STAT 6340 Statistical Machine Learning

Mini Project 5

Author:
Suarez Rodes, Randy

Supervisor:
Choudhary, Pankaj Ph.D.

April 19, 2021

1 Answers

Question 1.a We present the mean and standard deviation of the quantitative features on Table 1. The features present very different scales, therefore we recommend standardizing the data before carrying out principal components analysis.

	Mean	SD
AtBat	403.64	147.31
Hits	107.83	45.13
HmRun	11.62	8.76
Runs	54.75	25.54
RBI	51.49	25.88
Walks	41.11	21.72
Years	7.31	4.79
CAtBat	2657.54	2286.58
CHits	722.19	648.20
CHmRun	69.24	82.20
CRuns	361.22	331.20
CRBI	330.42	323.37
CWalks	260.27	264.06
PutOuts	290.71	279.93
Assists	118.76	145.08
Errors	8.59	6.61

Table 1: Summary

Question 1.b We performed PCA on standardized data. The obtained variances, proportion of variance explained (PVE) and the cumulative PVE for each principal component are presented on Table 2. We also graphed the PVE and Cummulative PVE on Figure 1 and 2 respectively. We consider 5 principal components are appropriate, explaining close to 84% of the variation.

PC	Variance	PVE	CumPVE
1	7.2797	0.3831	0.3831
2	4.1498	0.2184	0.6016
3	2.0304	0.1069	0.7084
4	1.5566	0.0819	0.7903
5	0.9987	0.0526	0.8429
6	0.8255	0.0434	0.8863
7	0.6893	0.0363	0.9226
8	0.5130	0.0270	0.9496
9	0.2507	0.0132	0.9628
10	0.1848	0.0097	0.9726
11	0.1372	0.0072	0.9798
12	0.1275	0.0067	0.9865
13	0.0956	0.0050	0.9915
14	0.0610	0.0032	0.9947
15	0.0520	0.0027	0.9975
16	0.0280	0.0015	0.9989
17	0.0141	0.0007	0.9997
18	0.0049	0.0003	0.9999
19	0.0012	0.0001	1.0000

Table 2: Variance, PVE Cummulative PVE by Components

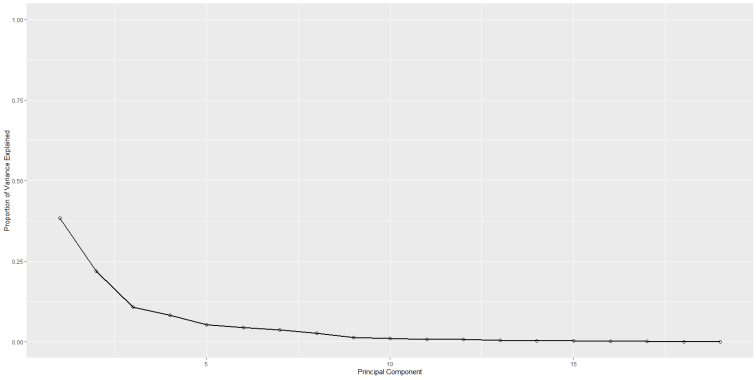

Figure 1: PVE

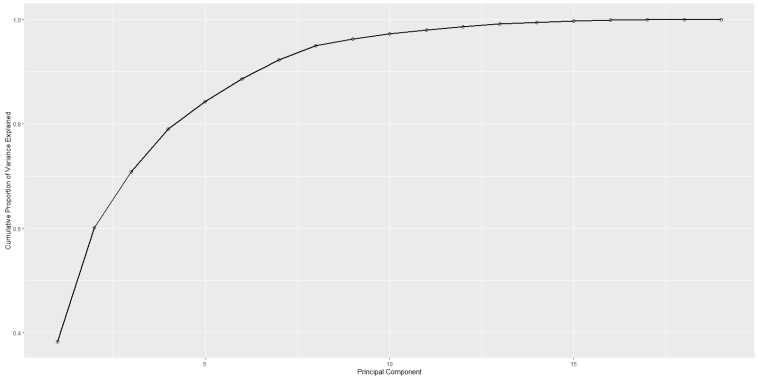

Figure 2: Cummulative PVE

Question 1.c We present the correlations of the standardized quantitative variables with the first two components in Table 3 and the scores on the components and the loadings on the biplot of Figure 3. We can see that the first principal component is mostly determined by variables of the career statistics (CAtBat, Chits, etc) while the second principal component is mostly determined by the variables of the current season performance. The career statistics are correlated as expected and load more heavily on the first principal component while the current season statistics are also correlated and load closer to the second component.

