StudentMarks.R

sharmaam

Sun Aug 7 13:28:11 2016

```
library(party)
```

```
## Loading required package: grid
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
rm(list=ls())

# Set the working directory
setwd('.')

# Load the dataframe from the CSV file
stMarksDF = read.csv('Students_Marks_2.csv')

stMarksDF$Roll.No. = NULL

# Check the data frame
head(stMarksDF)
```

```
##   MEFA EDC DS PS MFCS DLD EDC.LAB DS.LAB PCS.LAB
## 1   51  33 49 50   56  48      71     67      69
## 2   48  46 47 44   67  56      70     69      67
## 3   58  62 81 67   65  71      73     73      67
## 4   43  41 42 41   64  43      69     68      69
## 5   49  50 56 63   67  74      70     71      70
## 6   62  59 59 63   92  75      75     75      67
```

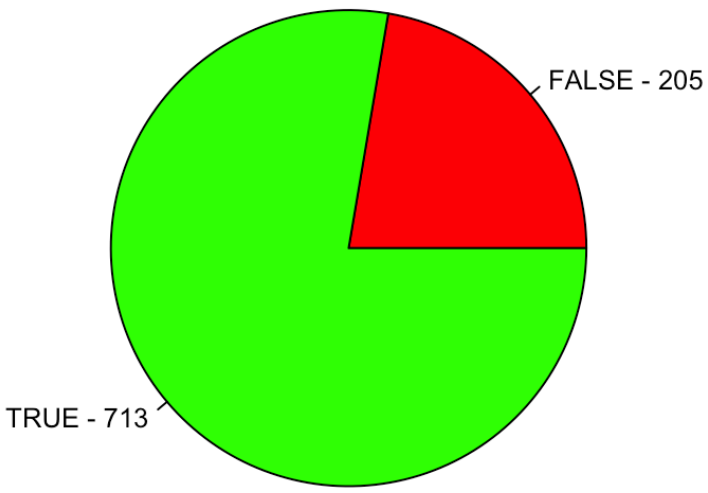
```
# Check the datatypes of marks DF
str(stMarksDF)
```

```
## 'data.frame':    103 obs. of  9 variables:
##  $ MEFA   : int  51 48 58 43 49 62 64 51 42 59 ...
##  $ EDC    : int  33 46 62 41 50 59 63 62 25 34 ...
##  $ DS     : int  49 47 81 42 56 59 60 56 46 49 ...
##  $ PS     : int  50 44 67 41 63 63 49 76 31 40 ...
##  $ MFCS   : int  56 67 65 64 67 92 49 79 53 73 ...
##  $ DLD    : int  48 56 71 43 74 75 74 73 20 60 ...
##  $ EDC.LAB: int  71 70 73 69 70 75 71 75 64 71 ...
##  $ DS.LAB : int  67 69 73 68 71 75 70 75 68 71 ...
##  $ PCS.LAB: int  69 67 67 69 70 67 66 66 67 65 ...
```

```
# Remove NA
stMarksDF = stMarksDF[complete.cases(stMarksDF),]


# Take out Roll Nos for the time being
marksDF = stMarksDF #subset(stMarksDF, select=-c(Roll.No.))
# Summarize the data in the marks DF
summary(marksDF)
```

```
##       MEFA            EDC              DS              PS
##  Min.   :13.00   Min.   : 6.00   Min.   :12.00   Min.   : 4.0
##  1st Qu.:33.00   1st Qu.:22.00   1st Qu.:40.00   1st Qu.:25.0
##  Median :45.50   Median :31.00   Median :46.00   Median :40.0
##  Mean   :43.07   Mean   :32.59   Mean   :44.73   Mean   :37.6
##  3rd Qu.:51.75   3rd Qu.:44.00   3rd Qu.:54.00   3rd Qu.:49.0
##  Max.   :65.00   Max.   :75.00   Max.   :81.00   Max.   :76.0
##       MFCS            DLD           EDC.LAB          DS.LAB
##  Min.   : 0.00   Min.   : 3.00   Min.   :42.00   Min.   :52.00
##  1st Qu.:42.00   1st Qu.:34.50   1st Qu.:62.25   1st Qu.:68.00
##  Median :52.00   Median :48.00   Median :67.00   Median :70.50
##  Mean   :51.05   Mean   :46.68   Mean   :65.33   Mean   :69.25
##  3rd Qu.:62.75   3rd Qu.:62.00   3rd Qu.:71.00   3rd Qu.:72.00
##  Max.   :92.00   Max.   :84.00   Max.   :75.00   Max.   :75.00
##     PCS.LAB
##  Min.   :54.00
##  1st Qu.:62.00
##  Median :64.00
##  Mean   :64.48
##  3rd Qu.:67.75
##  Max.   :72.00
```

```
# Tabulate pass vs fail
passFail = table(marksDF>=40)
pie(passFail, col = c('red', 'green'), main="Passed Vs Failed", cex=0.8,
    labels=paste0(names(passFail), " - ", passFail))
```



