Data Preparation

Chapter 1 : Handling Missing Values

1. Delete
   - data loss
   - bias

2. Replace with a constant such as average

```r
library(mice)
```

```
## Loading required package: lattice

##
## Attaching package: 'mice'

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```r
library(VIM)
```

```
## Loading required package: colorspace

## Loading required package: grid

## Loading required package: data.table

## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##           Please use the package to use the new (and old) GUI.

## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

```r
data <- read.csv("vehicleMissing_Value.csv",header = T,sep = ",")
str(data)
```

```
## 'data.frame':    1624 obs. of  7 variables:
##  $ vehicle: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ fm     : int  0 10 15 0 13 21 11 5 8 1 ...
##  $ Mileage: int  863 4644 16330 13 22537 40931 34762 11051 7003 11 ...
##  $ lh     : num  1.1 2.4 4.2 1 4.5 3.1 0.7 2.9 3.4 0.7 ...
##  $ lc     : num  66.3 233 325.1 66.6 328.7 ...
##  $ mc     : num  697 120 175 0 175 ...
##  $ State  : Factor w/ 50 levels "AK","AL","AR",..: 25 5 48 37 4 9 18 10 47 38 ...
```

```r
summary(data)
```

```
##     vehicle             fm             Mileage             lh
```

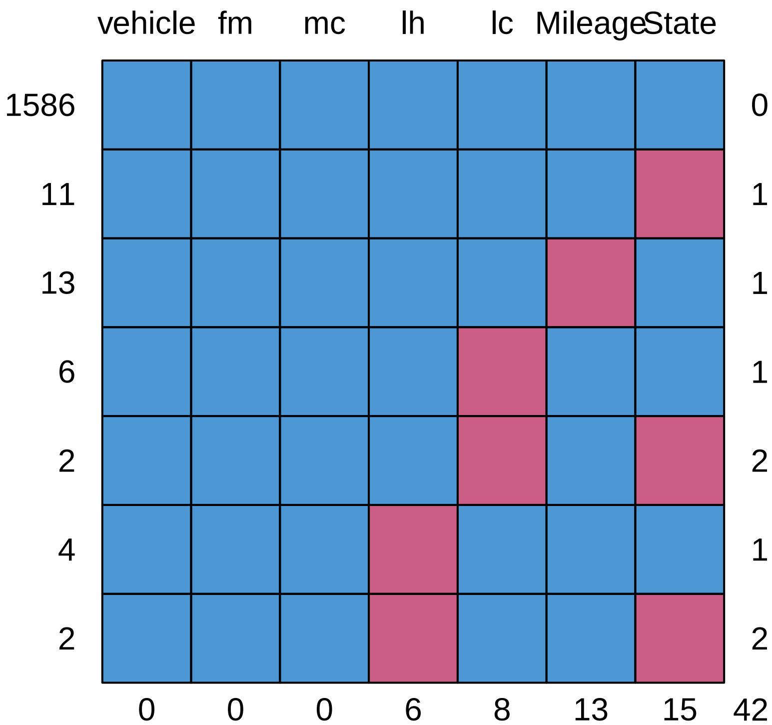
```
##  Min.   :   1.0   Min.   :-1.000   Min.   :    1   Min.   : 0.000
##  1st Qu.: 406.8   1st Qu.: 4.000   1st Qu.: 5778   1st Qu.: 1.500
##  Median : 812.5   Median :10.000   Median :17000   Median : 2.600
##  Mean   : 812.5   Mean   : 9.414   Mean   :20559   Mean   : 3.294
##  3rd Qu.:1218.2   3rd Qu.:14.000   3rd Qu.:30060   3rd Qu.: 4.300
##  Max.   :1624.0   Max.   :23.000   Max.   :99983   Max.   :35.200
##                                    NA's   :13      NA's   :6
##        lc               mc              State
##  Min.   :   0.0   Min.   :   0.0   TX     :290
##  1st Qu.: 106.5   1st Qu.: 119.7   CA     :199
##  Median : 195.4   Median : 119.7   FL     :167
##  Mean   : 242.8   Mean   : 179.4   GA     : 75
##  3rd Qu.: 317.8   3rd Qu.: 175.5   AZ     : 61
##  Max.   :3234.4   Max.   :3891.1   (Other):817
##  NA's   :8                         NA's   : 15
```

let's calculate what percentage data is missing

```
p <- function(x){sum(is.na(x))/length(x)*100}
apply(data, 2, p)
```

```
##   vehicle        fm   Mileage        lh        lc        mc     State
## 0.0000000 0.0000000 0.8004926 0.3694581 0.4926108 0.0000000 0.9236453
```

```
md.pattern(data)
```




```
##      vehicle fm mc lh lc Mileage State
## 1586       1  1  1  1  1       1     1  0
## 11         1  1  1  1  1       1     0  1
## 13         1  1  1  1  1       0     1  1
```

```
## 6           1  1  1  1  0       1      1  1
## 2           1  1  1  1  0       1      0  2
## 4           1  1  1  0  1       1      1  1
## 2           1  1  1  0  1       1      0  2
##             0  0  0  6  8      13     15 42
```

```r
md.pairs(data)
```

```
## $rr
##           vehicle   fm Mileage   lh   lc   mc State
## vehicle      1624 1624    1611 1618 1616 1624  1609
## fm           1624 1624    1611 1618 1616 1624  1609
## Mileage      1611 1611    1611 1605 1603 1611  1596
## lh           1618 1618    1605 1618 1610 1618  1605
## lc           1616 1616    1603 1610 1616 1616  1603
## mc           1624 1624    1611 1618 1616 1624  1609
## State        1609 1609    1596 1605 1603 1609  1609
##
## $rm
##           vehicle fm Mileage lh lc mc State
## vehicle         0  0      13  6  8  0    15
## fm              0  0      13  6  8  0    15
## Mileage         0  0       0  6  8  0    15
## lh              0  0      13  0  8  0    13
## lc              0  0      13  6  0  0    13
## mc              0  0      13  6  8  0    15
## State           0  0      13  4  6  0     0
##
## $mr
##           vehicle fm Mileage lh lc mc State
## vehicle         0  0       0  0  0  0     0
## fm              0  0       0  0  0  0     0
## Mileage        13 13       0 13 13 13    13
## lh              6  6       6  0  6  6     4
## lc              8  8       8  8  0  8     6
## mc              0  0       0  0  0  0     0
## State          15 15      15 13 13 15     0
##
## $mm
##           vehicle fm Mileage lh lc mc State
## vehicle         0  0       0  0  0  0     0
## fm              0  0       0  0  0  0     0
## Mileage         0  0      13  0  0  0     0
## lh              0  0       0  6  0  0     2
## lc              0  0       0  0  8  0     2
## mc              0  0       0  0  0  0     0
## State           0  0       0  2  2  0    15
```

Impute

```r
impute <- mice(data[,2:7],m=3,seed = 123)
```

```
##
```

```
##  iter imp variable
##   1   1  Mileage  lh  lc  State
##   1   2  Mileage  lh  lc  State
##   1   3  Mileage  lh  lc  State
##   2   1  Mileage  lh  lc  State
##   2   2  Mileage  lh  lc  State
##   2   3  Mileage  lh  lc  State
##   3   1  Mileage  lh  lc  State
##   3   2  Mileage  lh  lc  State
##   3   3  Mileage  lh  lc  State
##   4   1  Mileage  lh  lc  State
##   4   2  Mileage  lh  lc  State
##   4   3  Mileage  lh  lc  State
##   5   1  Mileage  lh  lc  State
##   5   2  Mileage  lh  lc  State
##   5   3  Mileage  lh  lc  State
```

```r
print(impute)
```

```
## Class: mids
## Number of multiple imputations:  3
## Imputation methods:
##        fm   Mileage        lh        lc        mc     State
##        ""     "pmm"     "pmm"     "pmm"        "" "polyreg"
## PredictorMatrix:
##         fm Mileage lh lc mc State
## fm       0       1  1  1  1     1
## Mileage  1       0  1  1  1     1
## lh       1       1  0  1  1     1
## lc       1       1  1  0  1     1
## mc       1       1  1  1  0     1
## State    1       1  1  1  1     0
```

pmm -> means predictive Mean Matching.

polyreg -> basically Multinomial Logistic Regression

```r
i <- impute$imp$Mileage
```

```r
summary(data$Mileage)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       1    5778   17000   20559   30060   99983      13
```

```r
data[20,]
```

```
##    vehicle fm Mileage  lh    lc   mc State
## 20      20  8      NA 1.4 87.42 1.85    NH
```

```r
data[253,]
```

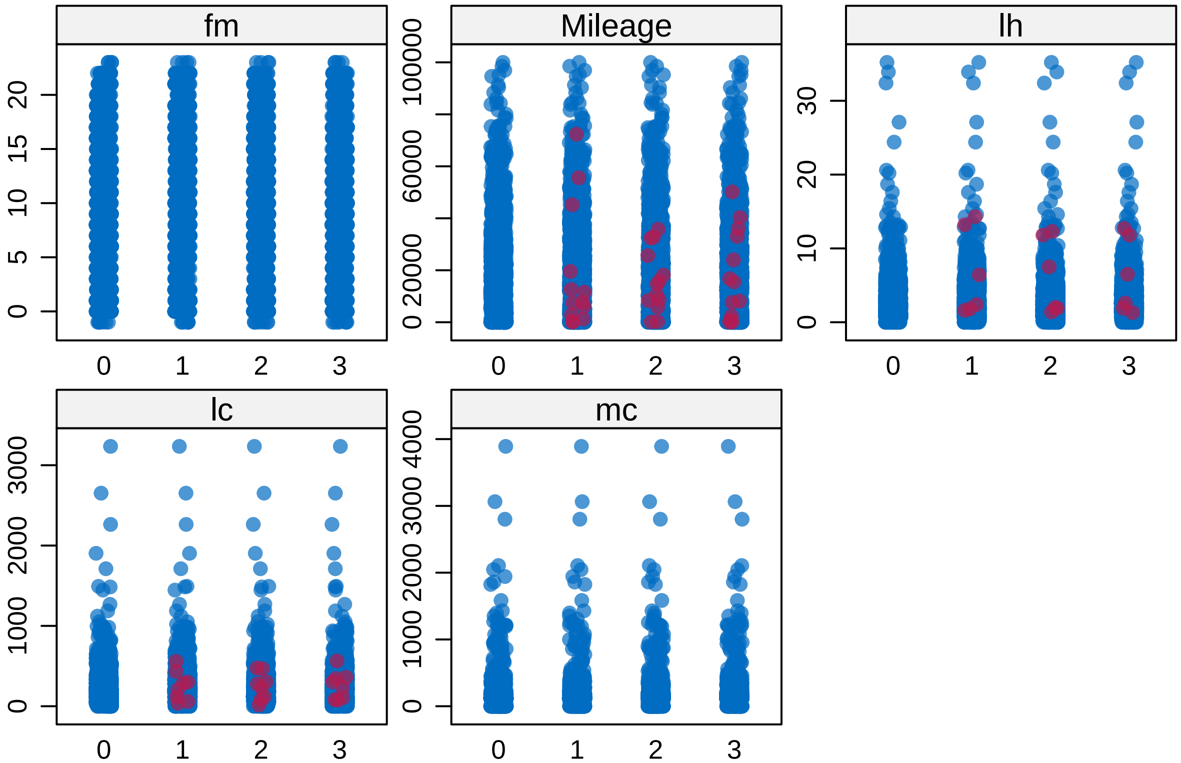
```
##     vehicle fm Mileage  lh    lc     mc State
## 253     253  1      NA 1.4 89.89 119.66    FL
```

Complete data

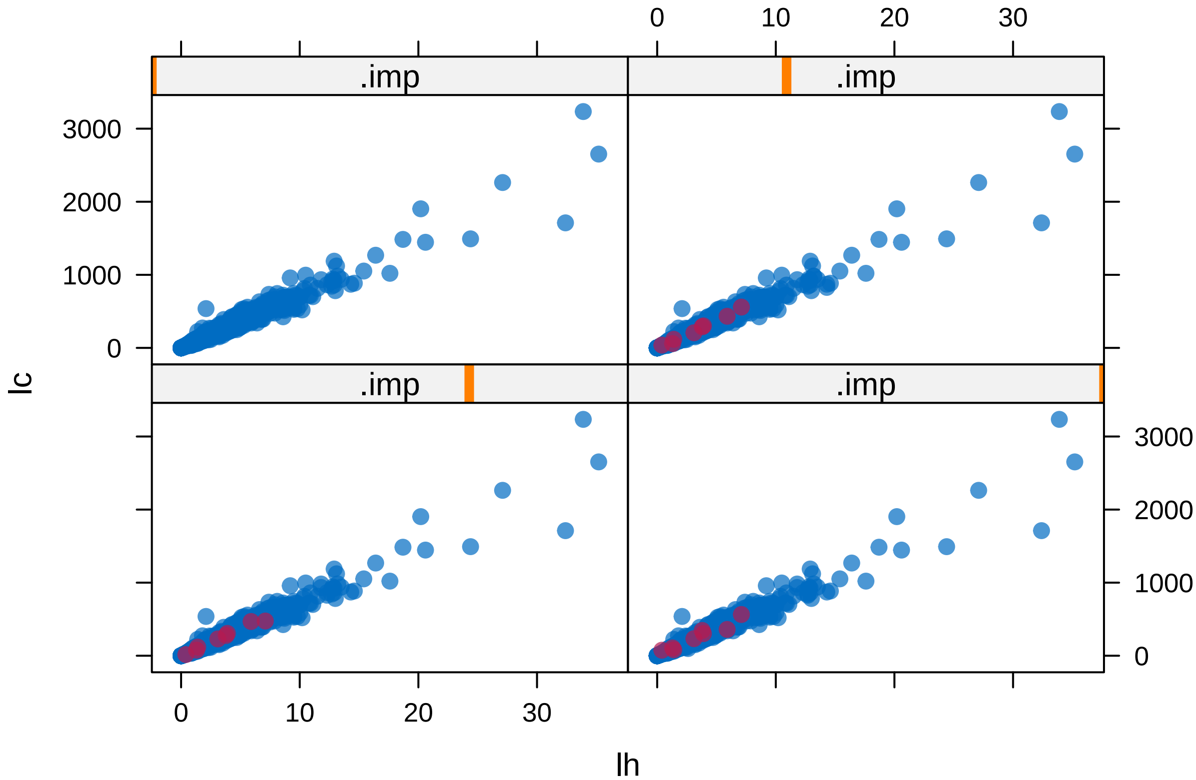
```
newData <- complete(impute,1)
```

Distribution of observed/imputed values