	PC1	PC2
AtBat	0.5350	0.7818
Hits	0.5285	0.7685
HmRun	0.5514	0.4831
Runs	0.5351	0.7695
RBI	0.6345	0.6407
Walks	0.5637	0.4677
Years	0.7624	-0.5345
CAtBat	0.8916	-0.3930
CHits	0.8924	-0.3726
CHmRun	0.8606	-0.2573
CRuns	0.9125	-0.3509
CRBI	0.9183	-0.3424
CWalks	0.8548	-0.3918
PutOuts	0.2096	0.3173
Assists	-0.0023	0.3436
Errors	-0.0212	0.4090

Table 3: Correlations PCs and Quantitative Variables

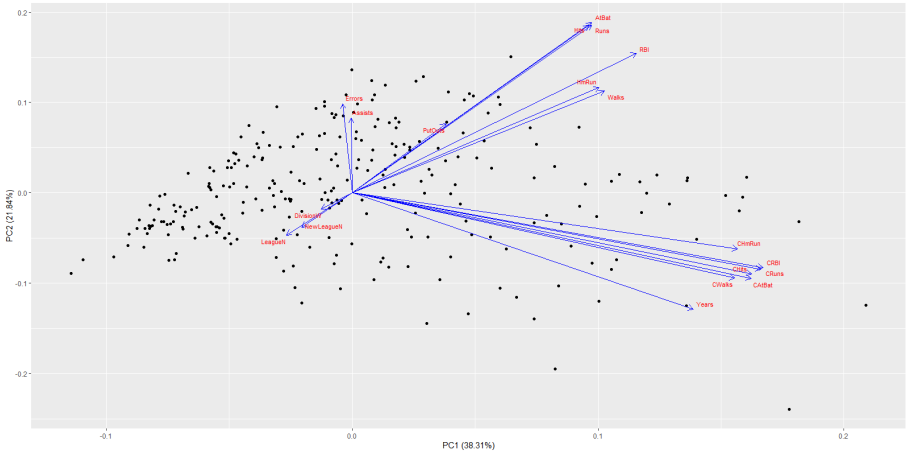

Figure 3: PVE

Question 2.a In general standardizing the data is a good idea because otherwise the scale of values in each feature can act as a weight to determine how to cluster the data. Concretely, a feature in a very different scale may lead to counter intuitive distances that may fail to group the data correctly.

Question 2.b Our interest is to stablish a difference in the performance between the players. As an example, we might be interested in figuring out if players with very different amount of hits in a season will be in the same cluster. To answer questions like this a geometrical distance (e.g. Euclidean distance) is appropriate, since it will figure out how far (different) are the two players. If we use a correlation-based distance, we will consider two players to be similar if their features are highly correlated, even if their values may be far apart in terms of Euclidean distance, which will not be useful for our data.

Question 2.c After standardizing the data we hierarchically clustered the players using complete linkage and Euclidean distance. We present the dendrogram of this process on Figure 4 (just a subset of the labels is displayed for quality visualization purposes).

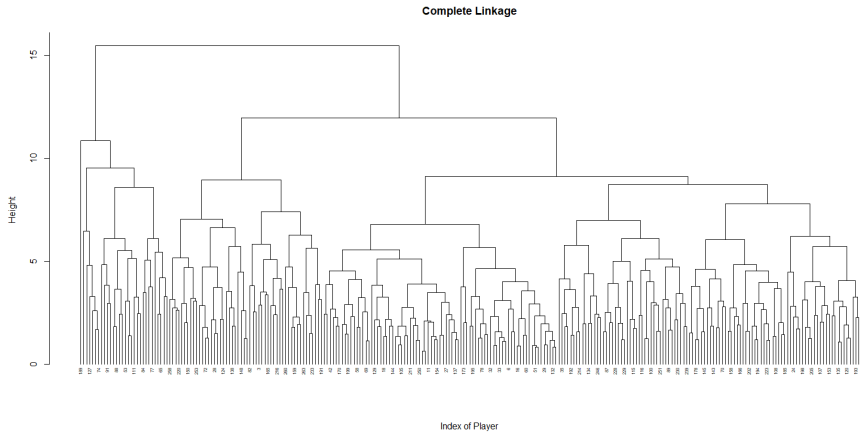

Figure 4: Dendrogram of Complete Linkage

We present on Table 4 the means of each cluster for the quantitative variables and salary (standardized and non standardized centers are presented). Observe how the means for the career statistics on cluster 2 are considerably higher than in cluster 1, hinting how these two clusters separate the players by their experience. However cluster 2 only groups 30 from the 263 observations, therefore the salary mean can be affected by the high salaries of top tier veteran players (this situation may also occur for other variables). As we know in reality, the salary for a player will not necessary increase with his career.

