##-----------------------PCA---##

library(data.table)

Creating new data frame for numerical values for doing PCA

tablechurnPCA = data.frame(tablechurn$MonthlyCharges,tablechurn$TotalCharges,tablechurn$Tenure)

setDT(tablechurnPCA)

##Setting column names for our new dataframe

names(tablechurnPCA) <- c('MonthlyCharges','TotalCharges','Tenure')

##head(tablechurnPCA)

tablechurnPCA

```
MonthlyCharges TotalCharges Tenure
   1:           29.85        29.85      1
   2:           56.95      1889.50     34
   3:           53.85       108.15      2
   4:           42.30      1840.75     45
   5:           70.70       151.65      2
  ---
7028:           84.80      1990.50     24
7029:          103.20      7362.90     72
7030:           29.60       346.45     11
7031:           74.40       306.60      4
7032:          105.65      6844.50     66
```

####Using prcomp function to compute the principal components (eigenvalues and eigenvectors).

#With scale=TRUE, variable means are set to zero, and variances set to one

ChurnPC <- prcomp(tablechurnPCA[,1:3],scale=TRUE)

ChurnPC

```
Standard deviations (1, .., p=3):
```

```
[1] 1.4764138 0.8722163 0.2438052

Rotation (n x k) = (3 x 3):
                     PC1         PC2        PC3
MonthlyCharges 0.4857136  0.79237469  0.3690862
TotalCharges   0.6650968 -0.06101971 -0.7442600
Tenure         0.5672112 -0.60697524  0.5566440
```

summary(ChurnPC)

```
Importance of components:
                          PC1    PC2     PC3
Standard deviation     1.4764 0.8722 0.24381
Proportion of Variance 0.7266 0.2536 0.01981
Cumulative Proportion  0.7266 0.9802 1.00000
```

##Steps to check variances between the principal components

##Sample scores are stored in ChurnPC$x

Square roots of eigenvalues are stored in ChurnPC$sdev

Eigenvectors are stored in ChurnPC$rotation

variable means stored in ChurnPC$center

variable standard deviations stored in ChurnPC$scale

Eigenvalues are sdev^2

##Calculating Eigen values

(eigen_churn <- ChurnPC$sdev^2)

```
[1] 2.1797978 0.7607612 0.0594410
```

names(eigen_churn) <- paste("PC",1:3,sep="")

names(eigen_churn)

```
[1] "PC1" "PC2" "PC3"
```

eigen_churn

```
   PC1       PC2       PC3
2.1797978 0.7607612 0.0594410
```

##Taking sum of all eigen values

sum_churn <- sum(eigen_churn)

sum_churn

```
[1] 3
```

##Calculating percentage of variance

pvarchurn <- (eigen_churn/sum_churn)*100

pvarchurn

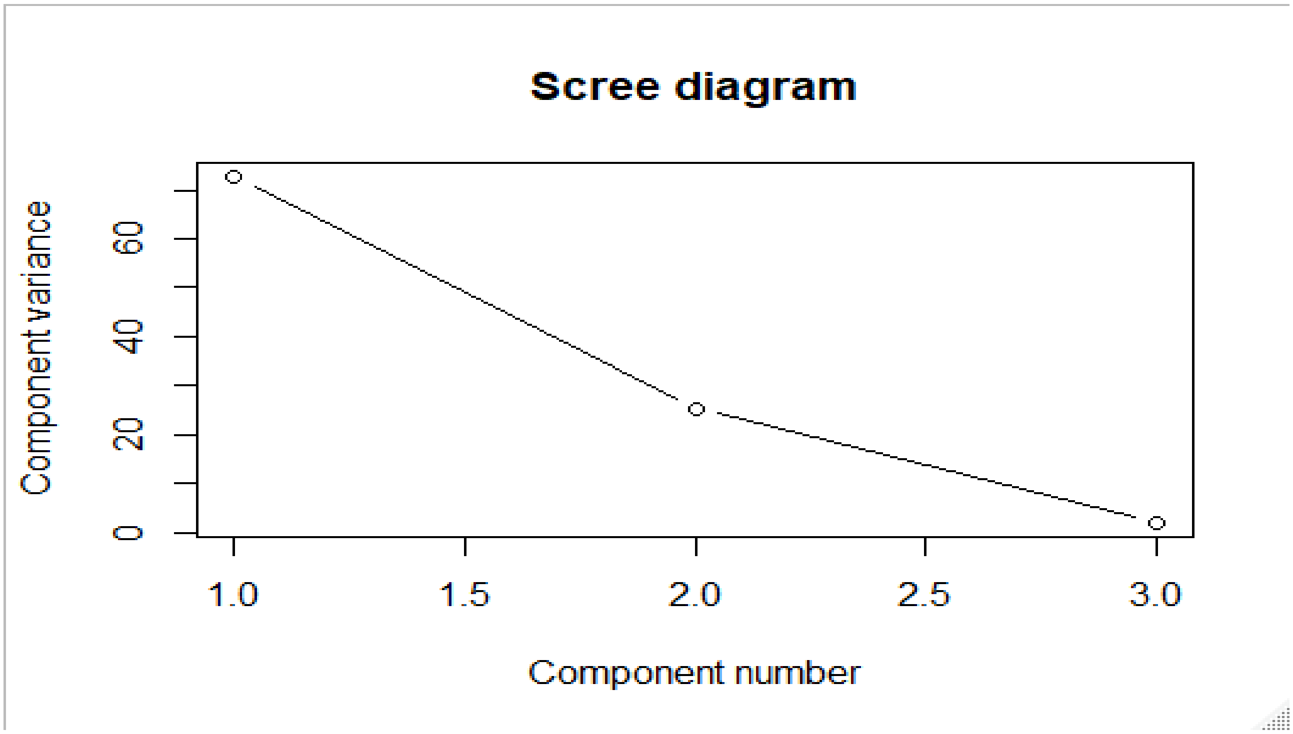
```
PC1       PC2       PC3
72.659927 25.358707  1.981367
```