```
stripplot(impute,pch = 20,cex=1.2)
```




```
xyplot(impute,lc ~ lh | .imp, pch = 20,cex = 1.4)
```



from Gourabh Nath's tutorials ## Consider the Problem ### Consider the following vector of Marks of 10 students. ### Math <- 88,95,85,NA,76,69,78,NA,70,68 ### Using the data given above calculate, ### 1. The Mean of the data. ### 2. The Median of the data ### 3. The SD of the data. ### 4. Impute the missing values by mean. ### 5. Impute the missing values by median.

```
Math <- c(88,95,85,NA,76,69,78,NA,70,68)
```

```
mean(Math)
```

```
## [1] NA
```

```
median(Math)
```

```
## [1] NA
```

```
sd(Math)
```

```
## [1] NA
```

calculate the function ignoring the missing data using is.na()

```
Math[is.na(Math)]
```

```
## [1] NA NA
```

```
Math[!is.na(Math)]
```

```
## [1] 88 95 85 76 69 78 70 68
```

```
mean(Math[!is.na(Math)])
```

```
## [1] 78.625
```

```r
sd(Math[!is.na(Math)])
```

```
## [1] 9.88415
```

Missing value imputation

```r
math1 <- Math
math2 <- Math
```

Mean Imputation replace values with mean value

```r
math1[is.na(math1)] <- mean(math1[!is.na(math1)])
```

```r
math1
```

```
##  [1] 88.000 95.000 85.000 78.625 76.000 69.000 78.000 78.625 70.000 68.000
```

Missing data replaced by mean

Median Imputation

```r
math2[is.na(math2)] <- median(math2[!is.na(math2)])
```

Missing values replaced by 77 median

```r
math2
```

```
##  [1] 88 95 85 77 76 69 78 77 70 68
```

```r
x <- 1:10
y <- c(11,12,18,14,17,NA,NA,19,NA,27)
z <- c(19,11,2,14,20,4,9,10,18,1)
w <- c(1,4,7,10,3,5,7,6,6,9)
```

```r
data <- data.frame(x,y,z,w)
data
```

```
##     x  y  z  w
## 1   1 11 19  1
## 2   2 12 11  4
## 3   3 18  2  7
## 4   4 14 14 10
## 5   5 17 20  3
## 6   6 NA  4  5
## 7   7 NA  9  7
## 8   8 19 10  6
## 9   9 NA 18  6
## 10 10 27  1  9
```

Step 1 : Finding the most correlated variable

```
cor(data)
```

```
##            x  y          z          w
## x  1.0000000 NA -0.2736766  0.5029477
## y         NA  1         NA         NA
## z -0.2736766 NA  1.0000000 -0.5276512
## w  0.5029477 NA -0.5276512  1.0000000
```

```
cor(data,use = "complete.obs")
```

```
##            x          y          z          w
## x  1.0000000  0.9088508 -0.4794970  0.5427928
## y  0.9088508  1.0000000 -0.6931033  0.5575189
## z -0.4794970 -0.6931033  1.0000000 -0.6438960
## w  0.5427928  0.5575189 -0.6438960  1.0000000
```

symnum()

```
symnum(cor(data,use="complete.obs"))
```

```
##   x y z w
## x 1
## y * 1
## z . , 1
## w . . , 1
## attr(,"legend")
## [1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
```

Step 2 : Creating an Indicator variable

```
Ind <- function(t){
  x <- dim(length(t))
  x[which(!is.na(t))] - 1
  x[which(is.na(t))] - 0
  return(x)
}
```

```
data$I <- Ind(data$y)
data
```

```
##     x  y  z  w
## 1   1 11 19  1
## 2   2 12 11  4
## 3   3 18  2  7
## 4   4 14 14 10
## 5   5 17 20  3
## 6   6 NA  4  5
## 7   7 NA  9  7
## 8   8 19 10  6
## 9   9 NA 18  6
## 10 10 27  1  9
```

Step 3 : Fitting a linear regression model of Y on X

```r
lm(y ~ x, data=data)
```

```
##
## Call:
## lm(formula = y ~ x, data = data)
##
## Coefficients:
## (Intercept)            x
##       9.743        1.509
```

```r
summary(lm(y ~ x, data=data))
```

```
##
## Call:
## lm(formula = y ~ x, data = data)
##
## Residuals:
##       1       2       3       4       5       8      10
## -0.2523 -0.7613  3.7297 -1.7793 -0.2883 -2.8153  2.1667
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.7432     1.7324   5.624  0.00246 **
## x             1.5090     0.3097   4.872  0.00458 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.467 on 5 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.826,  Adjusted R-squared:  0.7912
## F-statistic: 23.74 on 1 and 5 DF,  p-value: 0.004585
```

y = 9.7432 + 1.509 * x

```
## [1] 2
```

Chapter 2 : Outlier

What is an Outlier?

-> In statistics, an outlier is an observation point that is distant from other observations.

Some of Possible causes of outliers.

1. incorrect Entry

2. Mis-Reporting

3. Sampling Error

4. Exception but True value

Some ways of dealing with outliers ...

1. Discarding

2. Winsorizing

3. Dropping but keeping in records

4. variable transformation

5. Fit different Models

6. Use Non-Parametric Methods

STEP 1 : Outlier Treatment

```
data <- c(sample(x = 1:20,size = 40,replace = T,),65,80)
```

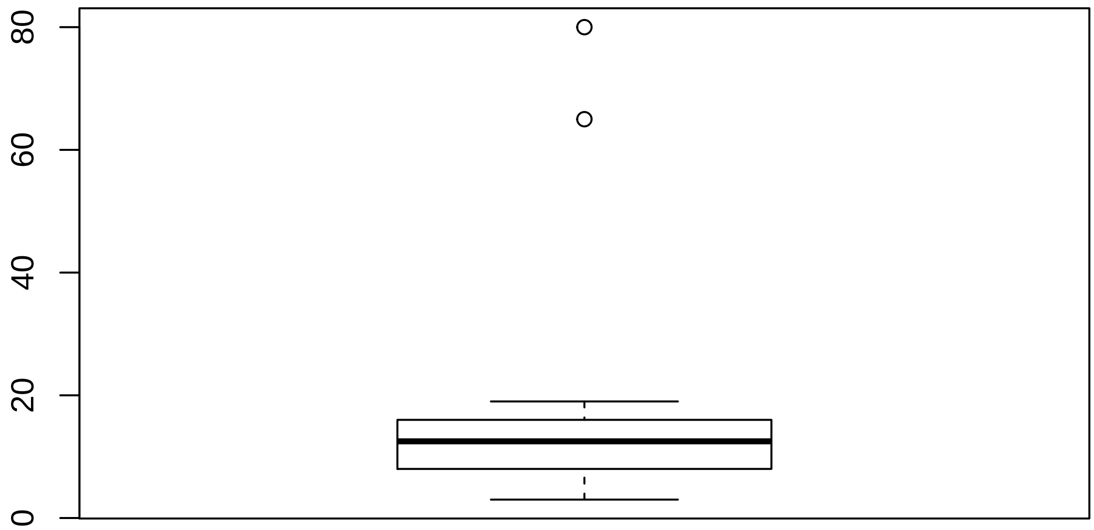
```
data
```

```
##  [1] 18 18  8  8  5 18  6  4  9 16 14 15  8 10 19 19 16 15  8  3 13 11  4
## [24]  3  8 14 16 19 10 13 16  6 15 12  4  7  9  3 13 16 65 80
```

```
summary(data)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    3.00    8.00   12.50   14.14   16.00   80.00
```

```
  boxplot(data)
```



Discarding the outliers from the data set

```r
data1 <- data
length(data1)
```

```
## [1] 42
```

#create a bench mark - if any of the observation falls above this bench mark we are going to delete it from dataset

```r
bench <- 17.75 + 1.5*IQR(data1)
bench
```

```
## [1] 29.75
```

any value above 29.75 we are going to discard it.

this displays outlier

```r
data1[data1 > 29.75]
```

```
## [1] 65 80
```

```r
data1[data1 < 29.75]
```

```
##  [1] 18 18  8  8  5 18  6  4  9 16 14 15  8 10 19 19 16 15  8  3 13 11  4
## [24]  3  8 14 16 19 10 13 16  6 15 12  4  7  9  3 13 16
```

```r
data1 <- data1[data1 < bench]
```

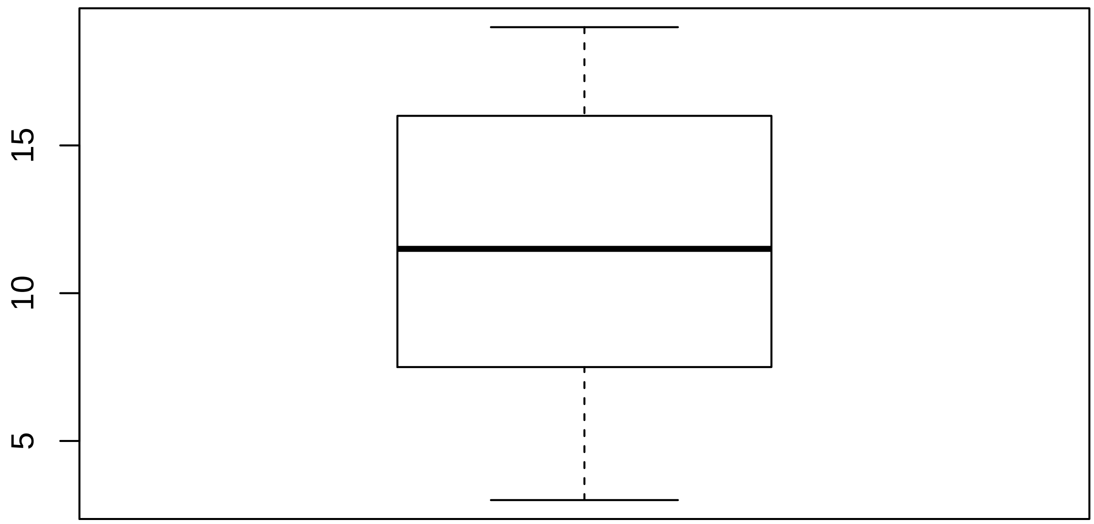
```r
data1
```

```
##  [1] 18 18  8  8  5 18  6  4  9 16 14 15  8 10 19 19 16 15  8  3 13 11  4
## [24]  3  8 14 16 19 10 13 16  6 15 12  4  7  9  3 13 16
```

```r
summary(data1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00    7.75   11.50   11.22   16.00   19.00
```