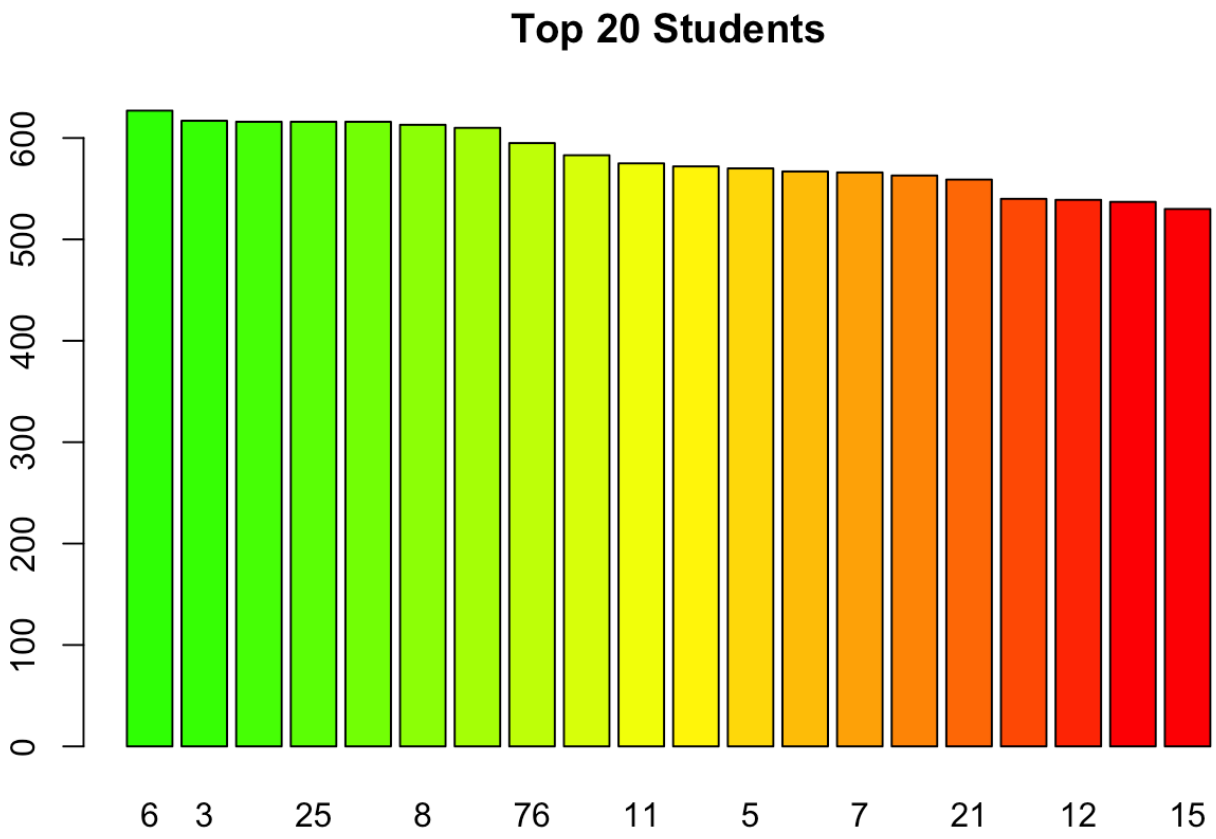

```
# Find total Marks for each student
stTotalMarks = apply(marksDF,1,sum)

# Max total marks
max(stTotalMarks)
```

```
## [1] 627
```