	C1	C2	C1.std	C2.std
AtBat	402.3734	413.5000	-0.0086	0.0669
Hits	107.6052	109.5667	-0.0050	0.0385
HmRun	11.0730	15.8667	-0.0624	0.4850
Runs	54.4764	56.8333	-0.0105	0.0818
RBI	49.8584	64.1333	-0.0629	0.4886
Walks	39.9657	50.0333	-0.0529	0.4107
Years	6.2146	15.8333	-0.2289	1.7777
CAtBat	2068.3648	7233.5000	-0.2577	2.0012
CHits	555.8927	2013.7333	-0.2565	1.9925
CHmRun	47.4893	238.1667	-0.2646	2.0551
CRuns	275.5150	1026.8667	-0.2588	2.0098
CRBI	239.7597	1034.5333	-0.2804	2.1774
CWalks	191.9270	791.0333	-0.2588	2.0101
PutOuts	282.5107	354.4000	-0.0293	0.2275
Assists	125.1030	69.5000	0.0437	-0.3395
Errors	8.7468	7.4000	0.0233	-0.1806
Salary	479.9491	970.6788	-0.1241	0.9637

Table 4: Hierarchical Clusters Means

	C1	C2	C1.std	C2.std
AtBat	377.5397	470.3108	-0.1772	0.4526
Hits	99.4868	129.1351	-0.1849	0.4722
HmRun	9.4127	17.2568	-0.2520	0.6437
Runs	49.5344	68.0541	-0.2040	0.5211
RBI	44.8148	68.5270	-0.2578	0.6584
Walks	35.9577	54.2838	-0.2374	0.6064
Years	5.3016	12.4459	-0.4193	1.0710
CAtBat	1571.8889	5430.3649	-0.4748	1.2126
CHits	414.7354	1507.4324	-0.4743	1.2114
CHmRun	31.3069	166.1216	-0.4615	1.1786
CRuns	199.6825	773.7973	-0.4877	1.2457
CRBI	171.3545	736.6757	-0.4919	1.2563
CWalks	135.6138	578.6351	-0.4721	1.2057
PutOuts	265.8201	354.2838	-0.0889	0.2271
Assists	125.2646	102.1486	0.0448	-0.1145
Errors	8.9153	7.7703	0.0488	-0.1246
Salary	368.5547	963.4010	-0.3710	0.9476

Table 5: KM Clusters Means

Question 2.d We run the KMeans algorithm with K=2 obtaining 189 observations for cluster 1 and 74 for cluster 2. We present on Table 5 the means of each clusters for the quantitative variables and salary (standardized and non standardized centers are presented). Similar to hierarchical clustering the two cluster obtained separate the players by experience. This time we obtained a slightly more equitable partition of the two clusters, leading to the results to be less affected by outliers.

Question 2.e We present a graph of the clusters using the variables Salary vs CRBI grouped by the hierarchical and KMeans clusters, Figures 5 and 6 respectively. The centers are represented in a different shape. Both algorithms clustered the players based on their experience, this might lead to false ideas like a player will improve the performance over time. The KMeans clusters are more equitably distributed being less susceptible to outliers that might cause such problems, therefore we consider KMeans to be more appropriate for the Hitters data.