```r
boxplot(data1)
```



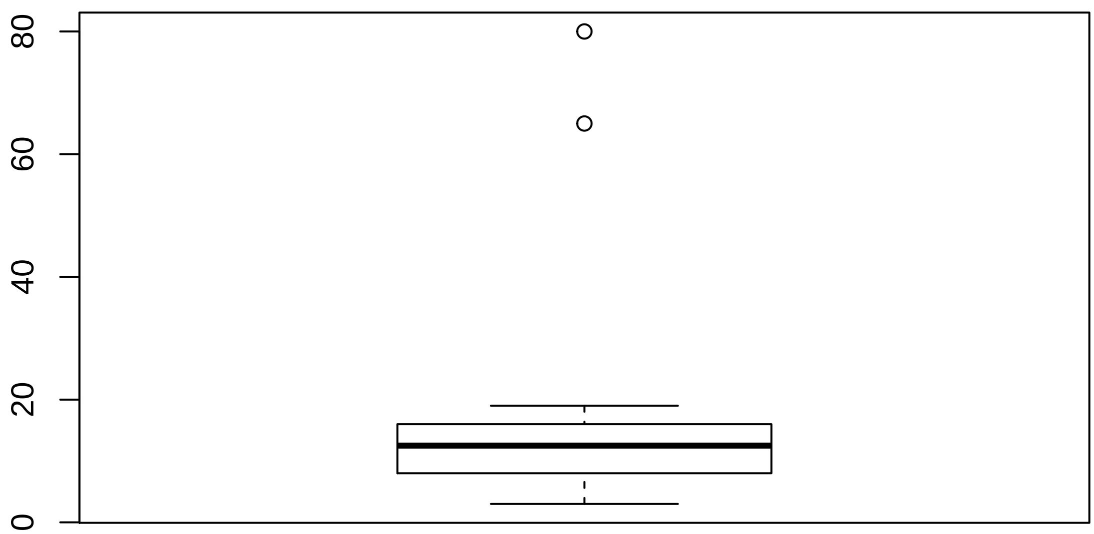
Winsorizing

```
data2 <- data
```

```
summary(data2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00    8.00   12.50   14.14   16.00   80.00
```

```
boxplot(data2)
```



Setting the bench mark

```
data2
```

```
##  [1] 18 18  8  8  5 18  6  4  9 16 14 15  8 10 19 19 16 15  8  3 13 11  4
## [24]  3  8 14 16 19 10 13 16  6 15 12  4  7  9  3 13 16 65 80
```

```
summary(data2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00    8.00   12.50   14.14   16.00   80.00
```

```
bench <- 16.00 + 1.5 * IQR(data2)
```

```r
data2[data2 > bench]
```

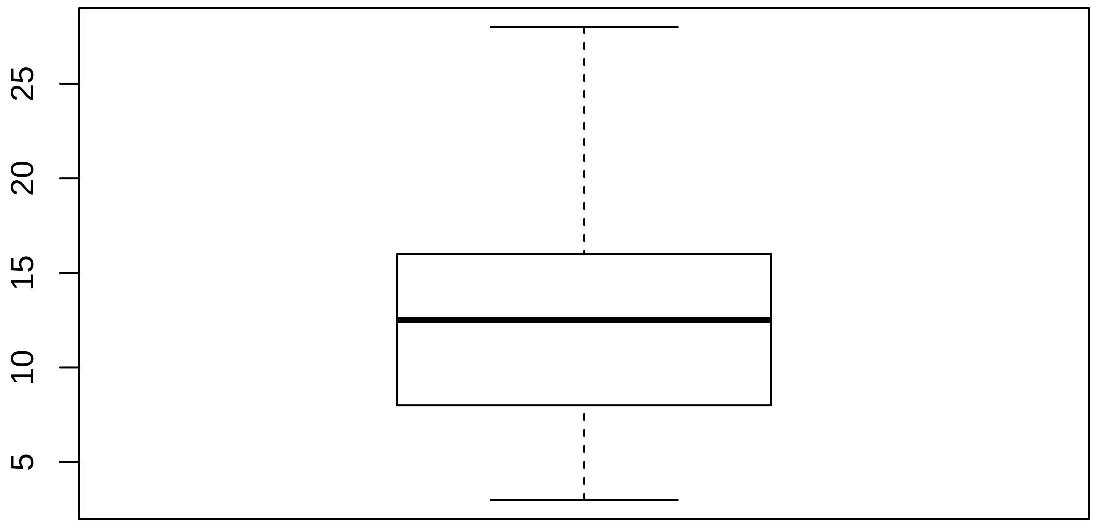
```
## [1] 65 80
```

```r
data2[data2 > bench] <- bench
```

```r
summary(data2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00    8.00   12.50   12.02   16.00   28.00
```

```r
boxplot(data2)
```



```
## [1] 2
```

Chapter 3 : Feature Selection

Too many features

```
-Time Consuming\newline
-low accuracy\newline
-consumes resources
```

SO we are going to use of BORUTA algorithm.

How it works? -> let's say if you have 60 different features in data then for each feature/attribute it creates a shadow attribute/feature and shadow attributes has all the values shuffled, then we create models which includes shadow and original attributes and assess the importance of each variable. so the idea is if attribute is not doing better then shadow attribute in terms of importance and obivously we should not have those kind of attributes in the model because they are not real important.

```r
library(Boruta)
```

```
## Loading required package: ranger
```

```r
library(mlbench)
library(caret)
```

```
## Loading required package: ggplot2
```

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:ranger':
##
##     importance
```

STEP 1 : Read Data

```r
data("Sonar")
```

```r
str(Sonar)
```

```
## 'data.frame':    208 obs. of  61 variables:
##  $ V1   : num  0.02 0.0453 0.0262 0.01 0.0762 0.0286 0.0317 0.0519 0.0223 0.0164 ...
##  $ V2   : num  0.0371 0.0523 0.0582 0.0171 0.0666 0.0453 0.0956 0.0548 0.0375 0.0173 ...
##  $ V3   : num  0.0428 0.0843 0.1099 0.0623 0.0481 ...
##  $ V4   : num  0.0207 0.0689 0.1083 0.0205 0.0394 ...
##  $ V5   : num  0.0954 0.1183 0.0974 0.0205 0.059 ...
##  $ V6   : num  0.0986 0.2583 0.228 0.0368 0.0649 ...
##  $ V7   : num  0.154 0.216 0.243 0.11 0.121 ...
##  $ V8   : num  0.16 0.348 0.377 0.128 0.247 ...
##  $ V9   : num  0.3109 0.3337 0.5598 0.0598 0.3564 ...
##  $ V10  : num  0.211 0.287 0.619 0.126 0.446 ...
##  $ V11  : num  0.1609 0.4918 0.6333 0.0881 0.4152 ...
##  $ V12  : num  0.158 0.655 0.706 0.199 0.395 ...
##  $ V13  : num  0.2238 0.6919 0.5544 0.0184 0.4256 ...
##  $ V14  : num  0.0645 0.7797 0.532 0.2261 0.4135 ...
##  $ V15  : num  0.066 0.746 0.648 0.173 0.453 ...
##  $ V16  : num  0.227 0.944 0.693 0.213 0.533 ...
##  $ V17  : num  0.31 1 0.6759 0.0693 0.7306 ...
##  $ V18  : num  0.3 0.887 0.755 0.228 0.619 ...
##  $ V19  : num  0.508 0.802 0.893 0.406 0.203 ...
##  $ V20  : num  0.48 0.782 0.862 0.397 0.464 ...
##  $ V21  : num  0.578 0.521 0.797 0.274 0.415 ...
##  $ V22  : num  0.507 0.405 0.674 0.369 0.429 ...
##  $ V23  : num  0.433 0.396 0.429 0.556 0.573 ...
##  $ V24  : num  0.555 0.391 0.365 0.485 0.54 ...
##  $ V25  : num  0.671 0.325 0.533 0.314 0.316 ...
##  $ V26  : num  0.641 0.32 0.241 0.533 0.229 ...
##  $ V27  : num  0.71 0.327 0.507 0.526 0.7 ...
##  $ V28  : num  0.808 0.277 0.853 0.252 1 ...
##  $ V29  : num  0.679 0.442 0.604 0.209 0.726 ...
##  $ V30  : num  0.386 0.203 0.851 0.356 0.472 ...
##  $ V31  : num  0.131 0.379 0.851 0.626 0.51 ...
##  $ V32  : num  0.26 0.295 0.504 0.734 0.546 ...
##  $ V33  : num  0.512 0.198 0.186 0.612 0.288 ...
##  $ V34  : num  0.7547 0.2341 0.2709 0.3497 0.0981 ...
##  $ V35  : num  0.854 0.131 0.423 0.395 0.195 ...
##  $ V36  : num  0.851 0.418 0.304 0.301 0.418 ...
##  $ V37  : num  0.669 0.384 0.612 0.541 0.46 ...
##  $ V38  : num  0.61 0.106 0.676 0.881 0.322 ...
##  $ V39  : num  0.494 0.184 0.537 0.986 0.283 ...
##  $ V40  : num  0.274 0.197 0.472 0.917 0.243 ...
##  $ V41  : num  0.051 0.167 0.465 0.612 0.198 ...
##  $ V42  : num  0.2834 0.0583 0.2587 0.5006 0.2444 ...
##  $ V43  : num  0.282 0.14 0.213 0.321 0.185 ...
##  $ V44  : num  0.4256 0.1628 0.2222 0.3202 0.0841 ...
##  $ V45  : num  0.2641 0.0621 0.2111 0.4295 0.0692 ...
##  $ V46  : num  0.1386 0.0203 0.0176 0.3654 0.0528 ...
##  $ V47  : num  0.1051 0.053 0.1348 0.2655 0.0357 ...
##  $ V48  : num  0.1343 0.0742 0.0744 0.1576 0.0085 ...
##  $ V49  : num  0.0383 0.0409 0.013 0.0681 0.023 0.0264 0.0507 0.0285 0.0777 0.0092 ...
##  $ V50  : num  0.0324 0.0061 0.0106 0.0294 0.0046 0.0081 0.0159 0.0178 0.0439 0.0198 ...
##  $ V51  : num  0.0232 0.0125 0.0033 0.0241 0.0156 0.0104 0.0195 0.0052 0.0061 0.0118 ...
##  $ V52  : num  0.0027 0.0084 0.0232 0.0121 0.0031 0.0045 0.0201 0.0081 0.0145 0.009 ...
##  $ V53  : num  0.0065 0.0089 0.0166 0.0036 0.0054 0.0014 0.0248 0.012 0.0128 0.0223 ...
```

```
## $ V54  : num  0.0159 0.0048 0.0095 0.015 0.0105 0.0038 0.0131 0.0045 0.0145 0.0179 ...
## $ V55  : num  0.0072 0.0094 0.018 0.0085 0.011 0.0013 0.007 0.0121 0.0058 0.0084 ...
## $ V56  : num  0.0167 0.0191 0.0244 0.0073 0.0015 0.0089 0.0138 0.0097 0.0049 0.0068 ...
## $ V57  : num  0.018 0.014 0.0316 0.005 0.0072 0.0057 0.0092 0.0085 0.0065 0.0032 ...
## $ V58  : num  0.0084 0.0049 0.0164 0.0044 0.0048 0.0027 0.0143 0.0047 0.0093 0.0035 ...
## $ V59  : num  0.009 0.0052 0.0095 0.004 0.0107 0.0051 0.0036 0.0048 0.0059 0.0056 ...
## $ V60  : num  0.0032 0.0044 0.0078 0.0117 0.0094 0.0062 0.0103 0.0053 0.0022 0.004 ...
## $ Class: Factor w/ 2 levels "M","R": 2 2 2 2 2 2 2 2 2 2 ...
```

