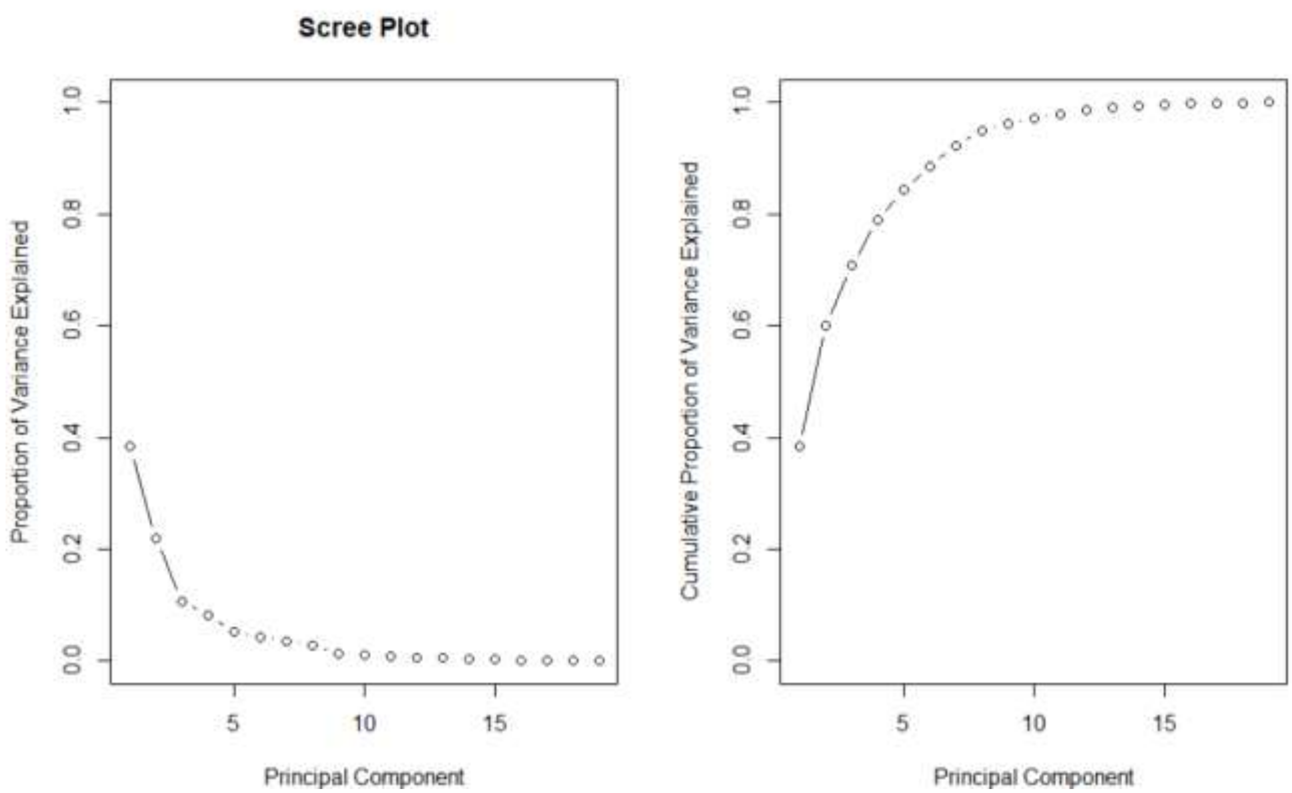
Mini Project 5 STAT6340: Statistical and Machine Learning

Section 1: Answers

Q1. The ***Hitters*** data from ISLR package has 263 observations on 19 independent variables and ***Salary*** is the response variable, others are predictor variable with ***League, Division** and **New League*** being the qualitative predictors.

(a) I examine the data using summary() for the range and mean and check the standard deviation. I opt for standardizing the predictor variables as the scales differ widely and this scaling difference may affect the results of PCA. PCA is a variance maximizing exercise and thus standardization ensures inherent scale (SD) in the data does not affect the outcome.

(b) I perform PCA on the predictor variables (19) after standardizing these and the PC loadings are computed, output attached at the end of the report here (before code) The scree plot is here, based on it I recommend 3 PCs as that point is the elbow. Together these 3 PCs explain 70.84% variation in the data.