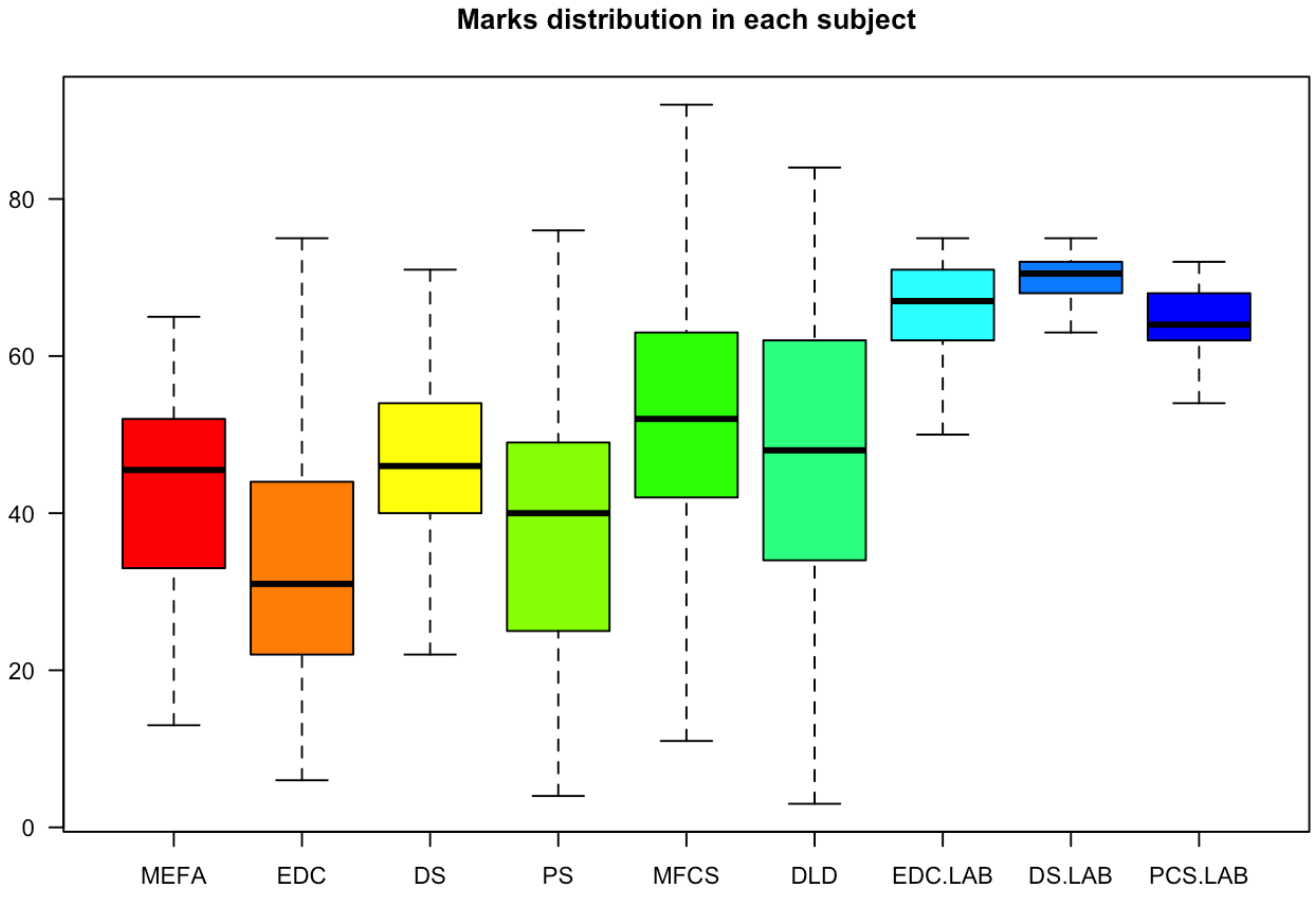
```
which(stTotalMarks == max(stTotalMarks))
```

```
## 6
## 6
```