STEP 2 : Feature Selection

```r
set.seed(111)
boruta <- Boruta(Class ~ ., data =  Sonar, doTrace = 2,maxRuns = 500)
```

```
##  1. run of importance source...
##  2. run of importance source...
##  3. run of importance source...
##  4. run of importance source...
##  5. run of importance source...
##  6. run of importance source...
##  7. run of importance source...
##  8. run of importance source...
##  9. run of importance source...
##  10. run of importance source...
##  11. run of importance source...
##  12. run of importance source...
##  13. run of importance source...
## After 13 iterations, +8.4 secs:
##  confirmed 14 attributes: V10, V11, V12, V13, V17 and 9 more;
##  rejected 5 attributes: V38, V56, V57, V60, V7;
##  still have 41 attributes left.
##  14. run of importance source...
##  15. run of importance source...
##  16. run of importance source...
##  17. run of importance source...
## After 17 iterations, +11 secs:
##  confirmed 1 attribute: V37;
##  rejected 3 attributes: V41, V50, V53;
##  still have 37 attributes left.
##  18. run of importance source...
```

```
##  19. run of importance source...
##  20. run of importance source...
##  21. run of importance source...
## After 21 iterations, +13 secs:
##  confirmed 6 attributes: V15, V16, V20, V21, V4 and 1 more;
##  rejected 1 attribute: V6;
##  still have 30 attributes left.
##  22. run of importance source...
##  23. run of importance source...
##  24. run of importance source...
## After 24 iterations, +15 secs:
##  confirmed 2 attributes: V51, V52;
##  still have 28 attributes left.
##  25. run of importance source...
##  26. run of importance source...
##  27. run of importance source...
## After 27 iterations, +17 secs:
##  rejected 1 attribute: V58;
##  still have 27 attributes left.
##  28. run of importance source...
##  29. run of importance source...
##  30. run of importance source...
## After 30 iterations, +19 secs:
##  confirmed 2 attributes: V18, V23;
##  rejected 1 attribute: V40;
##  still have 24 attributes left.
##  31. run of importance source...
##  32. run of importance source...
##  33. run of importance source...
## After 33 iterations, +20 secs:
##  rejected 1 attribute: V25;
##  still have 23 attributes left.
##  34. run of importance source...
##  35. run of importance source...
##  36. run of importance source...
##  37. run of importance source...
```

```
##  38. run of importance source...
##  39. run of importance source...
## After 39 iterations, +24 secs:
##  confirmed 2 attributes: V1, V31;
##  still have 21 attributes left.
##  40. run of importance source...
##  41. run of importance source...
##  42. run of importance source...
## After 42 iterations, +25 secs:
##  rejected 1 attribute: V3;
##  still have 20 attributes left.
##  43. run of importance source...
##  44. run of importance source...
##  45. run of importance source...
##  46. run of importance source...
##  47. run of importance source...
##  48. run of importance source...
## After 48 iterations, +28 secs:
##  rejected 1 attribute: V33;
##  still have 19 attributes left.
##  49. run of importance source...
##  50. run of importance source...
## After 50 iterations, +30 secs:
##  confirmed 1 attribute: V5;
##  still have 18 attributes left.
##  51. run of importance source...
##  52. run of importance source...
##  53. run of importance source...
##  54. run of importance source...
##  55. run of importance source...
##  56. run of importance source...
## After 56 iterations, +33 secs:
##  confirmed 1 attribute: V22;
##  still have 17 attributes left.
##  57. run of importance source...
##  58. run of importance source...
```

```
## After 58 iterations, +34 secs:
##  rejected 1 attribute: V55;
##  still have 16 attributes left.
##  59. run of importance source...
##  60. run of importance source...
##  61. run of importance source...
##  62. run of importance source...
##  63. run of importance source...
##  64. run of importance source...
## After 64 iterations, +37 secs:
##  confirmed 1 attribute: V43;
##  still have 15 attributes left.
##  65. run of importance source...
##  66. run of importance source...
## After 66 iterations, +38 secs:
##  confirmed 1 attribute: V19;
##  still have 14 attributes left.
##  67. run of importance source...
##  68. run of importance source...
##  69. run of importance source...
##  70. run of importance source...
##  71. run of importance source...
##  72. run of importance source...
##  73. run of importance source...
##  74. run of importance source...
## After 74 iterations, +42 secs:
##  confirmed 1 attribute: V35;
##  still have 13 attributes left.
##  75. run of importance source...
##  76. run of importance source...
##  77. run of importance source...
##  78. run of importance source...
##  79. run of importance source...
##  80. run of importance source...
##  81. run of importance source...
## After 81 iterations, +46 secs:
```

```
##  confirmed 1 attribute: V26;
##  still have 12 attributes left.
##  82. run of importance source...
##  83. run of importance source...
##  84. run of importance source...
##  85. run of importance source...
##  86. run of importance source...
##  87. run of importance source...
##  88. run of importance source...
##  89. run of importance source...
##  90. run of importance source...
##  91. run of importance source...
##  92. run of importance source...
##  93. run of importance source...
##  94. run of importance source...
##  95. run of importance source...
##  96. run of importance source...
##  97. run of importance source...
##  98. run of importance source...
##  99. run of importance source...
##  100. run of importance source...
##  101. run of importance source...
##  102. run of importance source...
##  103. run of importance source...
##  104. run of importance source...
## After 104 iterations, +59 secs:
##  rejected 1 attribute: V24;
##  still have 11 attributes left.
##  105. run of importance source...
##  106. run of importance source...
##  107. run of importance source...
##  108. run of importance source...
##  109. run of importance source...
##  110. run of importance source...
##  111. run of importance source...
##  112. run of importance source...
```

```
## 113. run of importance source...
## 114. run of importance source...
## 115. run of importance source...
## 116. run of importance source...
## 117. run of importance source...
## 118. run of importance source...
## 119. run of importance source...
## 120. run of importance source...
## 121. run of importance source...
## 122. run of importance source...
## 123. run of importance source...
## 124. run of importance source...
## 125. run of importance source...
## 126. run of importance source...
## 127. run of importance source...
## 128. run of importance source...
## 129. run of importance source...
## 130. run of importance source...
## 131. run of importance source...
## 132. run of importance source...
## 133. run of importance source...
## 134. run of importance source...
## 135. run of importance source...
## 136. run of importance source...
## 137. run of importance source...
## 138. run of importance source...
## 139. run of importance source...
## 140. run of importance source...
## 141. run of importance source...
## 142. run of importance source...
## 143. run of importance source...
## 144. run of importance source...
## 145. run of importance source...
## 146. run of importance source...
## 147. run of importance source...
## 148. run of importance source...
```

```
## 149. run of importance source...
## 150. run of importance source...
## 151. run of importance source...
## 152. run of importance source...
## 153. run of importance source...
## 154. run of importance source...
## 155. run of importance source...
## 156. run of importance source...
## 157. run of importance source...
## 158. run of importance source...
## 159. run of importance source...
## 160. run of importance source...
## 161. run of importance source...
## 162. run of importance source...
## 163. run of importance source...
## 164. run of importance source...
## 165. run of importance source...
## 166. run of importance source...
## 167. run of importance source...
## 168. run of importance source...
## 169. run of importance source...
## 170. run of importance source...
## 171. run of importance source...
## 172. run of importance source...
## 173. run of importance source...
## 174. run of importance source...
## 175. run of importance source...
## 176. run of importance source...
## 177. run of importance source...
## 178. run of importance source...
## 179. run of importance source...
## 180. run of importance source...
## 181. run of importance source...
## 182. run of importance source...
## 183. run of importance source...
## 184. run of importance source...
```

```
## 185. run of importance source...
## 186. run of importance source...
## 187. run of importance source...
## 188. run of importance source...
## 189. run of importance source...
## 190. run of importance source...
## 191. run of importance source...
## 192. run of importance source...
## 193. run of importance source...
## 194. run of importance source...
## 195. run of importance source...
## 196. run of importance source...
## 197. run of importance source...
## 198. run of importance source...
## 199. run of importance source...
## 200. run of importance source...
## 201. run of importance source...
## 202. run of importance source...
## 203. run of importance source...
## 204. run of importance source...
## 205. run of importance source...
## 206. run of importance source...
## 207. run of importance source...
## 208. run of importance source...
## 209. run of importance source...
## 210. run of importance source...
## 211. run of importance source...
## 212. run of importance source...
## 213. run of importance source...
## 214. run of importance source...
## 215. run of importance source...
## 216. run of importance source...
## 217. run of importance source...
## 218. run of importance source...
## 219. run of importance source...
## 220. run of importance source...
```

```
## 221. run of importance source...
## 222. run of importance source...
## 223. run of importance source...
## 224. run of importance source...
## 225. run of importance source...
## 226. run of importance source...
## 227. run of importance source...
## 228. run of importance source...
## 229. run of importance source...
## 230. run of importance source...
## 231. run of importance source...
## 232. run of importance source...
## 233. run of importance source...
## 234. run of importance source...
## 235. run of importance source...
## 236. run of importance source...
## 237. run of importance source...
## 238. run of importance source...
## 239. run of importance source...
## 240. run of importance source...
## 241. run of importance source...
## 242. run of importance source...
## 243. run of importance source...
## 244. run of importance source...
## 245. run of importance source...
## 246. run of importance source...
## 247. run of importance source...
## 248. run of importance source...
## 249. run of importance source...
## 250. run of importance source...
## 251. run of importance source...
## 252. run of importance source...
## 253. run of importance source...
## 254. run of importance source...
## 255. run of importance source...
## 256. run of importance source...
```

```
## 257. run of importance source...
## 258. run of importance source...
## 259. run of importance source...
## 260. run of importance source...
## 261. run of importance source...
## 262. run of importance source...
## 263. run of importance source...
## 264. run of importance source...
## 265. run of importance source...
## 266. run of importance source...
## 267. run of importance source...
## 268. run of importance source...
## 269. run of importance source...
## 270. run of importance source...
## 271. run of importance source...
## 272. run of importance source...
## 273. run of importance source...
## 274. run of importance source...
## 275. run of importance source...
## 276. run of importance source...
## 277. run of importance source...
## 278. run of importance source...
## 279. run of importance source...
## 280. run of importance source...
## 281. run of importance source...
## 282. run of importance source...
## 283. run of importance source...
## 284. run of importance source...
## 285. run of importance source...
## 286. run of importance source...
## 287. run of importance source...
## 288. run of importance source...
## 289. run of importance source...
## 290. run of importance source...
## 291. run of importance source...
## 292. run of importance source...
```

```
## 293. run of importance source...
## 294. run of importance source...
## 295. run of importance source...
## 296. run of importance source...
## 297. run of importance source...
## 298. run of importance source...
## 299. run of importance source...
## 300. run of importance source...
## 301. run of importance source...
## 302. run of importance source...
## 303. run of importance source...
## 304. run of importance source...
## 305. run of importance source...
## 306. run of importance source...
## 307. run of importance source...
## 308. run of importance source...
## 309. run of importance source...
## 310. run of importance source...
## 311. run of importance source...
## 312. run of importance source...
## 313. run of importance source...
## 314. run of importance source...
## 315. run of importance source...
## 316. run of importance source...
## 317. run of importance source...
## 318. run of importance source...
## 319. run of importance source...
## 320. run of importance source...
## 321. run of importance source...
## 322. run of importance source...
## 323. run of importance source...
## 324. run of importance source...
## 325. run of importance source...
## 326. run of importance source...
## 327. run of importance source...
## 328. run of importance source...
```

```
##  329. run of importance source...
##  330. run of importance source...
##  331. run of importance source...
##  332. run of importance source...
##  333. run of importance source...
##  334. run of importance source...
##  335. run of importance source...
##  336. run of importance source...
##  337. run of importance source...
##  338. run of importance source...
##  339. run of importance source...
##  340. run of importance source...
##  341. run of importance source...
##  342. run of importance source...
##  343. run of importance source...
##  344. run of importance source...
##  345. run of importance source...
##  346. run of importance source...
##  347. run of importance source...
##  348. run of importance source...
## After 348 iterations, +3.1 mins:
##  rejected 1 attribute: V29;
##  still have 10 attributes left.
##  349. run of importance source...
##  350. run of importance source...
##  351. run of importance source...
##  352. run of importance source...
##  353. run of importance source...
##  354. run of importance source...
##  355. run of importance source...
##  356. run of importance source...
##  357. run of importance source...
##  358. run of importance source...
##  359. run of importance source...
##  360. run of importance source...
##  361. run of importance source...
```

```
##  362. run of importance source...
##  363. run of importance source...
##  364. run of importance source...
##  365. run of importance source...
##  366. run of importance source...
##  367. run of importance source...
##  368. run of importance source...
##  369. run of importance source...
##  370. run of importance source...
##  371. run of importance source...
##  372. run of importance source...
##  373. run of importance source...
##  374. run of importance source...
##  375. run of importance source...
##  376. run of importance source...
##  377. run of importance source...
##  378. run of importance source...
##  379. run of importance source...
##  380. run of importance source...
##  381. run of importance source...
## After 381 iterations, +3.4 mins:
##  rejected 1 attribute: V34;
##  still have 9 attributes left.
##  382. run of importance source...
##  383. run of importance source...
##  384. run of importance source...
##  385. run of importance source...
##  386. run of importance source...
##  387. run of importance source...
##  388. run of importance source...
##  389. run of importance source...
##  390. run of importance source...
##  391. run of importance source...
##  392. run of importance source...
##  393. run of importance source...
##  394. run of importance source...
```

```
##  395. run of importance source...
##  396. run of importance source...
##  397. run of importance source...
##  398. run of importance source...
##  399. run of importance source...
##  400. run of importance source...
##  401. run of importance source...
##  402. run of importance source...
##  403. run of importance source...
##  404. run of importance source...
##  405. run of importance source...
##  406. run of importance source...
##  407. run of importance source...
##  408. run of importance source...
##  409. run of importance source...
##  410. run of importance source...
##  411. run of importance source...
##  412. run of importance source...
##  413. run of importance source...
##  414. run of importance source...
##  415. run of importance source...
##  416. run of importance source...
##  417. run of importance source...
##  418. run of importance source...
##  419. run of importance source...
## After 419 iterations, +3.7 mins:
##  rejected 1 attribute: V42;
##  still have 8 attributes left.
##  420. run of importance source...
##  421. run of importance source...
##  422. run of importance source...
##  423. run of importance source...
##  424. run of importance source...
##  425. run of importance source...
##  426. run of importance source...
##  427. run of importance source...
```

```
## 428. run of importance source...
## 429. run of importance source...
## 430. run of importance source...
## 431. run of importance source...
## 432. run of importance source...
## 433. run of importance source...
## 434. run of importance source...
## 435. run of importance source...
## 436. run of importance source...
## 437. run of importance source...
## 438. run of importance source...
## 439. run of importance source...
## 440. run of importance source...
## 441. run of importance source...
## 442. run of importance source...
## 443. run of importance source...
## 444. run of importance source...
## 445. run of importance source...
## 446. run of importance source...
## 447. run of importance source...
## 448. run of importance source...
## 449. run of importance source...
## 450. run of importance source...
## 451. run of importance source...
## 452. run of importance source...
## 453. run of importance source...
## 454. run of importance source...
## 455. run of importance source...
## 456. run of importance source...
## 457. run of importance source...
## 458. run of importance source...
## 459. run of importance source...
## 460. run of importance source...
## 461. run of importance source...
## 462. run of importance source...
## 463. run of importance source...
```