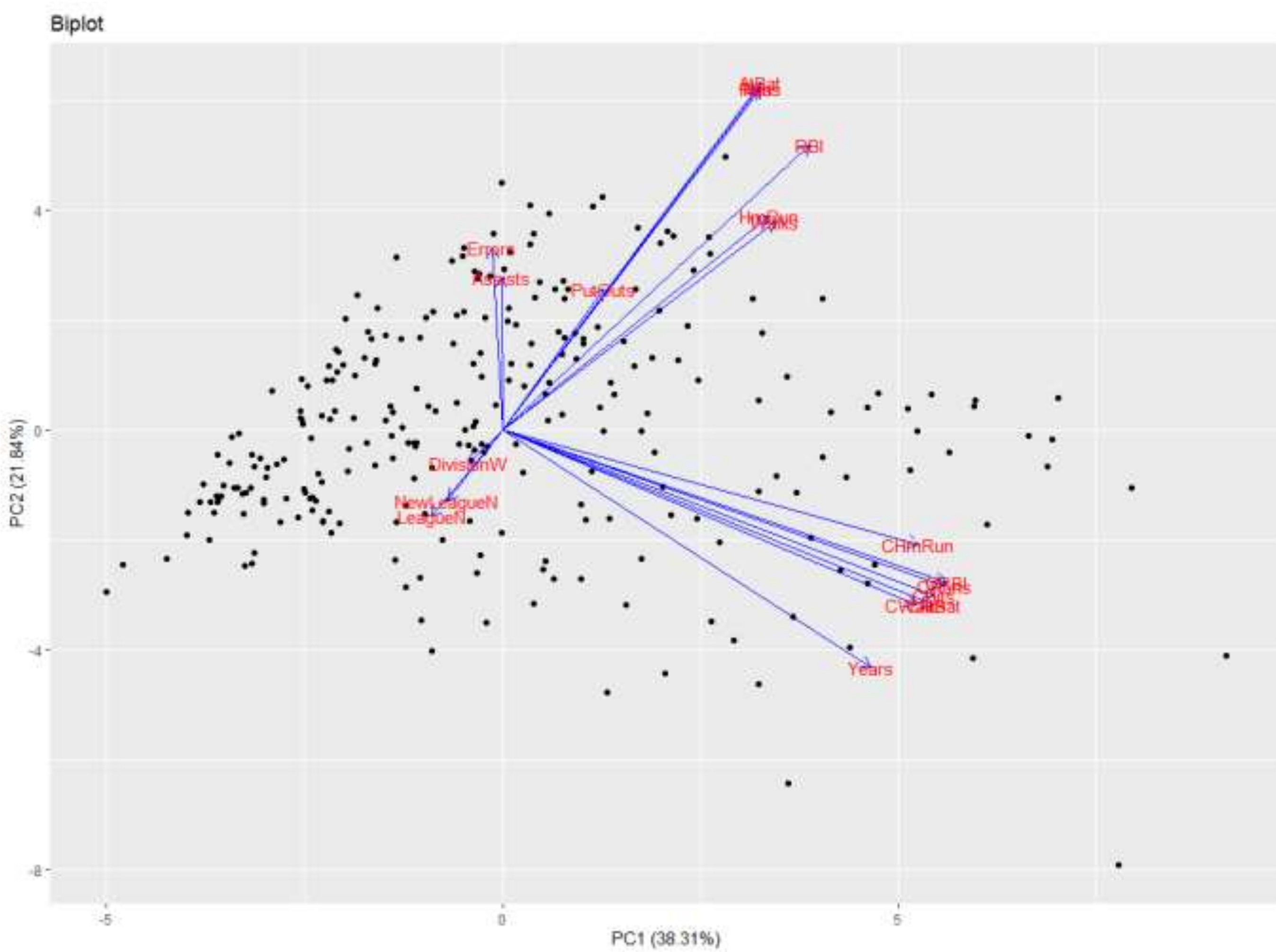


(c) Following table shows the correlation of the original variables (predictors) with the first two principal components. As seen from the biplot (next), the correlation values are the slopes of the eigenvectors with respect to each PC.

Biplot in PCA is a plot that shows the points, mapped as per the first two PC **scores** for each observation (labels, that is the row names, containing the baseball player names, is turned off to make sure the biplot is legible) and the directions of the original variables (**eigen vectors**) with respect to the first two PCs. These lines denote the relationship between the first two PCs and the original variables. Here I have used different scales for the scores and the eigen vectors with Scale=0 option in the code, following the best practice shown in the class. The magnitude of the slopes of the eigen vectors show the correlation of the original variables with the first two PCs, and this is consistent with the correlation values computed.