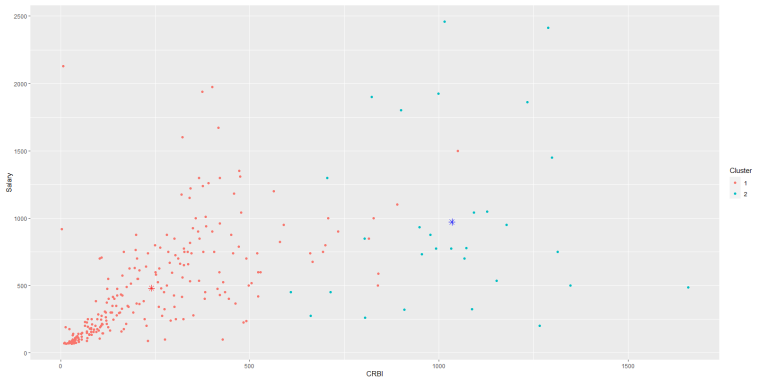

Figure 5: Hierarchical Clustering

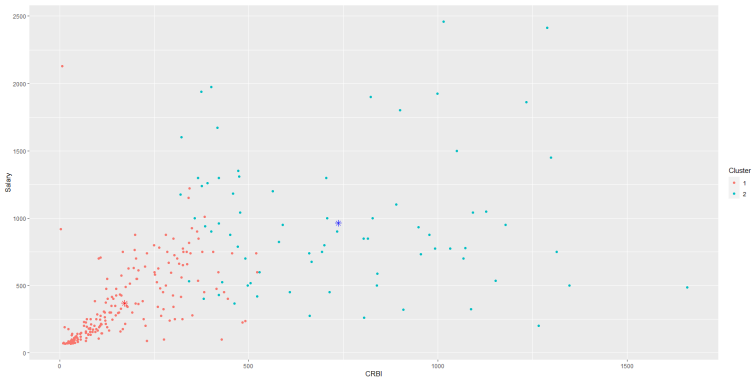

Figure 6: KMeans Clustering

Question 3.a We run a LOOCV on a full regression model obtaining an estimated test MSE of 0.4214063.

Question 3.b Using LOOCV we obtained the optimal PCR model with 16 components (Figure 7) and an estimated test MSE of 0.4104077.

Question 3.c Using LOOCV we obtained the optimal PLS model with 12 components (Figure 8) and an estimated test MSE of 0.4134874.

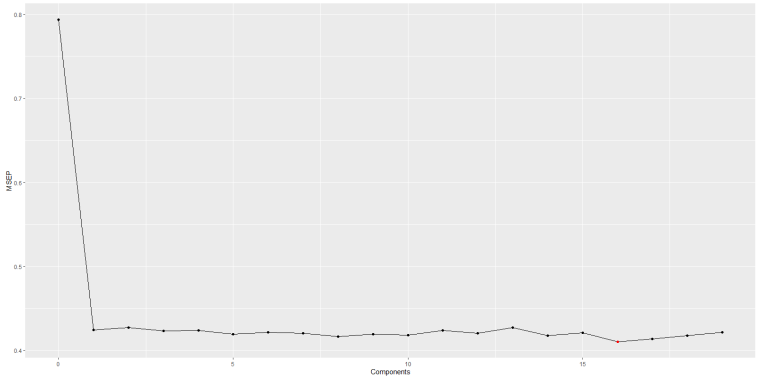

Figure 7: Validation Plot PCR

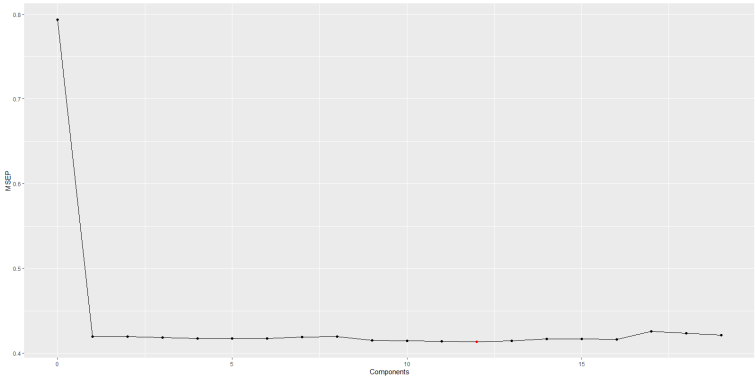

Figure 8: Validation Plot PLS

Question 3.d We run LOOCV to find the best lambda parameter for Ridge Regression. We used a grid of 100 possible lambdas ranging from 10^{-3} to 10^{10}. The results of this process can be found on Figure 9. The minimal estimated test MSE = 0.4083535 was obtained for lambda = 0.09326033.