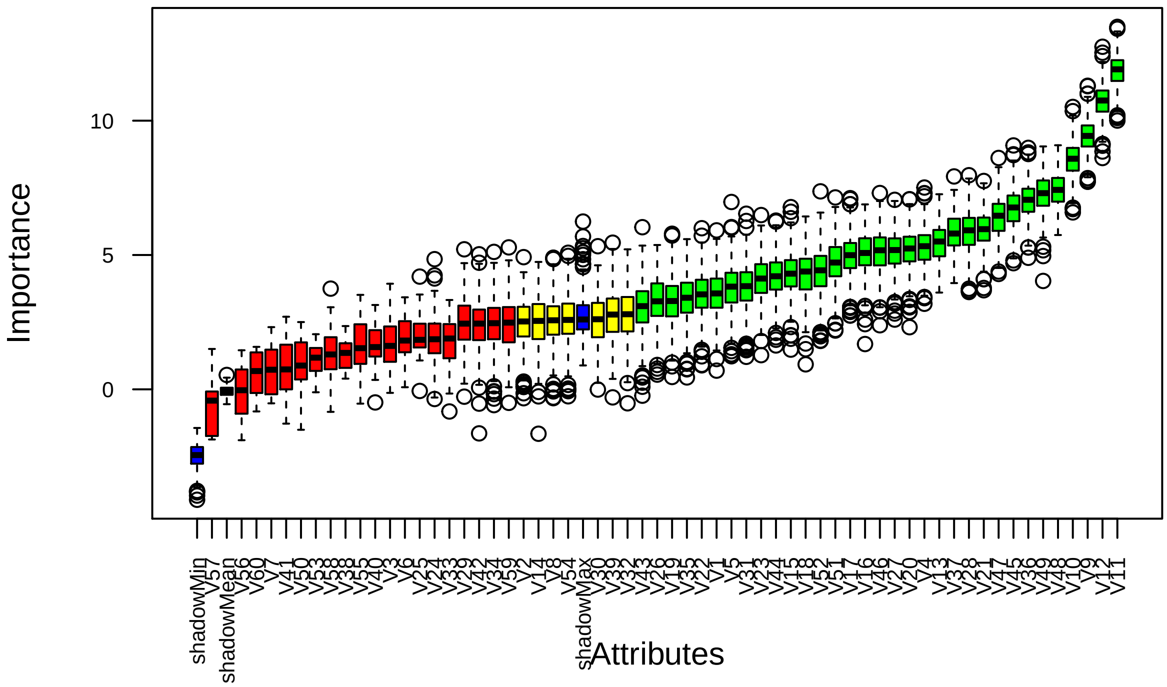
```
##  464. run of importance source...
##  465. run of importance source...
##  466. run of importance source...
##  467. run of importance source...
##  468. run of importance source...
##  469. run of importance source...
##  470. run of importance source...
##  471. run of importance source...
##  472. run of importance source...
##  473. run of importance source...
##  474. run of importance source...
##  475. run of importance source...
##  476. run of importance source...
##  477. run of importance source...
##  478. run of importance source...
##  479. run of importance source...
##  480. run of importance source...
##  481. run of importance source...
##  482. run of importance source...
##  483. run of importance source...
##  484. run of importance source...
##  485. run of importance source...
##  486. run of importance source...
##  487. run of importance source...
##  488. run of importance source...
##  489. run of importance source...
##  490. run of importance source...
##  491. run of importance source...
##  492. run of importance source...
##  493. run of importance source...
##  494. run of importance source...
##  495. run of importance source...
## After 495 iterations, +4.3 mins:
##  rejected 1 attribute: V59;
##  still have 7 attributes left.
##  496. run of importance source...
```

```
## 497. run of importance source...

## 498. run of importance source...

## 499. run of importance source...
```

```r
print(boruta)
```

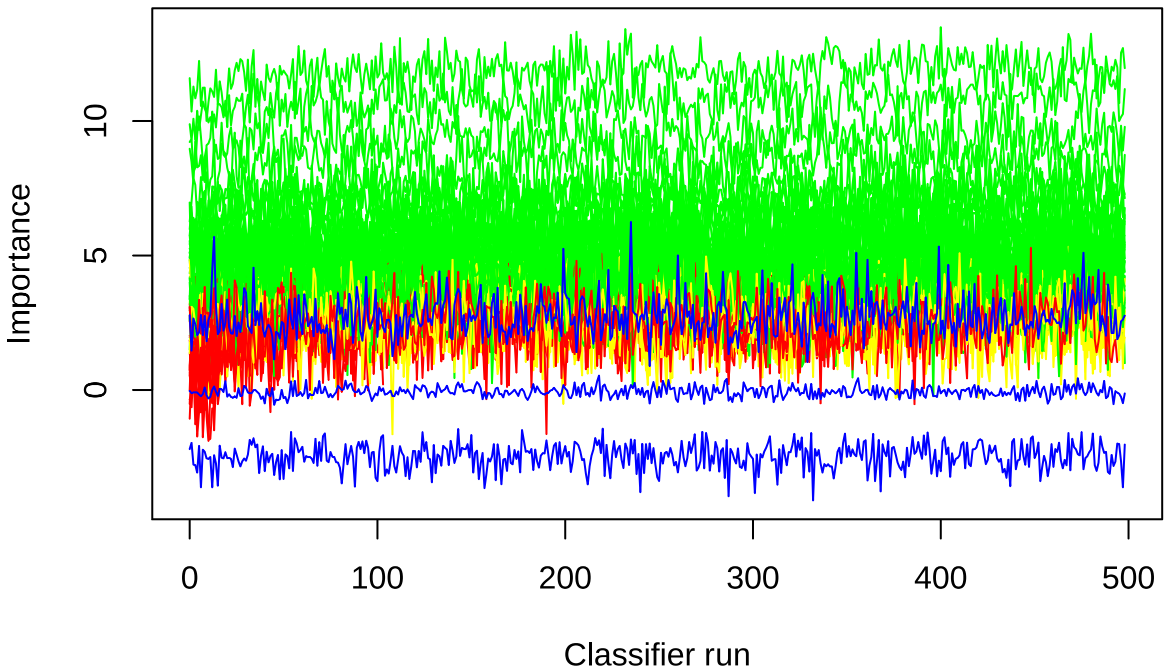
```
## Boruta performed 499 iterations in 4.363847 mins.
##  33 attributes confirmed important: V1, V10, V11, V12, V13 and 28
## more;
##  20 attributes confirmed unimportant: V24, V25, V29, V3, V33 and
## 15 more;
##  7 tentative attributes left: V14, V2, V30, V32, V39 and 2 more;
```

```r
plot(boruta,las =2,cex.axis =0.7)
```



Blue boxplot's basically corresponds to shadow attributes, we have three blue boxplots that corresponde to min Avge, and max importance to the shadow attributes. ### boxplot's in green that are confirmed important attributes. ### boxplot's in yellow that are Tentative ### boxplot's in Red that are confirmed unimportant attributes.

```r
plotImpHistory(boruta)
```



###

Tentative Fix

```
bor <- TentativeRoughFix(boruta)
print(bor)
```

```
## Boruta performed 499 iterations in 4.363847 mins.
## Tentatives roughfixed over the last 499 iterations.
##  37 attributes confirmed important: V1, V10, V11, V12, V13 and 32
## more;
##  23 attributes confirmed unimportant: V24, V25, V29, V3, V32 and
## 18 more;
```

```
a <- attStats(boruta)
```

Data Partition

```
set.seed(222)
ind <- sample(2,nrow(Sonar),replace = T,prob = c(0.6,0.4))
train <- Sonar[ind == 1,]
test <- Sonar[ind == 2,]
```

Random Forest Model

```
set.seed(333)
rf60 <- randomForest(Class ~., data = train)
rf60
```

```
##
## Call:
##  randomForest(formula = Class ~ ., data = train)
```

```
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 23.08%
## Confusion matrix:
##    M  R class.error
## M 51 10   0.1639344
## R 17 39   0.3035714
```

prediction & confusion Matrix - Test

```r
p <- predict(rf60,test)
confusionMatrix(p,test$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 46 17
##          R  4 24
##
##                Accuracy : 0.7692
##                  95% CI : (0.6691, 0.8511)
##     No Information Rate : 0.5495
##     P-Value [Acc > NIR] : 1.134e-05
##
##                   Kappa : 0.5202
##
##  Mcnemar's Test P-Value : 0.008829
##
##             Sensitivity : 0.9200
##             Specificity : 0.5854
##          Pos Pred Value : 0.7302
##          Neg Pred Value : 0.8571
##              Prevalence : 0.5495
##          Detection Rate : 0.5055
##    Detection Prevalence : 0.6923
##       Balanced Accuracy : 0.7527
##
##        'Positive' Class : M
##
```

```r
getNonRejectedFormula(boruta)
```

```
## Class ~ V1 + V2 + V4 + V5 + V8 + V9 + V10 + V11 + V12 + V13 +
##     V14 + V15 + V16 + V17 + V18 + V19 + V20 + V21 + V22 + V23 +
##     V26 + V27 + V28 + V30 + V31 + V32 + V35 + V36 + V37 + V39 +
##     V43 + V44 + V45 + V46 + V47 + V48 + V49 + V51 + V52 + V54
## <environment: 0xe012bc8>
```

```r
set.seed(333)
rf41 <- randomForest(Class ~ V1 + V2 + V4 + V5 + V8 + V9 + V10 + V11 + V12 + V13 +
```

```r
    V15 + V16 + V17 + V18 + V19 + V20 + V21 + V22 + V23 + V26 +
    V27 + V28 + V29 + V30 + V31 + V32 + V34 + V35 + V36 + V37 +
    V39 + V43 + V44 + V45 + V46 + V47 + V48 + V49 + V51 + V52 +
    V54,data = train)
```

```r
p41 <- predict(rf41,test)
confusionMatrix(p41,test$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 44 17
##          R  6 24
##
##                Accuracy : 0.7473
##                  95% CI : (0.6453, 0.8325)
##     No Information Rate : 0.5495
##     P-Value [Acc > NIR] : 7.793e-05
##
##                   Kappa : 0.4769
##
##  Mcnemar's Test P-Value : 0.03706
##
##             Sensitivity : 0.8800
##             Specificity : 0.5854
##          Pos Pred Value : 0.7213
##          Neg Pred Value : 0.8000
##              Prevalence : 0.5495
##          Detection Rate : 0.4835
##    Detection Prevalence : 0.6703
##       Balanced Accuracy : 0.7327
##
##        'Positive' Class : M
##
```

```r
getConfirmedFormula(boruta)
```

```
## Class ~ V1 + V4 + V5 + V9 + V10 + V11 + V12 + V13 + V15 + V16 +
##     V17 + V18 + V19 + V20 + V21 + V22 + V23 + V26 + V27 + V28 +
##     V31 + V35 + V36 + V37 + V43 + V44 + V45 + V46 + V47 + V48 +
##     V49 + V51 + V52
## <environment: 0xdc54510>
```

```r
set.seed(333)
rf33 <- randomForest(Class ~ V1 + V4 + V5 + V9 + V10 + V11 + V12 + V13 + V15 + V16 +
    V17 + V18 + V19 + V20 + V21 + V22 + V23 + V26 + V27 + V28 +
    V31 + V35 + V36 + V37 + V43 + V44 + V45 + V46 + V47 + V48 +
    V49 + V51 + V52,data = train)
```

Prediction and confustion matrix - Test

```r
p33 <- predict(rf33, test)
confusionMatrix(p33 , test$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 44 16
##          R  6 25
##
##                Accuracy : 0.7582
##                  95% CI : (0.6572, 0.8419)
##     No Information Rate : 0.5495
##     P-Value [Acc > NIR] : 3.058e-05
##
##                   Kappa : 0.5007
##
##  Mcnemar's Test P-Value : 0.05501
##
##             Sensitivity : 0.8800
##             Specificity : 0.6098
##          Pos Pred Value : 0.7333
##          Neg Pred Value : 0.8065
##              Prevalence : 0.5495
##          Detection Rate : 0.4835
##    Detection Prevalence : 0.6593
##       Balanced Accuracy : 0.7449
##
##        'Positive' Class : M
##
```

```
## [1] 2
```

Chapter 4 : Handling Class Imbalance Problem

What is the Class Imbalance Problem?