Correlation	PC1	PC2
AtBat	0.535006	0.781809
Hits	0.528452	0.768542
HmRun	0.551406	0.483071
Runs	0.535132	0.769459
RBI	0.634521	0.640734
Walks	0.563696	0.467732
Years	0.762415	-0.53454
CAtBat	0.89162	-0.39297
CHits	0.892372	-0.37258
CHmRun	0.860636	-0.25728

Correlation	PC1	PC2
CRuns	0.912516	-0.35095
CRBI	0.918277	-0.34242
CWalks	0.854764	-0.39177
LeagueN	-0.14697	-0.19396
DivisionW	-0.06941	-0.07472
PutOuts	0.209634	0.317252
Assists	-0.00227	0.343562
Errors	-0.02121	0.408969
NewLeagueN	-0.11308	-0.15805



Q2.

(a) The variables should be standardized before clustering as the scales of the variables differ significantly and we are yet to determine which features are most important in cluster determination of the observations. Scales of the variables affect the clustering based on Euclidean distance method.

(b) This data is for baseball players and the features (game statistics in this case) tend to move together, hence I prefer to use correlation-based method for clustering.

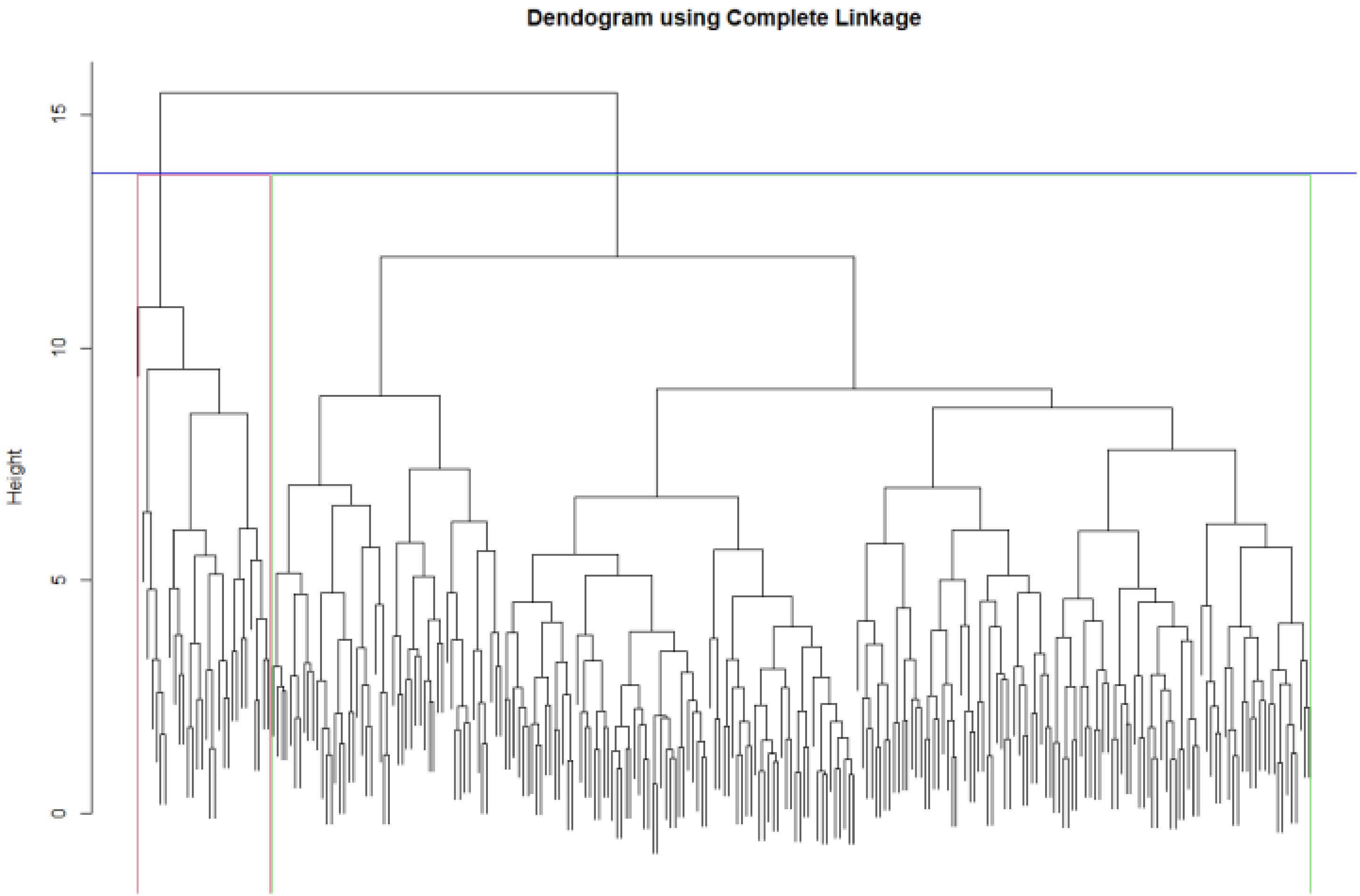
(c) Following is the Dendogram for hierarchical clustering of the data, variables centered and scaled before clustering, and I get 2 clusters by cutting it at h=13.75, the 2 clusters, with 233 and 30 observations, are encased by rectangles red and green. Following tables show the mean salary of