##The results show that PC1 and PC2 columns have good amount of data that is almost 98% variance

#is presented with these 2 columns and since there is only 2% information in PC3 component we can afford to lose it

##Plotting Scree Diagram for Principal Components wrt percentage of their variances

plot(pvarchurn, xlab = "Component number", ylab = "Component variance", type = "b", main = "Scree diagram")



##Checking cumulative variances and other values

cumvar_churn <- cumsum(pvarchurn)

cumvar_churn

```
     PC1       PC2       PC3
 72.65993  98.01863 100.00000
```

matchurn <- rbind(eigen_churn,pvarchurn,cumvar_churn)

rownames(matchurn) <- c("Eigenvalues","Prop. variance","Cum. prop. variance")

round(matchurn,4)

```
                        PC1     PC2      PC3
Eigenvalues          2.1798  0.7608   0.0594
Prop. variance      72.6599 25.3587   1.9814
Cum. prop. variance 72.6599 98.0186 100.0000
```

summary(ChurnPC)

```
Importance of components:
                          PC1    PC2     PC3
Standard deviation     1.4764 0.8722 0.24381
Proportion of Variance 0.7266 0.2536 0.01981
Cumulative Proportion  0.7266 0.9802 1.00000
```

ChurnPC$rotation

```
                     PC1         PC2        PC3
MonthlyCharges 0.4857136  0.79237469  0.3690862
TotalCharges   0.6650968 -0.06101971 -0.7442600
Tenure         0.5672112 -0.60697524  0.5566440
```

print(ChurnPC)

```
Standard deviations (1, .., p=3):
[1] 1.4764138 0.8722163 0.2438052

Rotation (n x k) = (3 x 3):
                     PC1         PC2        PC3
MonthlyCharges 0.4857136  0.79237469  0.3690862
TotalCharges   0.6650968 -0.06101971 -0.7442600
Tenure         0.5672112 -0.60697524  0.5566440
```

ChurnPC$x

sparrtyp_pca <- cbind(data.frame(tablechurn$Churn),ChurnPC$x)

sparrtyp_pca

Means of scores for all the PC's classified by Churn value

tabmeansPC <- aggregate(sparrtyp_pca[,2:4],by=list(Churn=tablechurn$Churn),mean)

tabmeansPC

tabmeansPC <- tabmeansPC[rev(order(tabmeansPC$Churn)),]

tabmeansPC

tabfmeans <- t(tabmeansPC[,-1])