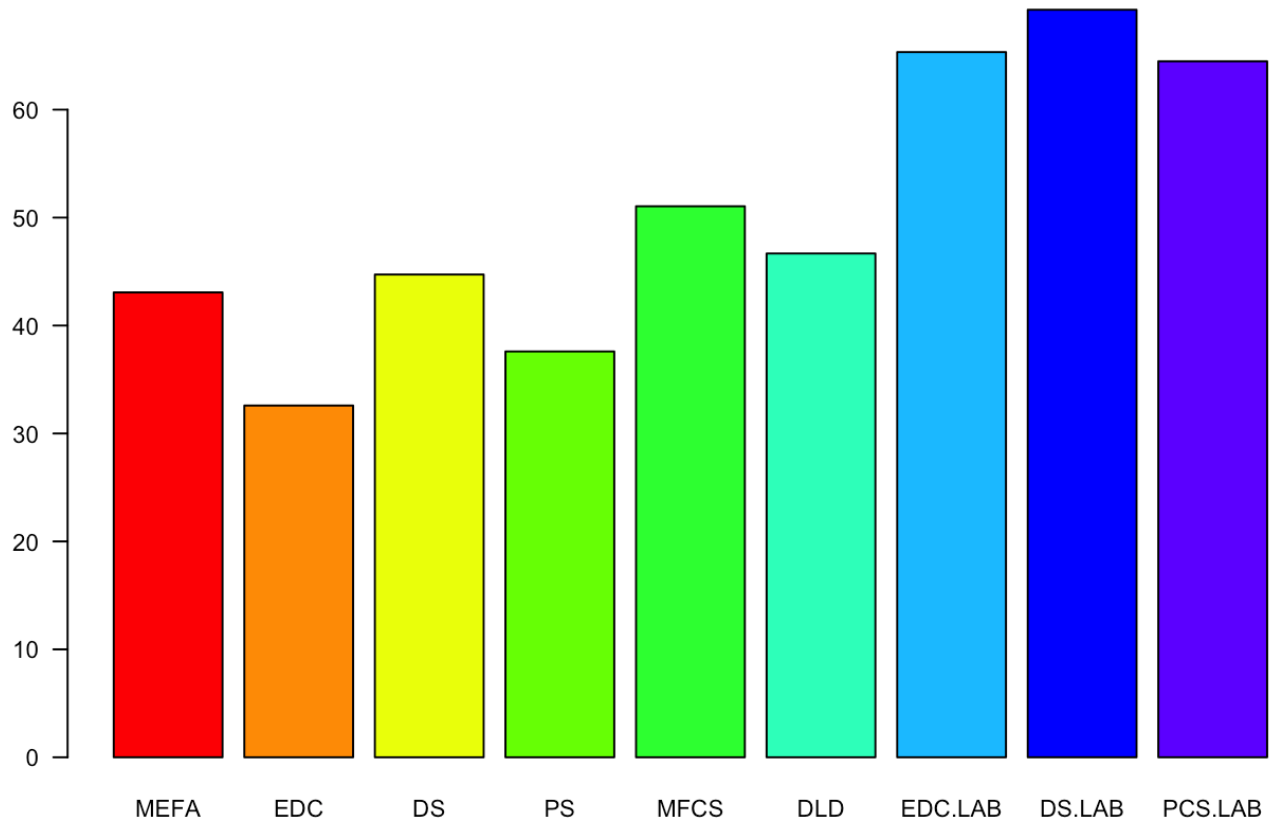
```
# Find the topper
# Show only one plot in the entire plot area
par(mfrow=c(1,1))
colfunc <- colorRampPalette(c("green", "yellow", "red"))
N = 20
stRank = sort(stTotalMarks, decreasing = T)
barplot(stRank[1:N], main="Top 20 Students",
        names=stMarksDF[names(stRank[1:N]),]$Roll.No., col=colfunc(N))
```



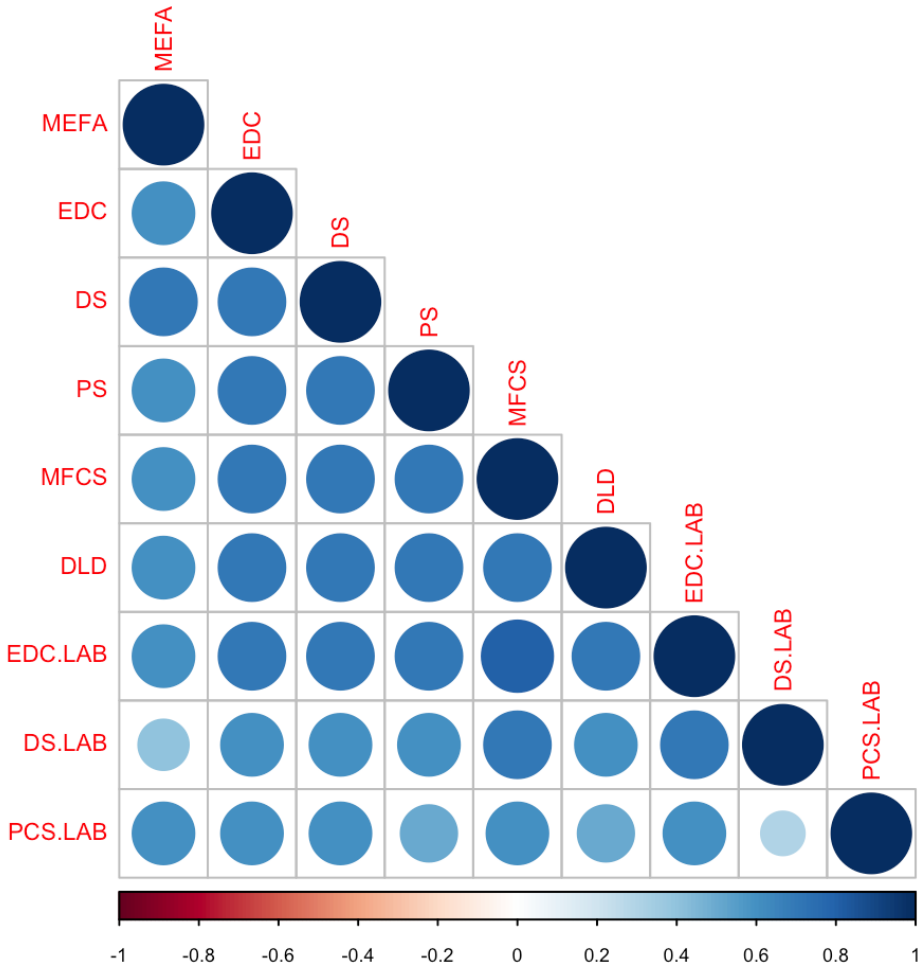

```
# Set plot window to single chart
par(mfrow=c(1,1), cex=0.7, las=1)
# Checking range and variance of marks across subjects
boxplot(marksDF, outline=F, main="Marks distribution in each subject", col=rainbo
w(12:14))
```