cluster 1 (479.949, 233 obs) and mean salary of cluster 2 (970.679, 30 obs). The means for cluster 2 is higher than that of cluster 1 observations, for all features except Errors, therefore showing that cluster 2 is a group of better players and that is supported by mean salary numbers as well.
I tried to use row indices as data labels in the Dendogram, instead of the player names, but it was still getting illegible and therefore the labels were turned off.

Means	Cluster 1	Cluster 2
#Players	233	30
AtBat	402.373	413.500
Hits	107.605	109.567
HmRun	11.073	15.867
Runs	54.476	56.833
RBI	49.858	64.133
Walks	39.966	50.033

Means	Cluster 1	Cluster 2
Years	6.215	15.833
CAtBat	2068.365	7233.500
CHits	555.893	2013.733
CHmRun	47.489	238.167
CRuns	275.515	1026.867
CRBI	239.760	1034.533
CWalks	191.927	791.033

Means	Cluster 1	Cluster 2
LeagueN	0.455	0.600
DivisionW	0.506	0.533
PutOuts	282.511	354.400
Assists	125.103	69.500
Errors	8.747	7.400
NewLeagueN	0.451	0.567
Salary	479.949	970.679



(d) Following is the summary of means from k-means clustering of the data, variables centered and scaled before clustering. The function kmeans() uses 'Hartigan-Wong' algorithm by default and thus Euclidean distance measure is used in clustering. Following tables show the mean salary of cluster 2 (368.555, 189 obs) and mean salary of cluster 1 (963.401, 74 obs). The means for cluster 1 is higher than that of cluster 1 observations, for all features except Errors, therefore showing that cluster 1 is a group of better players and that is supported by mean salary numbers as well.

Means	Cluster 2	Cluster 1
#Players	189	74
AtBat	377.540	470.311
Hits	99.487	129.135
HmRun	9.413	17.257
Runs	49.534	68.054
RBI	44.815	68.527
Walks	35.958	54.284

Means	Cluster 1	Cluster 2
Years	5.302	12.446
CAtBat	1571.889	5430.365
Chits	414.735	1507.432
CHmRun	31.307	166.122
CRuns	199.683	773.797
CRBI	171.354	736.676
CWalks	135.614	578.635

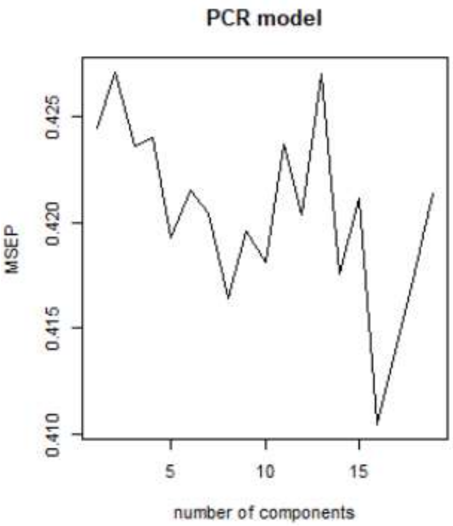
Means	Cluster 1	Cluster 2
LeagueN	0.529	0.324
DivisionW	0.550	0.405
PutOuts	265.820	354.284
Assists	125.265	102.149
Errors	8.915	7.770
NewLeagueN	0.508	0.351
Salary	368.555	963.401