It is the problem in machine learning where the total number of a class of data (positive) is far less than the total number of another class of data (negative). This problem is extremely common in practice and can be observed in various disciplines including fraud detection, anomaly detection, medical diagnosis, oil spillage detection, facial recognition, etc. ## Read data and see its structures

```r
data <- read.csv("binary.csv",header = T,sep = ",")
str(data)
```

```
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```

Convert numerical variables to factor and see the structure and summary

```r
data$admit <- as.factor(data$admit)
str(data)
```

```
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 1 2 1 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```

```r
summary(data)
```

```
##  admit        gre             gpa             rank      
##  0:273   Min.   :220.0   Min.   :2.260   Min.   :1.000  
##  1:127   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000  
##          Median :580.0   Median :3.395   Median :2.000  
##          Mean   :587.7   Mean   :3.390   Mean   :2.485  
##          3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000  
##          Max.   :800.0   Max.   :4.000   Max.   :4.000  
```

Class imbalance problem

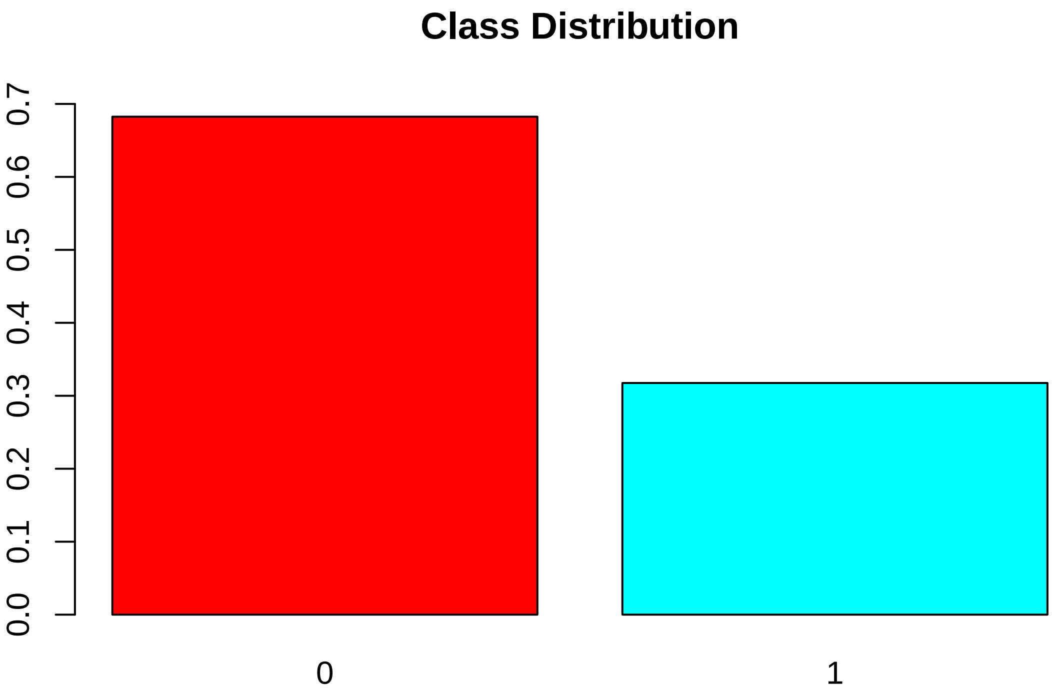
you can also convert data into proporations

```r
prop.table(table(data$admit))
```

```
## 
##      0      1 
## 0.6825 0.3175
```

you can convert into barplot also

```
barplot(prop.table(table(data$admit)),col=rainbow(2),ylim=c(0,0.7),main = "Class Distribution")
```



we are saying class imbalance because (look at above graph) 2/3 of data is belonged to (red or 0) category where the students was not admited, and 1/3 is belonged to 1 where the students was admited. so there is there is big difference in the amount of data available for the two classes so that's why we say there is a class imbalance; because blue wala bar is much lower and red wala bar is quit high.

When we develop prediction model on such data what will happen is, predicted model will be dominated by contribution from data in higher class. Accuracy of the model will be better when predicting 0 class vs 1 class. but there may be a situation we are more interested predicting 1 accuratly compared to 0. so this is called Class imbalance problem.

STEP 1 : Data Partion

```
set.seed(123)
ind <- sample(2,nrow(data), replace = TRUE, prob = c(0.7,0.3))
train <- data[ind == 1,]
test <- data[ind == 2,]
```

STEP 2 : Data for Developing Predictive Model.

Before that check how many data belongs to 0 and 1

```
table(train$admit)
```

```
##
```

```
##   0   1
## 188  97
```

```r
prop.table(table(train$admit))
```

```
##
##         0         1
## 0.6596491 0.3403509
```

STEP 3 : Build Predictive Model using RandomForest Algorithm

```r
library(randomForest)
rftrain <- randomForest(admit ~ ., data = train)
```

STEP 4 : Model Evaluation with Test Data

for Evaluation of model we will make use of test data, so this test data consist of 30% observations of population that randomForest model has not seen, because model has made with training data so it is only seen by training data.

```r
library(caret)
library(e1071)
confusionMatrix(predict(rftrain,test),test$admit,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 70 22
##          1 15  8
##
##                Accuracy : 0.6783
##                  95% CI : (0.5847, 0.7623)
##     No Information Rate : 0.7391
##     P-Value [Acc > NIR] : 0.9418
##
##                   Kappa : 0.0976
##
##  Mcnemar's Test P-Value : 0.3239
##
##             Sensitivity : 0.26667
##             Specificity : 0.82353
##          Pos Pred Value : 0.34783
##          Neg Pred Value : 0.76087
##              Prevalence : 0.26087
##          Detection Rate : 0.06957
##    Detection Prevalence : 0.20000
##       Balanced Accuracy : 0.54510
##
##        'Positive' Class : 1
##
```

If you want predict 0 then Specificity is not bad it is very good model.

but if you want to predict 1 then Sensitivity is very bad for this model since this is not a good model so what can be done to improve the situation and

one of the way to do this is go for Oversampling

STEP 5 : Oversampling for Better Sensitivity

```
library(ROSE) # ROSE -> Randomly Over sampling Examples
```

```
## Loaded ROSE 0.0-3
```

```
over <- ovun.sample(admit ~ ., data = train,method = "over",N =376)$data
```

we can also specify total sample in aboce model, if you want to keep same sample i.e 188 for 0 as well as 1

```
table(train$admit)
```

```
##
##   0   1
## 188  97
```

```
188*2
```

```
## [1] 376
```

See below samples are same now

```
table(over$admit)
```

```
##
##   0   1
## 188 188
```

```
summary(over)
```

```
##  admit        gre             gpa             rank
##  0:188   Min.   :220.0   Min.   :2.260   Min.   :1.000
##  1:188   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
##          Median :580.0   Median :3.450   Median :2.000
##          Mean   :587.2   Mean   :3.408   Mean   :2.415
##          3rd Qu.:660.0   3rd Qu.:3.692   3rd Qu.:3.000
##          Max.   :800.0   Max.   :4.000   Max.   :4.000
```

STEP 6 : Let's make randomForest Model for Over dataset

```
rfover <- randomForest(admit ~. , data = over)
confusionMatrix(predict(rfover,test),test$admit,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 59 14
##          1 26 16
##
##                Accuracy : 0.6522
##                  95% CI : (0.5577, 0.7386)
##     No Information Rate : 0.7391
##     P-Value [Acc > NIR] : 0.98516
##
##                   Kappa : 0.2014
##
##  Mcnemar's Test P-Value : 0.08199
##
##             Sensitivity : 0.5333
##             Specificity : 0.6941
##          Pos Pred Value : 0.3810
##          Neg Pred Value : 0.8082
##              Prevalence : 0.2609
##          Detection Rate : 0.1391
##    Detection Prevalence : 0.3652
##       Balanced Accuracy : 0.6137
##
##        'Positive' Class : 1
##
```

Under Sampling

```r
table(train$admit)
```

```
##
##   0   1
## 188  97
```

```r
97 * 2 # for under sampling
```

```
## [1] 194
```

```r
library(ROSE)
under <- ovun.sample(admit ~ ., data = train,method = "under",N =194)$data
```

```r
table(under$admit)
```

```
##
##  0  1
## 97 97
```

build a randomForset model based on under dataset

```r
rfunder <- randomForest(admit ~.,data = under)
confusionMatrix(predict(rfunder,test),test$admit,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 38 10
##          1 47 20
##
##                Accuracy : 0.5043
##                  95% CI : (0.4096, 0.5989)
##     No Information Rate : 0.7391
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0813
##
##  Mcnemar's Test P-Value : 1.858e-06
##
##             Sensitivity : 0.6667
##             Specificity : 0.4471
##          Pos Pred Value : 0.2985
##          Neg Pred Value : 0.7917
##              Prevalence : 0.2609
##          Detection Rate : 0.1739
##    Detection Prevalence : 0.5826
##       Balanced Accuracy : 0.5569
##
##        'Positive' Class : 1
##
```

Both Under & Over Sampling

```
both <- ovun.sample(admit ~ ., data = train,method = "both",p = 0.5, seed = 222,N = 285)$data
```

```
table(both$admit)
```

```
##
##   0   1
## 134 151
```

```
rfboth <- randomForest(admit ~., data = both)
confusionMatrix(predict(rfboth,test),test$admit,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 47 15
##          1 38 15
##
##                Accuracy : 0.5391
##                  95% CI : (0.4437, 0.6325)
##     No Information Rate : 0.7391
##     P-Value [Acc > NIR] : 0.999999
##
##                   Kappa : 0.0424
```

```
##
##  Mcnemar's Test P-Value : 0.002512
##
##             Sensitivity : 0.5000
##             Specificity : 0.5529
##          Pos Pred Value : 0.2830
##          Neg Pred Value : 0.7581
##              Prevalence : 0.2609
##          Detection Rate : 0.1304
##    Detection Prevalence : 0.4609
##       Balanced Accuracy : 0.5265
##
##        'Positive' Class : 1
##
```

Synthetic Data

```
rose <- ROSE(admit ~., data = train, N = 500,seed = 111)$data
```

```
table(rose$admit)
```

```
##
##   0   1
## 234 266
```

```
summary(rose)
```

```
##  admit        gre              gpa              rank
##  0:234   Min.   :129.0   Min.   :2.218   Min.   :-0.2386
##  1:266   1st Qu.:504.8   1st Qu.:3.150   1st Qu.: 1.6703
##          Median :585.1   Median :3.462   Median : 2.3475
##          Mean   :586.6   Mean   :3.442   Mean   : 2.4003
##          3rd Qu.:679.4   3rd Qu.:3.737   3rd Qu.: 3.1480
##          Max.   :907.0   Max.   :4.438   Max.   : 4.7961
```

```
rfrose <- randomForest(admit ~.,data = rose)
confusionMatrix(predict(rfrose,test),test$admit,positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 37 11
##          1 48 19
##
##                Accuracy : 0.487
##                  95% CI : (0.3927, 0.5819)
##     No Information Rate : 0.7391
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0491
##
##  Mcnemar's Test P-Value : 2.775e-06
##
##             Sensitivity : 0.6333
```

```
##             Specificity : 0.4353
##          Pos Pred Value : 0.2836
##          Neg Pred Value : 0.7708
##              Prevalence : 0.2609
##          Detection Rate : 0.1652
##    Detection Prevalence : 0.5826
##       Balanced Accuracy : 0.5343
##
##        'Positive' Class : 1
##
```

```r
30/115
```

```
## [1] 0.2608696
```

NOTE : Sensitivity can not be < 0.2608696 otherwise it will not be usefull. as long as it is better sensitivity > than specificity.

```
## [1] 2
```

Chapter 5 : Principal Component Analysis

Principal Component Analysis (PCA) is a feature extraction methods that use orthogonal(statistically independant) linear projections to capture the underlying variance of the data.