tabfmeans

colnames(tabfmeans) <- t(as.vector(tabmeansPC[1]))

```
tabfmeans

# Standard deviations of scores for all the PC's classified by Churn value

tabsdsPC <- aggregate(sparrtyp_pca[,2:4],by=list(Churn=tablechurn$Churn),sd)
tabfsds <- t(tabsdsPC[,-1])
colnames(tabfsds) <- t(as.vector(tabsdsPC[1]))
tabfsds

#T-Test

t.test(PC1~tablechurn$Churn,data=sparrtyp_pca)
t.test(PC2~tablechurn$Churn,data=sparrtyp_pca)
t.test(PC3~tablechurn$Churn,data=sparrtyp_pca)

# F ratio test

var.test(PC1~tablechurn$Churn,data=sparrtyp_pca)
var.test(PC2~tablechurn$Churn,data=sparrtyp_pca)
var.test(PC3~tablechurn$Churn,data=sparrtyp_pca)

# Levene's tests (one-sided)

library(car)
(LTPC1 <- leveneTest(PC1~tablechurn$Churn,data=sparrtyp_pca))
(p_PC1_1sided <- LTPC1[[3]][1]/2)
(LTPC2 <- leveneTest(PC2~tablechurn$Churn,data=sparrtyp_pca))
(p_PC2_1sided=LTPC2[[3]][1]/2)
(LTPC3 <- leveneTest(PC2~tablechurn$Churn,data=sparrtyp_pca))
```

```
(p_PC3_1sided <- LTPC3[[3]][1]/2)

# Plotting the scores for the first and second PC components

plot(sparrtyp_pca$PC1, sparrtyp_pca$PC2,pch=ifelse(sparrtyp_pca$Churn == "Yes",1,16),xlab="PC1", ylab="PC2", main="7043 customer values for PC1 & PC2")

abline(h=0)

abline(v=0)

legend("bottomleft", legend=c("Yes","No"), pch=c(1,16))

plot(eigen_churn, xlab = "Component number", ylab = "Component variance", type = "l", main = "Scree diagram")

plot(log(eigen_churn), xlab = "Component number",ylab = "log(Component variance)", type="l",main = "Log(eigenvalue) diagram")

print(summary(ChurnPC))

View(ChurnPC)

diag(cov(ChurnPC$x))

xlim <- range(ChurnPC$x[,1])

ChurnPC$x[,1]

ChurnPC$x

plot(ChurnPC$x,xlim=xlim,ylim=xlim)

ChurnPC$rotation[,1]

ChurnPC$rotation

ChurnPC$x

plot(ChurnPC)

#get the original value of the data based on PCA

center <- ChurnPC$center

scale <- ChurnPC$scale

new_ChurnPC <- as.matrix(tablechurnPCA[,-3])

new_ChurnPC

predict(ChurnPC)[,1]
```

```
##------------------------------Cluster Analysis-----------------------------------##

library(cluster)
tablechurnclust = data.frame(
  tablechurn$Tenure,
  tablechurn$TotalCharges,
  tablechurn$MonthlyCharges)

##Making cluster on the basis of Total charges
rownames(tablechurnclust) <- tablechurn$TotalCharges

##Scaling done to make the data on scale
scaleTotalCharges <- scale(tablechurnclust[,1:ncol(tablechurnclust)])
scaleTotalCharges

#Here we have selected first row to see how our scaled matrix is like
head(scaleTotalCharges,1)

# We will find K-means by taking k=2, 3, 4, 5, 6...

#For k-means = 2
(kmeans2.scaleTotalCharges <- kmeans(scaleTotalCharges,2,nstart = 10))
# Computing the percentage of variation accounted for two clusters
perc_var_kmeans2 <- round(100*(1 -
kmeans2.scaleTotalCharges$betweenss/kmeans2.scaleTotalCharges$totss),1)
names(perc_var_kmeans2) <- "Perc. 2 clus"
perc_var_kmeans2
```

```
#For  k-means = 3

(kmeans3.scaleTotalCharges <- kmeans(scaleTotalCharges,3,nstart = 10))

# Computing the percentage of variation accounted for. Two clusters

perc_var_kmeans3 <- round(100*(1 - kmeans3.scaleTotalCharges$betweenss/kmeans3.scaleTotalCharges$totss),1)

names(perc_var_kmeans3) <- "Perc. 3 clus"

perc_var_kmeans3


#For  k-means = 4

(kmeans4.scaleTotalCharges <- kmeans(scaleTotalCharges,4,nstart = 10))

# Computing the percentage of variation accounted for. Two clusters

perc_var_kmeans4 <- round(100*(1 - kmeans4.scaleTotalCharges$betweenss/kmeans4.scaleTotalCharges$totss),1)

names(perc_var_kmeans4) <- "Perc. 4 clus"

perc_var_kmeans4


#Using k means 4 could be good to preseent our data

# Saving above 4 k-means (1,2,3)  in a list



##Now we will plot these clusters

library(fpc)

plotcluster(tablechurnclust,kmeans3.scaleTotalCharges$cluster)


##We didnt find any significant result using cluster analysis hence we will not be using it

#for our dataset
```