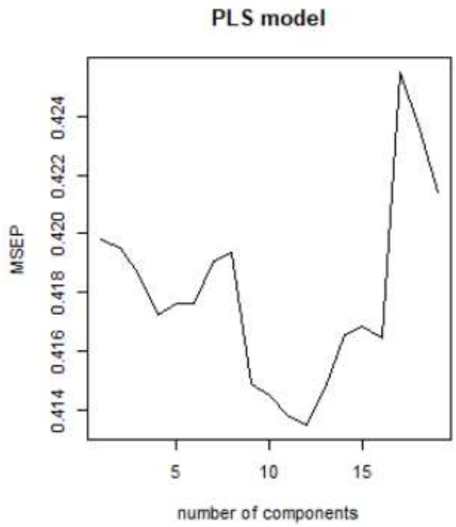
(e) Comparing the clusters created by two different methods show both times, clusters divide the players between good and bad. However, since the criteria of such classes are not defined, rather explored by the technique involved, k-means method classify more (74 as compared to 30) players as 'good' as compared to hierarchical clustering. This reflects in mean salary, hierarchical clustering has higher mean salary for 'good' players (only 30 'good' players) as compared to k-means clustering and the mean salary for 'bad' players is lesser for k-means as compared to corresponding statistic for hierarchical clustering because more 'good' players are taken out of this cluster.
As the k-means put less players in the 'bad' cluster, as compared to hierarchical clustering method, the 'bad' cluster for k-means has worse statistic for the feature means, as can be seen in the tables.

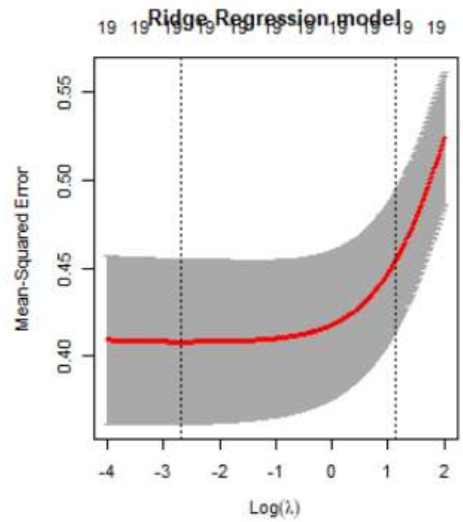
Q3. The following table provides the summary of the 4 models, the coefficients, and the test MSE (LOOCV) as the entire dataset is used to train the data. The ridge regression yields the lowest test MSE, and this is the model I recommend for future dataset on baseball players.

Regression Coefficients	**Linear Regression**	**PCR**	**PLS**	**Ridge Regression**
(Intercept)	4.618143	N/A	N/A	4.597223
AtBat	-0.00298	0.454724	-1.69982	-0.000458
Hits	0.013085	1.482605	3.026946	0.004239
HmRun	0.011793	0.363215	0.523394	0.003133
Runs	-0.00142	1.276258	0.925258	0.003103
RBI	-0.00168	0.684081	0.061207	0.000796
Walks	0.010955	1.209551	1.71794	0.00523
Years	0.056964	2.085076	2.620447	0.040817
CAtBat	0.000128	1.439658	1.048002	0.000039
CHits	-0.00044	1.615829	1.711571	0.000193
CHmRun	-7.8E-05	0.105516	0.063328	-0.000031
CRuns	0.001513	1.101849	1.401916	0.000291
CRBI	0.000131	1.149182	0.580604	0.000158
CWalks	-0.00147	0.104552	-1.92418	-0.000365
LeagueN	0.282475	0.963236	1.39015	0.200656
DivisionW	-0.16564	-1.12089	-0.98126	-0.166098
PutOuts	0.000339	1.042361	1.009995	0.000291
Assists	0.000621	0.56564	0.797369	0.000385
Errors	-0.01197	-0.7686	-0.93238	-0.011484
NewLeagueN	-0.17416	-0.28105	-0.77619	-0.101903
MSE (LOOCV)	0.4214	0.4104	0.4135	0.4083
ncomp	N/A	16	12	N/A

Plot for optimal choice for different regression models follow.