Step 1 : Read Data

```r
data("iris")
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Step 2 : Data partition

```r
set.seed(111)
ind <- sample(2,nrow(iris),replace = T,prob = c(0.8,0.2))
training <- iris[ind == 1,]
testing <- iris[ind == 2,]
```

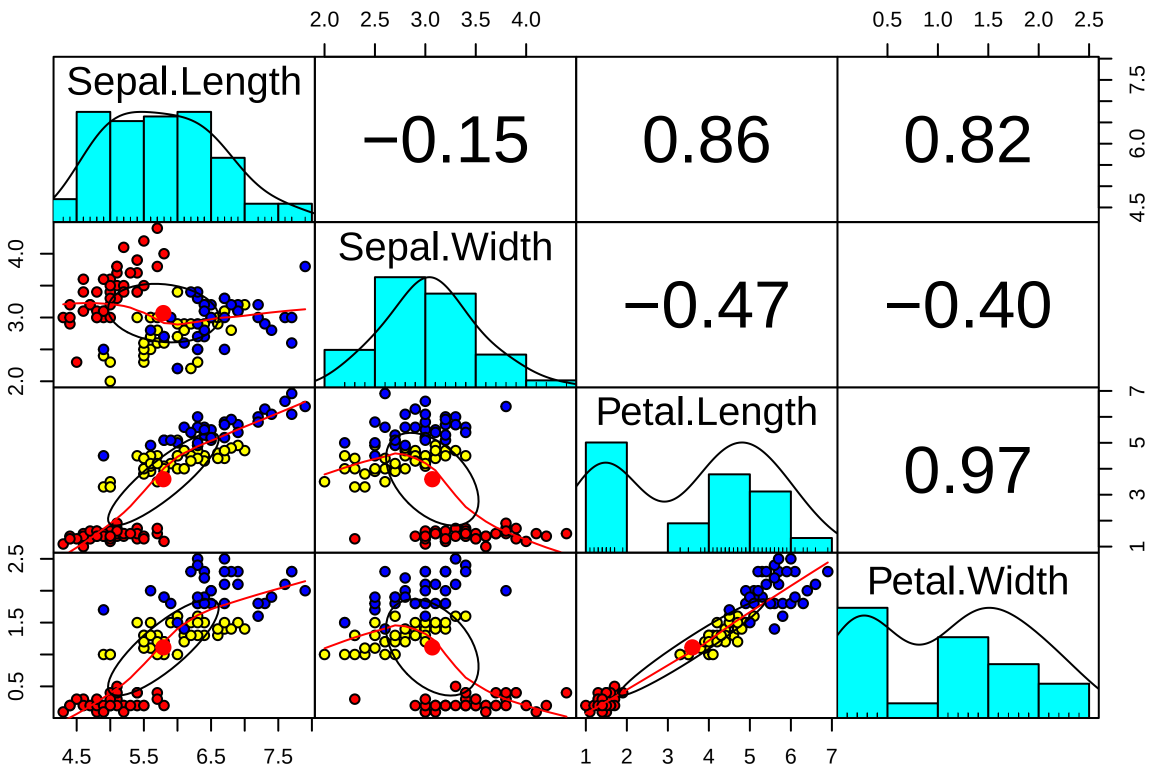
Step 3 : Check correlation with help scatterplot

```r
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following object is masked from 'package:randomForest':
##
##     outlier
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
pairs.panels(training[,-5],gap = 0,bg=c('red','yellow','blue')[training$Species],pch =21)
```



##
NOTE : High correlations among independent variables lead to "Multicollinearity" problem. and an estimate for the modelthat we get, is very unstable or predictions are not going to be accurate, so one way we can handle this is using PRINCIPAL COMPONENT ANALYSIS.

```
pc <- prcomp(training[,-5],center = TRUE,scale. = TRUE  )
```

NOTE : PCA is done only on the independent variables

Centre argument make sure that variables are converted such a way that the Average become 0.

Scale. make sure that before PCA done all the four variables are normalized.

```
attributes(pc)
```

```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

```r
pc$center # this will give us the Average of all four variables.
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.790000     3.069167     3.597500     1.111667
```

```r
print(pc)
```

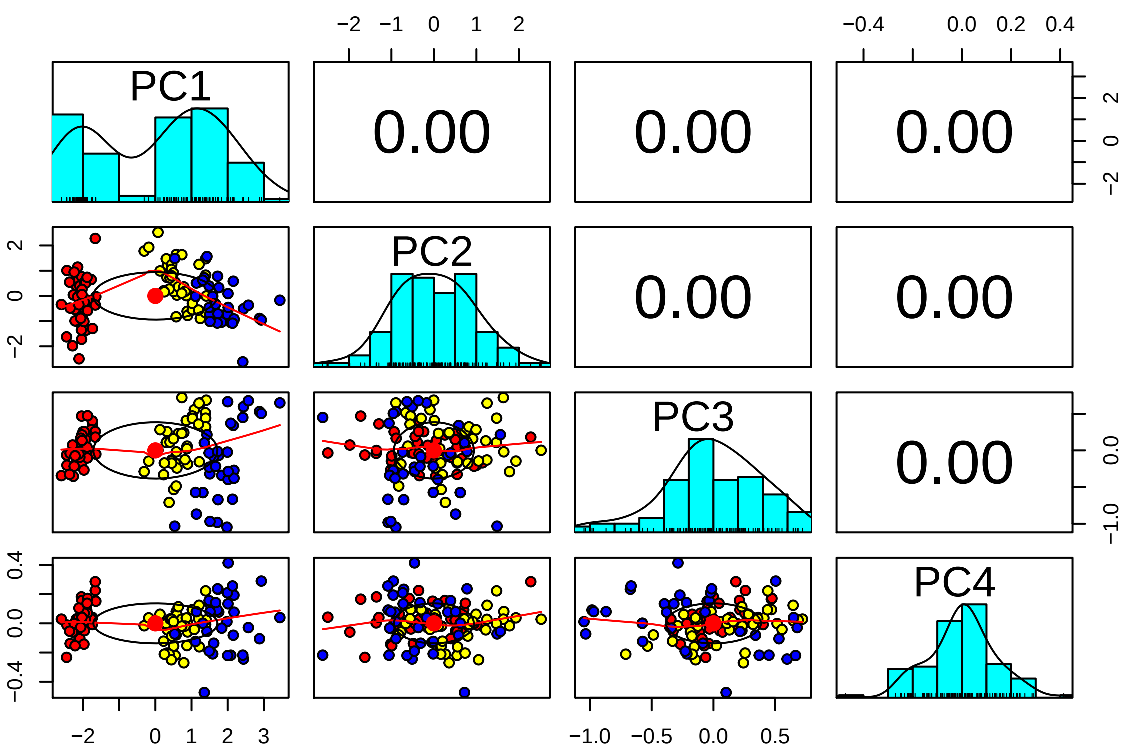
```
## Standard deviations (1, .., p=4):
## [1] 1.7173318 0.9403519 0.3843232 0.1371332
##
## Rotation (n x k) = (4 x 4):
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5147163 -0.39817685  0.7242679  0.2279438
## Sepal.Width  -0.2926048 -0.91328503 -0.2557463 -0.1220110
## Petal.Length  0.5772530 -0.02932037 -0.1755427 -0.7969342
## Petal.Width   0.5623421 -0.08065952 -0.6158040  0.5459403
```

```r
summary(pc)
```

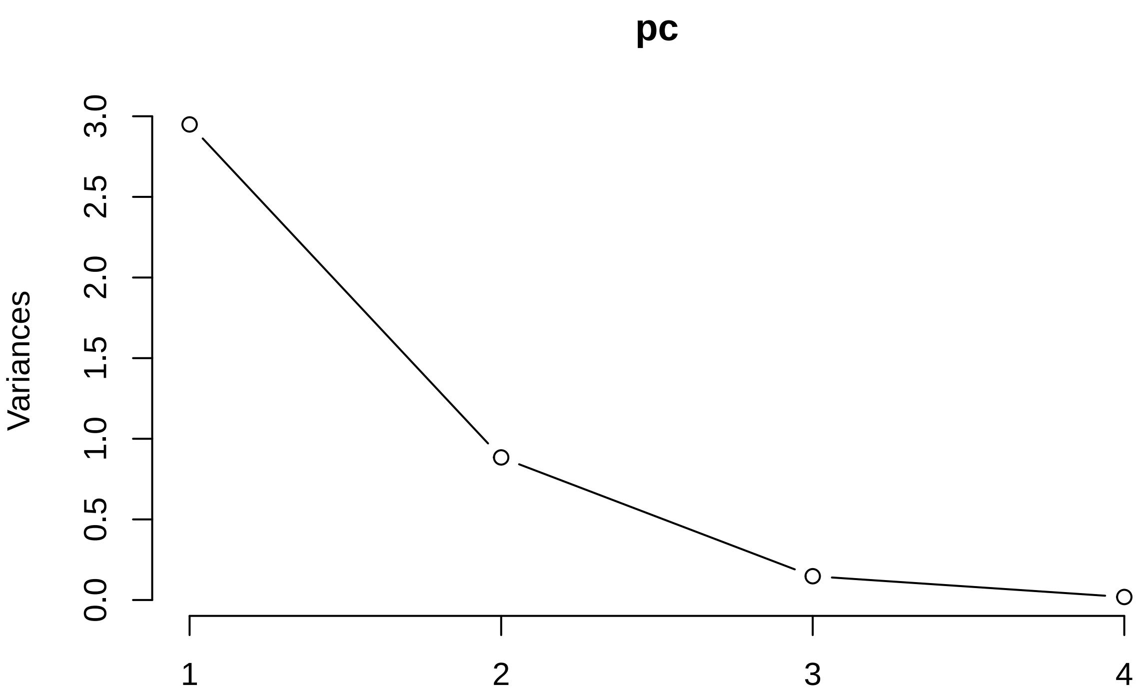
```
## Importance of components:
##                           PC1    PC2     PC3    PC4
## Standard deviation     1.7173 0.9404 0.38432 0.1371
## Proportion of Variance 0.7373 0.2211 0.03693 0.0047
## Cumulative Proportion  0.7373 0.9584 0.99530 1.0000
```

Orthogonality of Principal COmponent

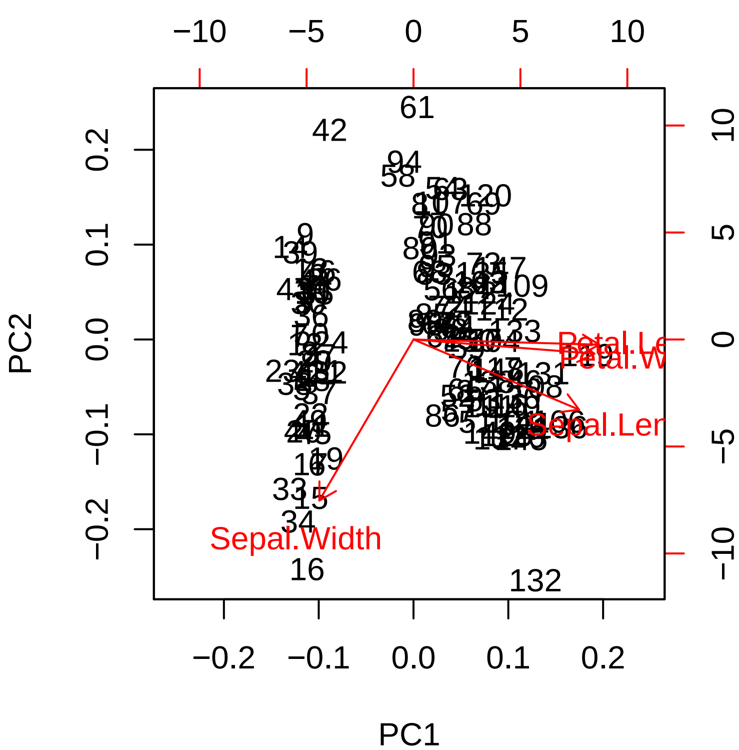
```r
library(psych)
pairs.panels(pc$x,bg=c("red","yellow","blue")[training$Species],pch=21)
```



```r
plot(pc,type="l")
```




```r
biplot(pc)
```




```r
attributes(pc)
```

```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
```

```
## $class
## [1] "prcomp"
```

```
## [1] 2
```