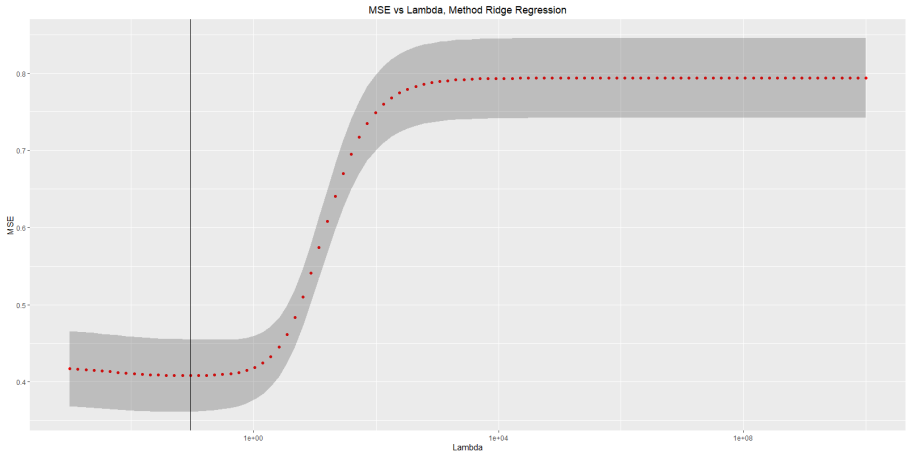

Figure 9: MSE vs Lambda

Question 3.e We present on Table 6 a summary of the results of LOOCV for the different models. All the models are very close in MSE. Since the models are close in flexibility, ridge regression is our choice having the slight least MSE.