Section 2: R Code

Q1.

```
# Data input and EDA
library(ISLR)
Hitters = na.omit(Hitters) #Load the data
dim(Hitters) # check the dimensions, 263x20
str(Hitters) # examine the data
colnames(Hitters) # Salary at Col 19

# Q1.(a) code
summary(Hitters[-19]) # checking mean and quantiles
apply(Hitters[19], 2, sd) # checking SD, except the qualitative variables
# Q1.(a) code ends

# Q1.(b) code
Hitters.pca = prcomp(model.matrix(Salary ~ ., Hitters)[, -1], center = T, scale = T)
summary(Hitters.pca)

out = as.data.frame(Hitters.pca$rotation)
write.csv(out, "C:/Users/Kaustav/Downloads/STAT 6340/Projects/P5 due 19-Apr-2021/out.csv", row.names =
T)
# this csv is printed and attached at the end of report, before the code

pc.var = Hitters.pca$sdev^2 # variance of each PC
pve = pc.var/sum(pc.var) # sum(Var(PC)) = sum(Var(X)) for all p-variable

plot.new() # start a new plot frame, useful to remove bugs in plot display
par(mfrow = c(1,2))
plot(pve, main = "Scree Plot", xlab="Principal Component", ylab="Proportion of Variance Explained", ylim =
c(0,1), type = 'b')
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of Variance Explained", ylim =
c(0,1), type = 'b')
cumsum(pve)[3] # 3 PC selected, 70.84% var explained
# Q1.(b) code ends

# Q1.(c) code
PC1.score = Hitters.pca$x[,1]
PC2.score = Hitters.pca$x[,2]
X = model.matrix(Salary ~ ., Hitters)[, -1]

correl.X.PC = array(1:2, dim=c(2, ncol(X)))
for (i in 1:ncol(X)){
  correl.X.PC[1,i] = cor(X[,i], PC1.score)
  correl.X.PC[2,i] = cor(X[,i], PC2.score)
}

out = as.data.frame(X)
```

```
write.csv(out, "C:/Users/Kaustav/Downloads/STAT 6340/Projects/P5 due 19-Apr-2021/out5.csv", row.names =
F)

library(ggfortify)
autoplot(Hitters.pca, scale=0, label=FALSE, loadings = TRUE, loadings.colour = 'blue', loadings.label = TRUE,
loadings.label.size = 4, main = "Biplot")
# Scale=1 is default, and that refers to same scale for all plots. This is turned off.
# Q1.(c) code ends
```

Q2.

```
# Q2.(c) code
X = model.matrix(Salary ~ ., Hitters)[, -1] # class(X) = "matrix" "array" # WHY???
Xsc = scale(X)

## Pre-processing of data, row names replaced with row index, to make the visual representation legible
col.of.indices = seq(1:nrow(Hitters))
col.of.players = row.names(Hitters)

player.index = cbind(col.of.indices , col.of.players)
Xsc = cbind(Xsc, col.of.indices) # the index is in Column 20
row.names(Xsc) = Xsc[,20] # Declare column 20 as the row names
Xsc = Xsc[,-20] # the column of indices is removed, dim(X)= 263 by 19
rm(col.of.indices, col.of.players)
## Pre-processing of data complete.
## Skipped as vene with row indices, Dendogram is ver hard to read.

set.seed(6340)
hc.complete = hclust(dist(Xsc), method = "complete")
# hc.complete$dist.method = "euclidean" CHECKED
plot.new() # start a new plot frame, useful to remove bugs in plot display
plot(hc.complete, main = "Dendogram using Complete Linkage", xlab = "", sub = "", cex = 0.7, labels = FALSE)
# labels = FALSE removes the labels from the Dendogram, easier to read.
rect.hclust(hc.complete, k=2, border = 2:6) # k=number of clusters, h=clusters at this height.
abline(h=13.75, col = "blue")

## to color the branches according to clusters. skipped.
# install.packages("dendextend", dependencies = TRUE)
library(dendextend)
hc.complete_dend = as.dendrogram(hc.complete)
hc.complete_colcl = color_branches(hc.complete_dend, k=2, labels = FALSE)
plot(hc.complete_colcl)
## to color the branches according to clusters. skipped.

Xsc.hc = cutree(hc.complete, k=2)
Hitters.hc = cbind(X, Hitters$Salary, Xsc.hc) # class = "matrix" "array"
colnames(Hitters.hc)[20] = "Salary"
Hitters.hc.cl1 = subset(Hitters.hc, Xsc.hc==1) # dim 233 by 21
```

```
Hitters.hc.cl2 = subset(Hitters.hc, Xsc.hc==2) # dim 30 by 21
# this method of subsetting is for dataframe, the code is modified ++
# ++ accordingly in the k-means clustering part.
# code for subsetting matrix is ++
# ++ new.matrix = matrix[ matrix[,"i"]==1, ], i is var/column name.
colMeans(Hitters.hc.cl1)
colMeans(Hitters.hc.cl2)
# Q2.(c) code ends

# Q2.(d) code
set.seed(6340)
Xsc.km = kmeans(Xsc, centers=2, nstart=20)
Hitters.km = as.data.frame(cbind(X, Hitters$Salary, Xsc.km$cluster))
colnames(Hitters.km)[20] = "Salary"
colnames(Hitters.km)[21] = "Cl.km"
Hitters.km.cl1 = subset(Hitters.km, Cl.km==1) # dim 189 by 21
Hitters.km.cl2 = subset(Hitters.km, Cl.km==2) # dim 74 by 21
colMeans(Hitters.km.cl1)
colMeans(Hitters.km.cl2)
# Q2.(d) code ends
```