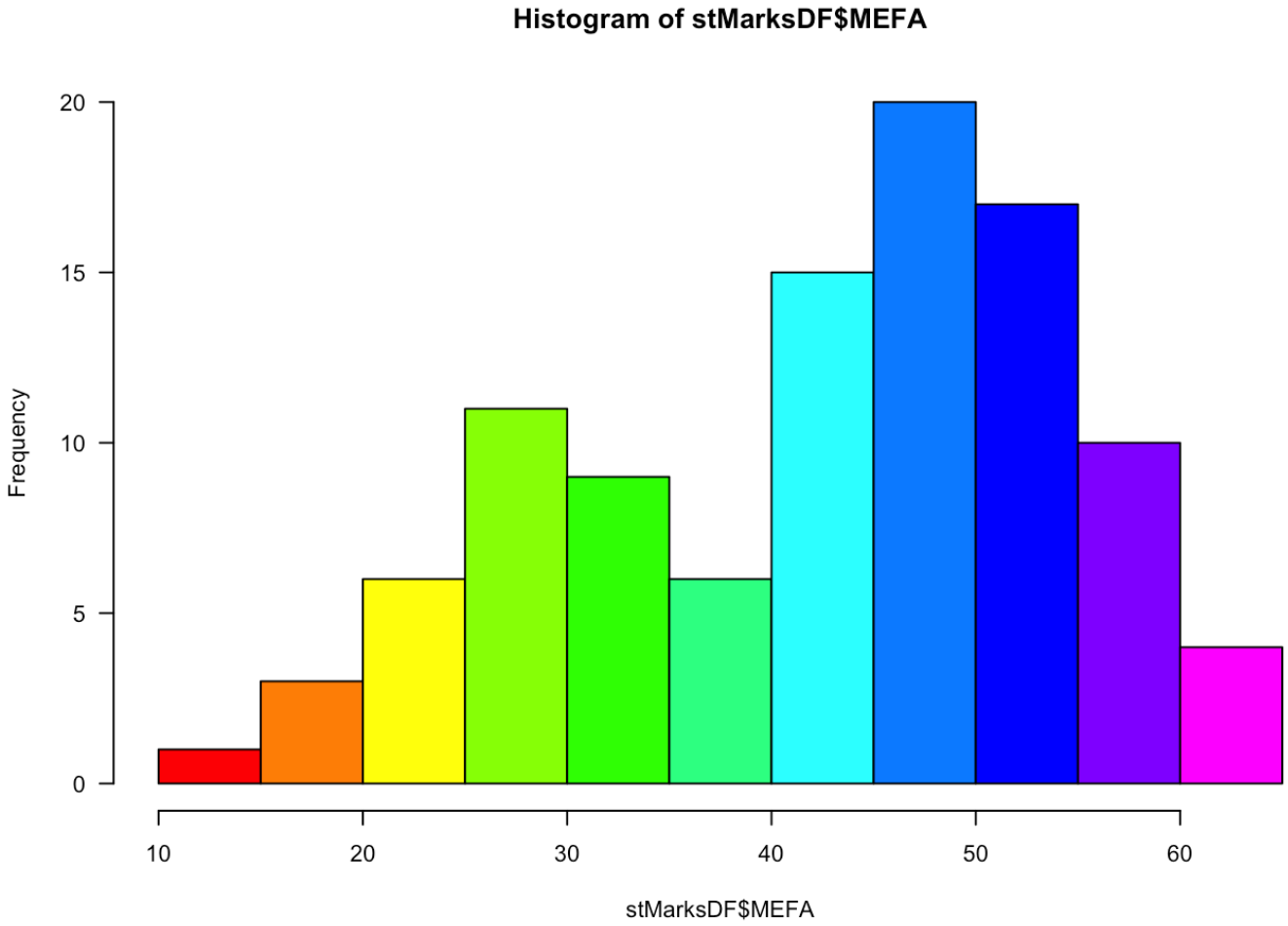

```
# Check the mean of each subject
barplot(sapply(marksDF, mean), main="Avg Marks in each Subject", col=rainbow(11:1
4))
```



