the plot in
an object unemp

head(economics)

yrng <- range(economics$unemploy)


# defining x range and y range which will be used later to create a rectangle

# geom_rect(Ds , aes( xmin , xmax , ymin , ymax ))

# Need to create a rectangle. for creating a rectangle I have to define 4 things :
x min, xmax , y min , ymax

# rect based on unemploy column in the economics dataset

# adding a layer to this rectangle from presidential table

# xmin and xmax from presidential table

# ymin and ymax from economics table

# fill by party from presidential table


unemp + geom_rect( data = presidential , aes( xmin = start , xmax = end , NULL ,
NULL ,  fill = party ) , ymin = yrng[1] , ymax = yrng[2] , alpha = 0.2 )


# ymin and ymax comes from a different dataset . Hence , parsed as "NULL" in "aes"
fn

# alpha = 0.2 => is used for changing the transparency level , 0.2 value gives the
80% transparency
```