Q3.

```
# Q3.(a) code
Hitters.sup = cbind(log(Hitters$Salary), Hitters[,-19])
colnames(Hitters.sup)[1] = "logSalary"
fit.3a = glm(logSalary ~ ., family = gaussian, data = Hitters.sup)

library(boot)
set.seed(6340)
cv.err.3a = cv.glm(Hitters.sup, fit.3a)$delta[1]
# default K=nrow(df), hence this is LOOCV

out1 = as.data.frame(fit.3a$coefficients)
write.csv(out1, "C:/Users/Kaustav/Downloads/STAT 6340/Projects/P5 due 19-Apr-2021/out1.csv", row.names
= T)
# Q3.(a) code ends

# Q3.(b) code
library(pls)
set.seed(6340)

fit.3b = pcr(logSalary ~ ., data = Hitters.sup, scale = TRUE, validation = "LOO")
MSEP(fit.3b)
which.min(MSEP(fit.3b)$val[1, 1,]) # 16 comps

fit.3b.2 = pcr(logSalary ~ ., data = Hitters.sup, ncomp=16, scale = TRUE, validation = "none")

out2 = as.data.frame(fit.3b.2$coefficients)
```

```
write.csv(out2, "C:/Users/Kaustav/Downloads/STAT 6340/Projects/P5 due 19-Apr-2021/out2.csv", row.names
= T)
# Q3.(b) code ends

# Q3.(c) code
set.seed(6340)

fit.3c = plsr(logSalary ~ ., data = Hitters.sup, scale = TRUE, validation = "LOO")
MSEP(fit.3c)
which.min(MSEP(fit.3c)$val[1, 1,]) # 12 comps

fit.3c.2 = plsr(logSalary ~ ., data = Hitters.sup, ncomp=12, scale = TRUE, validation = "none")

out3 = as.data.frame(fit.3c.2$coefficients)
write.csv(out3, "C:/Users/Kaustav/Downloads/STAT 6340/Projects/P5 due 19-Apr-2021/out3.csv", row.names
= T)
# Q3.(c) code ends

# Q3.(d) code

library(glmnet) ## ElasticNet data prep
y = Hitters.sup$logSalary
x = model.matrix(logSalary ~ ., Hitters.sup)[, -1]
# this reflects encoding for the qualitative var and 1 refers to the intercept, removed

set.seed(6340)
grid = (exp(1))^seq(2, -4, length = 400)
# fit.3d = glmnet(x, y, alpha = 0, lambda = grid, thresh = 1e-7) # alpha=0 for Ridge regression
cv.out.3d = cv.glmnet(x, y, alpha = 0, lambda = grid, type.measure="mse", nfolds = nrow(Hitters.sup), grouped
= FALSE) # nfolds=10 default
plot(cv.out.3d)
lambda.3d = cv.out.3d$lambda.min # [312] lambda.min = 0.06776365, cvm (cross validated mean) =
0.4082729
coef.glmnet(cv.out.3d, s= lambda.3d)
# Q3.(d) code ends

## Plot of the optimal model selection. Q3.
plot.new()
layout(matrix(c(1,2,3), 1, 3, byrow = TRUE))
validationplot(fit.3b, val.type = "MSEP", estimate = "CV", intercept = FALSE, type = "l", main="PCR model")
validationplot(fit.3c, val.type = "MSEP", estimate = "CV", intercept = FALSE, type = "l", main="PLS model")
plot(cv.out.3d, main="Ridge Regression model")
```