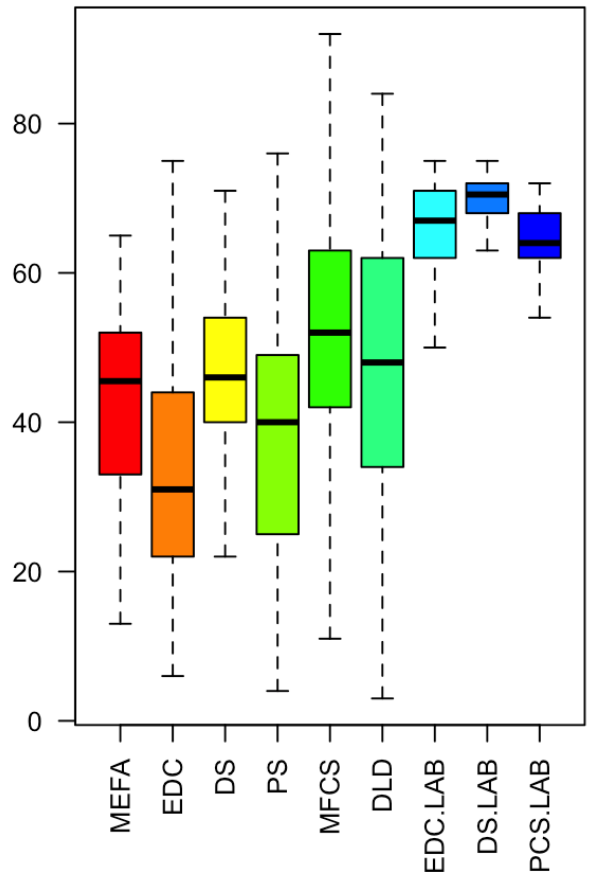

```
# Check correlation of marks
cr = round(cor(marksDF), 1)

# Using library corrplot
library(corrplot)
corrplot(cr, type="lower")
```