Model	MSE
Full	0.421406
PCR	0.410408
PLS	0.413487
RidgeReg	0.408353

Table 6: Results of LOOCV applied on the different models

2 Code

```
#Packages
library(ggplot2) #Used for graphics and visual representations
library(ggfortify)  #Used to generate biplot ggplot object
library(caret) #Used to handle LOOCV training
library(glmnet) #Use for Ridge Regression
library(broom) #To create tidy objects for ggplot visualization
library(pls) #Used for PCR and PLS
library(ISLR) #Library for the data


rdseed=8466 #Seed to replicate results


#Experiment 1
print("Experiment 1")

#Importing data and removing NA values
Hitters=na.omit(Hitters)

#Number of observations
n=nrow(Hitters)

####Question 1.a####

#Exploring the data
summary(Hitters[,-c(14,15,19,20)])

df.stats=data.frame(Mean=apply(Hitters[,-c(14,15,19,20)], 2, mean),
                    SD=apply(Hitters[,-c(14,15,19,20)], 2, sd))

#Different scales observed, standardizing the data is recommended
print(df.stats)

####Question 1.b####
#Separating features and creating dummy variables
y = Hitters$Salary
x = model.matrix(Salary ~ ., Hitters)[, -1]

#Carrying out PCA
pca_x=prcomp(x,center = TRUE, scale = TRUE)

#Variance and Percent of Variance explained
pc_var=pca_x$sdev^2
pve=pc_var/sum(pc_var)

print(data.frame(Variance=pc_var, PVE=pve,CumPVE=cumsum(pve)))


g_pve=ggplot(data.frame(PC=1:ncol(x),pve=pve),aes(x=PC,y=pve))+geom_line(size=1)+
  geom_point(shape=1,size=2)+ylim (0,1)+
  xlab("Principal Component")+  ylab("Proportion of Variance Explained")

print(g_pve) #5 principal components seems to be appropriate


g_cum=ggplot(data.frame(PC=1:ncol(x),CumPVE=cumsum(pve)),aes(x=PC,y=CumPVE))+geom_line(size=1)+
  geom_point(shape=1,size=2)+
  xlab("Principal Component")+  ylab("Cumulative Proportion of Variance Explained")

print(g_cum)
```

```
#With 5 principal components we explained close to 84% of the variation


####Question 1.c####

#Correlations of quantitative variables and the first two principal components
df_corr=data.frame(PC1=pca_x$rotation[-c(14,15,19),1]*pca_x$sdev[1],
                   PC2=pca_x$rotation[-c(14,15,19),2]*pca_x$sdev[2])
print(df_corr)

#Scores
head(pca_x$x[,1:2])

#Biplot
autoplot(pca_x,loadings=TRUE, loadings.colour = 'blue',
         loadings.label = TRUE, loadings.label.size = 3,loadings.label.repel=TRUE)



#Experiment 2
print("Experiment 2")

####Question 2.a####
#Included on report

####Question 2.b####
#Included on report

####Question 2.c####

#Standardizing the data
x.std=scale(x)

x4clust=x.std

#Changing the names of players for an id for graphing pruposes
row.names(x4clust)=1:n

#Hierarchical clustering
hc.x = hclust(dist(x4clust), method = "complete")

#Subset of labels to graph
labs=hc.x$order[seq(1,n,3)]

#Use ids as labels
ids=rownames(x4clust)
#Eliminating labels not present in our selection
ids[ !(ids %in% labs) ] = ""

#Dendrogram
plot(hc.x,main = "Complete Linkage", cex = 0.5,labels=ids,xlab="Index of Player",sub="",hang = -1)


#Cutting at a height for two clusters
hc2=cutree(hc.x, 2)

#Indexes of cluster 1
c1.hc=which(hc2==1)

#Mean of standardized variables by cluster
df.means.std=data.frame(C1.std=apply(x.std[c1.hc,],2,mean),C2.std=apply(x.std[-c1.hc,],2,mean))

#Mean of variables by cluster
```

```r
df.means=data.frame(C1=apply(x[c1.hc,],2,mean),C2=apply(x[-c1.hc,],2,mean))

#Mean of Salary by cluster
df.Sal.std=data.frame(C1.std=mean(scale(y)[c1.hc]),C2.std=mean(scale(y)[-c1.hc]))
row.names(df.Sal.std)="Salary"

#Mean of Salary by cluster
df.Sal=data.frame(C1=mean(y[c1.hc]),C2=mean(y[-c1.hc]))
row.names(df.Sal)="Salary"

#Displaying quantitative variables only
print(cbind(rbind(df.means,df.Sal),rbind(df.means.std,df.Sal.std))[-c(14,15,19),])

#Some Clusters Visualization
lab=ifelse(hc2==1,"1","2")

#Salary vs CRuns
ggdf1=data.frame(CRuns=Hitters$CRuns,Salary=Hitters$Salary,Cluster=lab)
print(ggplot(data=ggdf1,aes(x=CRuns,y=Salary,color=Cluster))+geom_point()+
        geom_point(data=data.frame(CRuns=c(df.means["CRuns","C1"],
                                           df.means["CRuns","C2"]),
                                   Salary=c(df.Sal$C1,df.Sal$C2),
                                   Cluster=c("1","2")),
                   colour=c("red","blue"),size=3,shape=8))

#Salary vs CRBI
ggdf2=data.frame(CRBI=Hitters$CRBI,Salary=Hitters$Salary,Cluster=lab)
print(ggplot(data=ggdf2,aes(x=CRBI,y=Salary,color=Cluster))+geom_point()+
        geom_point(data=data.frame(CRBI=c(df.means["CRBI","C1"],
                                          df.means["CRBI","C2"]),
                                   Salary=c(df.Sal$C1,df.Sal$C2),
                                   Cluster=c("1","2")),