Chapter 6 : Partitioning Data into Training and Validation Datasets

STEP 1 : Read Data

```r
data <- read.csv("vehicle.csv",header = T,sep=",")
```

STEP 2 : Split dataset into "training (80%) and"validation" (20%)

```r
ind <- sample(2,nrow(data),replace=T,prob = c(0.8,0.2))
tdata <- data[ind == 1,]
vdata <- data[ind == 2,]
```

```r
head(tdata)
```

```
##   vehicle fm Mileage  lh     lc     mc State
## 2       2 10    4644 2.4 233.03 119.66    CA
## 3       3 15   16330 4.2 325.08 175.46    WI
## 4       4  0      13 1.0  66.64   0.00    OR
## 5       5 13   22537 4.5 328.66 175.46    AZ
## 6       6 21   40931 3.1 205.28 175.46    FL
## 8       8  5   11051 2.9 208.80 270.04    GA
```

```r
head(vdata)
```

```
##    vehicle fm Mileage  lh     lc     mc State
## 1        1  0     863 1.1  66.30 697.23    MS
## 7        7 11   34762 0.7  49.17 145.20    LA
## 18      18  3   15365 2.0 158.94 175.46    CO
## 21      21 18   29987 2.6 182.17 128.21    IN
## 31      31 17   24719 2.4 147.98 119.66    NY
## 40      40  8   20370 0.7  56.60 128.21    IL
```

Multiple linear regression model

```r
results <- lm(lc ~ Mileage + lh,tdata)
```

```r
summary(results)
```

```
##
## Call:
## lm(formula = lc ~ Mileage + lh, data = tdata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -619.45  -15.31   -3.14   13.87  448.13
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.434e+00  2.291e+00   2.808  0.00506 **
## Mileage     -6.309e-05  6.872e-05  -0.918  0.35874
## lh           7.174e+01  4.308e-01 166.522  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 44.83 on 1300 degrees of freedom
##   (19 observations deleted due to missingness)
## Multiple R-squared:  0.9554, Adjusted R-squared:  0.9553
## F-statistic: 1.392e+04 on 2 and 1300 DF,  p-value: < 2.2e-16
```

```r
results$coefficients
```

```
##   (Intercept)       Mileage            lh
##  6.433585e+00 -6.309247e-05  7.174487e+01
```

prediction

```r
pred <- predict(results, vdata)
```

```r
head(pred)
```

```
##         1         7        18        21        31        40
##  85.29850  54.46178 148.95392 191.07830 177.06170  55.36980
```

```
## [1] 2
```

Chapter 7 : BINNING : Changing Variables from Numeric to factors

STEP 1 : Read Data and add colClasses = "factor"

```
mydata <- read.csv("vehicle.csv",header = T,colClasses = "factor")
```

```
str(mydata)
```

```
## 'data.frame':    1624 obs. of  7 variables:
##  $ vehicle: Factor w/ 1624 levels "1","10","100",..: 1 737 848 959 1070 1181 1292 1403 1514 2 ...
##  $ fm     : Factor w/ 25 levels "-1","0","1","10",..: 2 4 9 2 7 16 5 21 24 3 ...
##  $ Mileage: Factor w/ 1537 levels "1","10","1000",..: 1484 1147 285 131 548 1055 939 50 1385 46 ...
##  $ lh     : Factor w/ 124 levels "0","0.2","0.3",..: 11 48 73 10 76 59 7 53 62 7 ...
##  $ lc     : Factor w/ 1504 levels "0","10.57","100",..: 1313 603 862 1317 874 498 1133 513 523 1075
##  $ mc     : Factor w/ 347 levels "0","1.75","1.85",..: 322 14 124 1 124 124 87 219 14 1 ...
##  $ State  : Factor w/ 50 levels "AK","AL","AR",..: 25 5 48 37 4 9 18 10 47 38 ...
```

Using "cut" command

```
df <- data.frame(a = rnorm(10))
```

```
df
```

```
##             a
## 1  -1.0796230
## 2   0.2608337
## 3  -0.8652300
## 4   0.2587184
## 5   1.0967773
## 6  -0.5423281
## 7   1.3252205
## 8   0.7383331
## 9  -0.1152228
## 10 -0.2457786
```

```
df$a <- cut(df$a,breaks = 3, labels = c("low","medium","high"))
```

```
df
```

```
##         a
## 1     low
## 2  medium
## 3     low
## 4  medium
## 5    high
## 6     low
## 7    high
## 8    high
## 9  medium
```

```
## 10 medium
```

```
## [1] 2
```

Chapter 7 : ENCODING : One Hot Encoding/dummy variables.

Dummy or Indicator Variables and their use in Regression Model. we can include categorical or quanlitative variables, also known as factors, in a regression model using dummy or indicator variables.

STEP 1 : Manual

Create or Read data set

```
id <- factor(1:10)
height <- c(175 + nrow(10) * 10)
Nationality <- c("AUS","UK","NZ","NZ","AUS","UK","NZ","UK","NZ","NZ")
```

```
dummies <- data.frame(cbind(id,height,Nationality))
```

```
dummies
```

```
##    id Nationality
## 1   1         AUS
## 2   2          UK
## 3   3          NZ
## 4   4          NZ
## 5   5         AUS
## 6   6          UK
## 7   7          NZ
## 8   8          UK
## 9   9          NZ
## 10 10          NZ
```

using for for loop we can convert categorical variable to Numerical variables

```
for (i in unique(dummies$Nationality)) {dummies[paste("Nationality",i,sep = "_")] <- ifelse(dummies$Nati
```

```
dummies
```

```
##    id Nationality Nationality_AUS Nationality_UK Nationality_NZ
## 1   1         AUS               1              0              0
## 2   2          UK               0              1              0
## 3   3          NZ               0              0              1
## 4   4          NZ               0              0              1
## 5   5         AUS               1              0              0
## 6   6          UK               0              1              0
## 7   7          NZ               0              0              1
## 8   8          UK               0              1              0
## 9   9          NZ               0              0              1
## 10 10          NZ               0              0              1
```

STEP 2 : Using caret package

```
customers <- data.frame(id=c(10, 20, 30, 40, 50),gender=c('male', 'female', 'female', 'male', 'female')
  outcome=c(1, 1, 0, 0, 0))
```

```
customers
```

```
##   id gender  mood outcome
## 1 10   male happy       1
## 2 20 female   sad       1
## 3 30 female happy       0
## 4 40   male   sad       0
## 5 50 female happy       0
```

dummify the data.

```
dmy <- dummyVars(" ~ .", data = customers)
trsf <- data.frame(predict(dmy, newdata = customers))
print(trsf)
```

```
##   id gender.female gender.male mood.happy mood.sad outcome
## 1 10             0           1          1        0       1
## 2 20             1           0          0        1       1
## 3 30             1           0          1        0       0
## 4 40             0           1          0        1       0
## 5 50             1           0          1        0       0
```

STEP 3 : Using dummies package

Choosing which variables to create dummies for

To create dummies only for one variable or a subset of variables, we can use the names argument to specify the column names of the variables we want dummies for: (students.new1 <- dummy.data.frame(students, names = c("State","Gender") , sep = ".")

```
df1 <- data.frame(id = c(1:4),year=c(1991:1994))
df1
```

```
##   id year
## 1  1 1991
## 2  2 1992
## 3  3 1993
## 4  4 1994
```

```
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
df1 <- cbind(df1,dummy(df1$year,sep="_"))
df1
```

```
##   id year /cloud/project/Data_Preparation.Rmd_1991
## 1  1 1991                                        1
## 2  2 1992                                        0
## 3  3 1993                                        0
```

```
## 4  4 1994                                        0
##   /cloud/project/Data_Preparation.Rmd_1992
## 1                                         0
## 2                                         1
## 3                                         0
## 4                                         0
##   /cloud/project/Data_Preparation.Rmd_1993
## 1                                         0
## 2                                         0
## 3                                         1
## 4                                         0
##   /cloud/project/Data_Preparation.Rmd_1994
## 1                                         0
## 2                                         0
## 3                                         0
## 4                                         1
```

STEP 4 : Using dplyr & tidyr package

```
df2 <- data.frame(id = c(1:4),year=c(1991:1994))
df2
```

```
##   id year
## 1  1 1991
## 2  2 1992
## 3  3 1993
## 4  4 1994
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:mice':
##
##     complete
```

```r
df2 %>% mutate(v = 1,yr = year) %>% spread(yr,v,fill=0)
```

```
##   id year 1991 1992 1993 1994
## 1  1 1991    1    0    0    0
## 2  2 1992    0    1    0    0
## 3  3 1993    0    0    1    0
## 4  4 1994    0    0    0    1
```

STEP 5 : Using mlr package

```r
#library(mlr)
#df3 <- data.frame(var = sample(c("A", "B", "C"), 10, replace = TRUE))
#df3
```

```r
#c <- createDummyFeatures(df, cols = "var")
```

```
## [1] 2
```

Chapter 7 : Multicollinearity

What is multicollinearity?

-> Moderate to high intercorrelations among independent variables.

What problems multicollinearity creates?

-> if two independent variables contain essentially some information to a large extent, one gains little by using both in the regression model.

-> Multicollinearity leads to unstable estimates as it tends to increase the variances of regression coefficients.

How to assess presence of multicollinearity?

-> One way is to obtain variance inflation factor (VIF)

- `VIF > 10 indicates presence of multicollinearity.`

Example with R

- R Package : faraway • Dataset : divusa • Dependent variable : divorce • Independent variables : unemployed, femlab, marriage, birth, & military

```r
library(faraway)
```

```
##
## Attaching package: 'faraway'

## The following object is masked from 'package:psych':
##
##     logit

## The following object is masked from 'package:mice':
##
##     mammalsleep

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```r
data("divusa")
```

```r
head(divusa)
```

```
##   year divorce unemployed femlab marriage birth military
## 1 1920     8.0        5.2  22.70     92.0 117.9   3.2247
## 2 1921     7.2       11.7  22.79     83.0 119.8   3.5614
## 3 1922     6.6        6.7  22.88     79.7 111.2   2.4553
## 4 1923     7.1        2.4  22.97     85.2 110.5   2.2065
## 5 1924     7.2        5.0  23.06     80.3 110.9   2.2889
```

```
## 6 1925     7.2         3.2  23.15     79.2 106.6   2.1735
```

```r
mydata <- data.frame(divusa[,-1]) #get rid of year column
```

```r
head(mydata)
```

```
##   divorce unemployed femlab marriage birth military
## 1     8.0        5.2  22.70     92.0 117.9   3.2247
## 2     7.2       11.7  22.79     83.0 119.8   3.5614
## 3     6.6        6.7  22.88     79.7 111.2   2.4553
## 4     7.1        2.4  22.97     85.2 110.5   2.2065
## 5     7.2        5.0  23.06     80.3 110.9   2.2889
## 6     7.2        3.2  23.15     79.2 106.6   2.1735
```

```r
round(cor(mydata),2)
```

```
##            divorce unemployed femlab marriage birth military
## divorce       1.00      -0.21   0.91    -0.53 -0.72     0.02
## unemployed   -0.21       1.00  -0.26    -0.27 -0.31    -0.40
## femlab        0.91      -0.26   1.00    -0.65 -0.60     0.05
## marriage     -0.53      -0.27  -0.65     1.00  0.67     0.26
## birth        -0.72      -0.31  -0.60     0.67  1.00     0.14
## military      0.02      -0.40   0.05     0.26  0.14     1.00
```

Using Multiple linear Regression

```r
mymodel <- lm(divorce  ~., data = mydata)
```

```r
mymodel
```

```
## 
## Call:
## lm(formula = divorce ~ ., data = mydata)
## 
## Coefficients:
## (Intercept)   unemployed       femlab     marriage        birth  
##     2.48784     -0.11125      0.38365      0.11867     -0.12996  
##    military  
##    -0.02673
```

```r
summary(mymodel)
```

```
## 
## Call:
## lm(formula = divorce ~ ., data = mydata)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -3.8611 -0.8916 -0.0496  0.8650  3.8300 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  2.48784    3.39378   0.733   0.4659    
## unemployed  -0.11125    0.05592  -1.989   0.0505 .  
## femlab       0.38365    0.03059  12.543  < 2e-16 ***
## marriage     0.11867    0.02441   4.861 6.77e-06 ***
```

```
## birth       -0.12996    0.01560  -8.333 4.03e-12 ***
## military    -0.02673    0.01425  -1.876   0.0647 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.65 on 71 degrees of freedom
## Multiple R-squared:  0.9208, Adjusted R-squared:  0.9152
## F-statistic: 165.1 on 5 and 71 DF,  p-value: < 2.2e-16
```

Let's check if we have multi-collinearity problem or not

```
vif(mymodel)
```

```
## unemployed     femlab   marriage      birth   military
##   2.252888   3.613276   2.864864   2.585485   1.249596
```

our above values in this model are less than 5. for sever multicollineary problem, model value should have more than 10 but since for all 5 independent variables the values are much smaller than 10, so we can safly conclude that we don't have multicollinear problem.