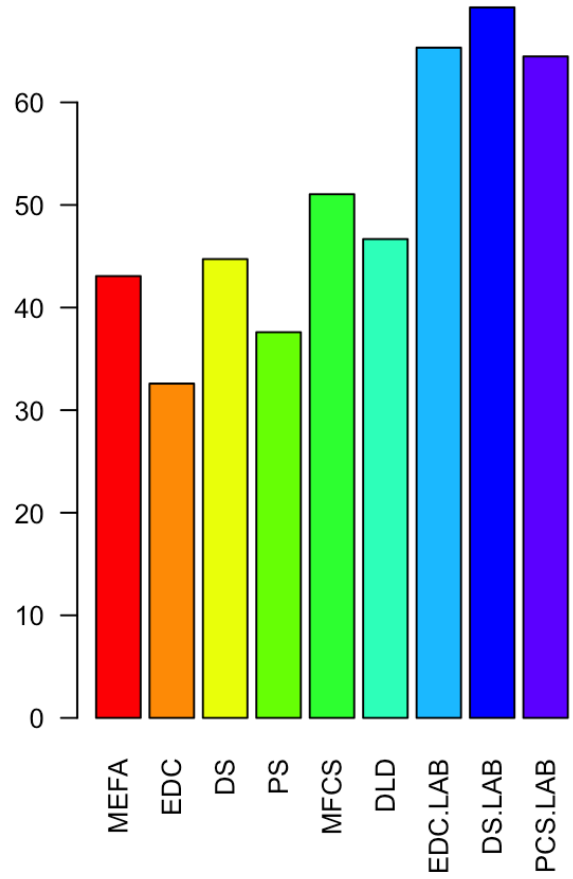



```
# Check correlation plot for mtcars
# corrplot(cor(mtcars), type="lower", method="circle")

# Histogram plots
hist(stMarksDF$MEFA, col=rainbow(12:14))
```




```
# Combining plots
par(mfrow=c(1,2), cex=0.8, las=2)
boxplot(marksDF, outline=F, col=rainbow(12:14))
barplot(sapply(marksDF, mean), col=rainbow(11:14))
```






```
#########################
# REGRESSSION ANALYSIS  #
#########################

# Lets take some students as test set and remaining as test
N = nrow(marksDF)
# Randomly sample 90% of records to be used for train
trainRows = sample(1:N, N*0.9, replace=F)
# Use the sample to separate train/test from df
train = marksDF[trainRows,]
test = marksDF[-trainRows,]

# Now lets fit the model
head(train)
```

```
##    MEFA EDC DS PS MFCS DLD EDC.LAB DS.LAB PCS.LAB
## 95   28  24 28 25   53  23      68     69      62
## 5    49  50 56 63   67  74      70     71      70
## 14   47  39 40 36   52  75      72     71      66
## 52   32  25 45 40   51  51      45     70      62
## 13   58  50 71 32   55  29      70     68      68
## 57   54  49 49 50   59  43      72     73      70
```

```
# Let's make PS score as output variable and all others input
model = lm(MFCS~.,train)
summary(model)
```

```
##
## Call:
## lm(formula = MFCS ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.3554  -5.6645  -0.4787   7.0682  23.6132
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -74.68059   33.08425  -2.257  0.02665 *
## MEFA          0.03548    0.13465   0.264  0.79281
## EDC           0.11626    0.11024   1.055  0.29469
## DS            0.18105    0.14108   1.283  0.20300
## PS            0.23802    0.09602   2.479  0.01523 *
## DLD           0.07971    0.08844   0.901  0.37007
## EDC.LAB       0.65742    0.24864   2.644  0.00981 **
## DS.LAB        0.51444    0.35328   1.456  0.14916
## PCS.LAB       0.32392    0.41280   0.785  0.43489
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.814 on 82 degrees of freedom
## Multiple R-squared:  0.7606, Adjusted R-squared:  0.7372
## F-statistic: 32.56 on 8 and 82 DF,  p-value: < 2.2e-16
```

```
preds = predict(model, newdata = test)
res = data.frame(test$MFCS, round(preds,0))
names(res) = c("Actual", "Pred")
res
```

```
##    Actual Pred
## 2      67   59
## 8      79   78
## 11     64   72
## 16     73   71
## 32     61   56
## 51     44   44
## 63     58   56
## 77     48   56
## 83     52   36
## 88     56   52
## 91     41   45
```

```
mse = sum((res$Actual-res$Pred)^2)/nrow(res)
mse
```

```
## [1] 46.72727
```

```
# Is the MSE any better than mean of train output
sum((res$Actual-mean(train$MFCS))^2)/nrow(res)
```

```
## [1] 197.5132
```

```
##############################
# CLASSIFICATION TECHNIQUES #
##############################

# Lets create some data for classification algorithm
df = marksDF

# Lets find a good cutoff for marks to be classified as Great Score (1) or not (0)
cutoffMarks = 70
summary(df$MFCS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   42.00   52.00   51.05   62.75   92.00
```

```
table(df$MFCS>cutoffMarks)
```

```
##
## FALSE  TRUE
##    86    16
```

```
# Set the new column to be 1 if MFCS is greater than 70 else 0
df$MFCS_Great_Score = ifelse(marksDF$MFCS>cutoffMarks, 1, 0)
df$MFCS_Great_Score = as.factor(df$MFCS_Great_Score)
# Now we shall remove the MFCS col to remove direct calculation
df$MFCS = NULL

# Lets take some students as test set and remaining as test
# Using library caTools to create a uniform train/test
library(caTools)
trainRows = sample.split(df$MFCS_Great_Score, SplitRatio=0.9)
train = df[trainRows,]
test  = df[!trainRows,]

############################
# 1- LOGISTIC REGRESSSION  #
############################

model <- glm(MFCS_Great_Score ~.,family=binomial(link='logit'),data=train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

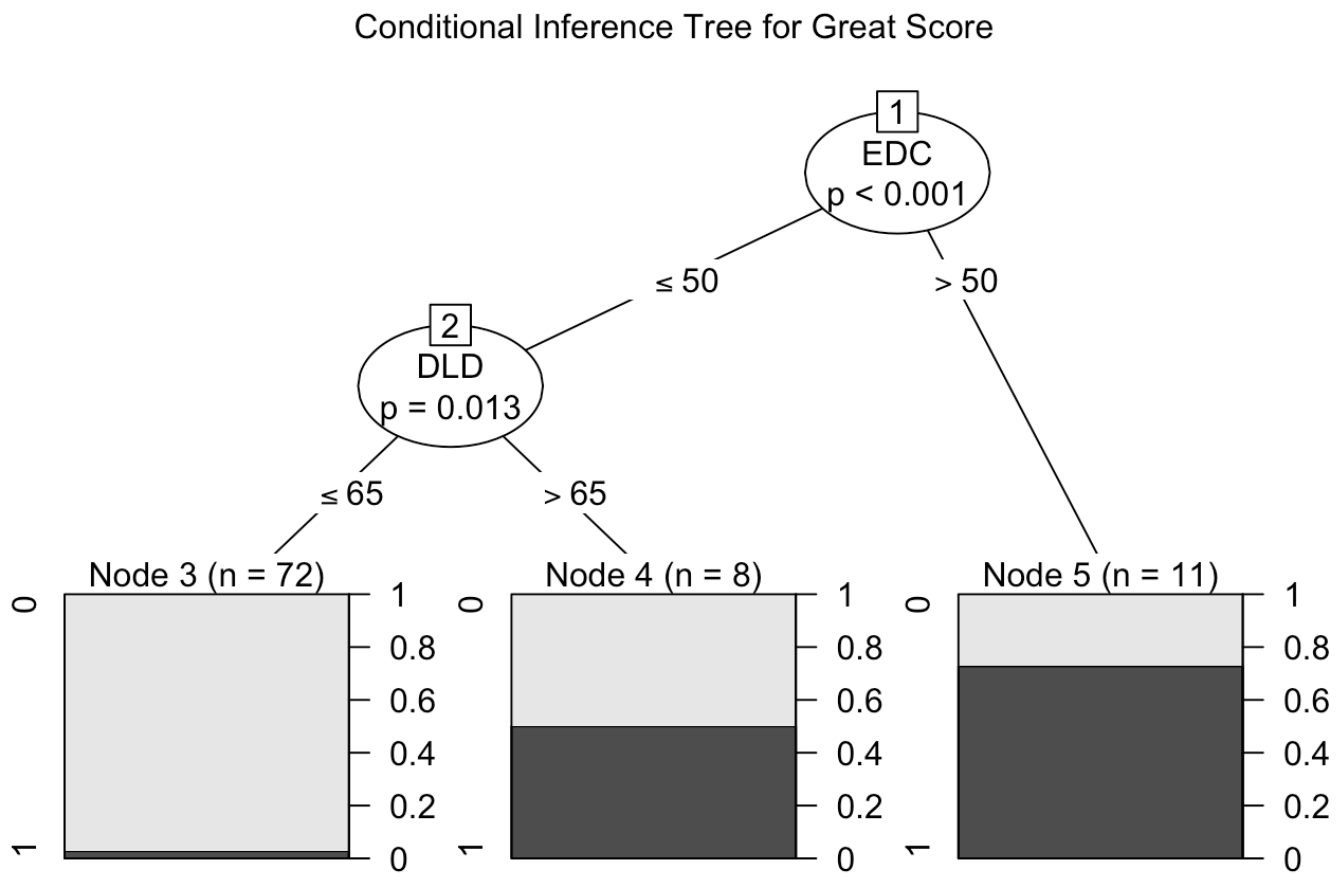
```
summary(model)
```