                   colour=c("red","blue"),size=3,shape=8))


####Question 2.d####

#Fixing seed for kmeans
set.seed(rdseed)
km2 = kmeans(x.std, centers=2, nstart = 20)

#Cluster means (standardized)
km2$centers

#Indexes of cluster 1
c1.km=which(km2$cluster==1)

#Mean of standardized variables by cluster
df.means.stdK=data.frame(C1.std=km2$centers[1,],C2.std=km2$centers[2,])

#Mean of variables by cluster
df.meansK=data.frame(C1=apply(x[c1.km,],2,mean),C2=apply(x[-c1.km,],2,mean))

#Mean of Salary by cluster
df.Sal.stdK=data.frame(C1.std=mean(scale(y)[c1.km]),C2.std=mean(scale(y)[-c1.km]))
row.names(df.Sal.std)="Salary"

#Mean of salaries by cluster
df.SalK=data.frame(C1=mean(y[c1.km]),C2=mean(y[-c1.km]))
row.names(df.SalK)="Salary"

#Displaying quantitative variables only
print(cbind(rbind(df.meansK,df.SalK),rbind(df.means.stdK,df.Sal.stdK))[-c(14,15,19),])
```

```
#Some Clusters Visualization
lab=ifelse(hc2==1,"1","2")
labkm=ifelse(km2$cluster==1,"1","2")



#Salary vs CRuns
ggdf3=data.frame(CRuns=Hitters$CRuns,Salary=Hitters$Salary,Cluster=labkm)
print(ggplot(data=ggdf3,aes(x=CRuns,y=Salary,color=Cluster))+geom_point()+
        geom_point(data=data.frame(CRuns=c(df.meansK["CRuns","C1"],
                                           df.meansK["CRuns","C2"]),
                                   Salary=c(df.SalK$C1,df.SalK$C2),
                                   Cluster=c("1","2")),
                   colour=c("red","blue"),size=3,shape=8))


dfK=
#Salary vs CRBI
ggdf4=data.frame(CRBI=Hitters$CRBI,Salary=Hitters$Salary,Cluster=labkm)
print(ggplot(data=ggdf4,aes(x=CRBI,y=Salary,color=Cluster))+geom_point()+
        geom_point(data=data.frame(CRBI=c(df.meansK["CRBI","C1"],
                                           df.meansK["CRBI","C2"]),
                                   Salary=c(df.SalK$C1,df.SalK$C2),
                                   Cluster=c("1","2")),
                   colour=c("red","blue"),size=3,shape=8))


#Experiment 3
print("Experiment 3")

#Dataframe to track errors of each model
error.df=data.frame(MSE=rep(0,4))
row.names(error.df)=c("Full","PCR","PLS","RidgeReg")

####Question 3.a####

control=trainControl(method = "LOOCV")

#LOOCV on full regression model
m_fullloocv = train(log(Salary)~.,
                    data = Hitters,
                    preProcess=c("scale"),
                    method = "lm",
                    trControl = control)

#Estimated MSE
error.df["Full","MSE"]=m_fullloocv$results$RMSE^2

####Question 3.b####
pcr.fit=pcr(log(Salary) ~ ., data = Hitters,scale = TRUE, center=TRUE,
            validation = "LOO")

#Dataframe to create validation plot
df.msep1=data.frame(Components=0:19,MSEP=MSEP(pcr.fit)$val[1, 1,])

#Validation Plot
val1=ggplot(df.msep1, aes(x=Components,y=MSEP))+geom_line()+geom_point()

#Optimal M = 16
M_pcr=which.min(MSEP(pcr.fit)$val[1, 1,])
print(val1+geom_point(data=df.msep1[M_pcr,],colour="red"))
```

```
#Estimated MSE
error.df["PCR","MSE"]=MSEP(pcr.fit)$val[1, 1,M_pcr]

#Best Model 16 Components
pcr.fitBest=pcr(log(Salary) ~ ., data = Hitters,scale = TRUE, center=TRUE,
                validation = "LOO",ncomp=M_pcr-1)
summary(pcr.fitBest)

####Question 3.c####
pls.fit=plsr(log(Salary) ~ ., data = Hitters,scale = TRUE, center=TRUE,
             validation = "LOO")

#Dataframe to create validation plot
df.msep2=data.frame(Components=0:19,MSEP=MSEP(pls.fit)$val[1, 1,])

#Validation Plot
val2=ggplot(df.msep2,aes(x=Components,y=MSEP))+geom_line()+geom_point()

#Optimal M = 12
M_pls=which.min(MSEP(pls.fit)$val[1, 1,])
print(val2+geom_point(data=df.msep2[M_pls,],colour="red"))

#Estimated MSE
error.df["PLS","MSE"]=MSEP(pls.fit)$val[1, 1,M_pls]

#Best Model 12 Components
pls.fitBest=plsr(log(Salary) ~ ., data = Hitters,scale = TRUE, center=TRUE,
                 validation = "LOO",ncomp=M_pls-1)
summary(pls.fitBest)


####Question 3.d####
lambdas = 10^seq(10, -3, length = 100)

#Applying LOOCV to find best lambda
cv.ridge = cv.glmnet(x, log(y), alpha = 0, nfolds = n,
                     lambda = lambdas, grouped = FALSE, type.measure = "mse")

#Best lambda
cv.ridge$lambda.min

#Tidy data frames to graph
tidy_cv <- tidy(cv.ridge)
glance_cv <- glance(cv.ridge)

#Plot of MSE as a function of lambda
g.ridge = ggplot(tidy_cv, aes(lambda, estimate))+
  geom_point(color="red") + scale_x_log10()+
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)+
  geom_vline(xintercept = glance_cv$lambda.min)+
  labs(x="Lambda",y="MSE")+
  theme(plot.title=element_text(hjust=0.5))+
  ggtitle("MSE vs Lambda, Method Ridge Regression")

print(g.ridge)

#Estimated MSE
error.df["RidgeReg","MSE"]=min(cv.ridge$cvm)

####Question 3.e####

print(error.df)
```