```
##
## Call:
## glm(formula = MFCS_Great_Score ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.36164  -0.08307  -0.00436   0.00000   1.96617
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -110.61929   43.86688  -2.522   0.0117 *
## MEFA           0.03278    0.07128   0.460   0.6456
## EDC           -0.05051    0.07053  -0.716   0.4739
## DS            -0.05927    0.08248  -0.719   0.4724
## PS             0.06432    0.06660   0.966   0.3342
## DLD            0.06744    0.06550   1.030   0.3032
## EDC.LAB        1.13487    0.53133   2.136   0.0327 *
## DS.LAB         0.25636    0.41249   0.621   0.5343
## PCS.LAB        0.09577    0.30051   0.319   0.7500
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 78.137  on 90  degrees of freedom
## Residual deviance: 22.701  on 82  degrees of freedom
## AIC: 40.701
##
## Number of Fisher Scoring iterations: 9
```

```
preds = predict(model, newdata = test, type = "response")
res = data.frame(test$MFCS_Great_Score, round(preds,0))
names(res) = c("Actual", "Pred")
res
```

```
##     Actual Pred
## 2        0    0
## 8        1    1
## 9        0    0
## 12       1    1
## 40       0    0
## 64       0    0
## 75       0    0
## 85       0    0
## 87       0    0
## 94       0    0
## 101      0    0
```

```
#################################
#        DECISION TREE          #
#################################
# We can use the same train/test as above (from Logistic regression)
# Conditional Inference Tree
fit <- ctree(MFCS_Great_Score ~.,data=train)
# Lets print the tree to see what it has learnt
plot(fit, main="Conditional Inference Tree for Great Score")
```




```
preds = predict(fit, newdata = test, type = "response")
res = data.frame(test$MFCS_Great_Score, preds)
names(res) = c("Actual", "Pred")
res
```

```
##    Actual Pred
## 1       0    0
## 2       1    1
## 3       0    0
## 4       1    0
## 5       0    0
## 6       0    0
## 7       0    0
## 8       0    0
## 9       0    0
## 10      0    0
## 11      0    0
```