Q1.(b) PC Loading for Hitters data (after Standardization)

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15	PC16	PC17	PC18	PC19
AtBat	0.19829	0.383784	-0.088626	0.031967	-0.028117	0.070646	-0.107044	0.26981	-0.012183	0.145621	0.097328	-0.10315	0.03985	0.306248	-0.532433	0.510331	-0.139342	0.10679	0.053777
Hits	0.195861	0.377271	-0.074032	0.017982	0.004652	0.08224	-0.130026	0.388722	-0.061604	0.130476	0.014433	-0.121009	-0.003593	0.210616	0.023442	-0.720168	0.167363	-0.043568	-0.097782
HmRun	0.204369	0.237136	0.216186	-0.235831	-0.07766	0.149646	0.505833	-0.226278	0.127422	-0.351111	-0.20219	0.314852	0.108689	-0.001353	-0.355455	-0.200408	-0.047702	0.058377	-0.024805
Runs	0.198337	0.377721	0.017166	-0.049942	0.038536	0.13666	-0.201764	0.114518	-0.17123	0.032245	-0.312187	0.3217	0.381219	-0.266583	0.4683	0.220518	-0.140946	-0.04705	0.059145
RBI	0.235174	0.314531	0.073085	-0.138985	-0.024299	0.111675	0.31944	0.005082	0.131146	-0.172233	0.243415	-0.347752	-0.440143	-0.007486	0.461468	0.237366	0.106688	-0.063998	0.019351
Walks	0.208924	0.229606	-0.045636	-0.130615	0.032495	0.01948	-0.55842	-0.623342	-0.021438	-0.12094	0.176393	-0.185278	-0.04102	-0.23794	-0.176549	-0.102541	0.04224	-0.006655	-0.01803
Years	0.282575	-0.262402	-0.034581	0.095312	0.010361	-0.033243	0.012029	0.138314	-0.010645	-0.512507	0.191547	-0.354594	0.60501	0.08608	0.066239	0.024135	0.095586	0.08578	0.020309
CAtBat	0.330463	-0.192904	-0.083574	0.091114	-0.011716	-0.024377	-0.012057	0.14703	-0.054657	-0.101137	-0.030238	0.06239	-0.148585	-0.168155	-0.157743	0.056835	-0.182264	-0.720128	-0.409277
CHits	0.330742	-0.182899	-0.086251	0.083751	-0.008524	-0.029395	-0.02	0.194547	-0.094925	-0.07722	-0.029848	0.083475	-0.266807	-0.290311	-0.136632	-0.110442	-0.031999	0.003477	0.770272
CHmRun	0.318979	-0.126297	0.086272	-0.074278	-0.032652	0.04078	0.22883	-0.24949	0.167949	0.650534	0.07979	-0.074114	0.330107	0.039828	0.009399	0.026816	0.291752	-0.254301	0.166104
CRuns	0.338208	-0.172276	-0.052996	0.069177	0.017569	-0.006874	-0.063057	0.084597	-0.093523	0.010116	-0.151711	0.229114	-0.202301	-0.129123	-0.049945	0.162387	0.622512	0.400103	-0.344135
CRBI	0.340343	-0.168092	-0.014993	0.006674	-0.027985	-0.011476	0.118539	-0.00223	0.051372	0.281008	0.09719	-0.120661	-0.032869	-0.208779	0.069401	-0.142912	-0.609594	0.475372	-0.25994
CWalks	0.316803	-0.192315	-0.042121	0.030371	0.033969	-0.033988	-0.177639	-0.263124	-0.002584	-0.081613	-0.19055	0.215694	-0.1667	0.736944	0.24482	-0.004483	-0.169723	-0.00474	0.087714
LeagueN	-0.054471	-0.095213	-0.547725	-0.396013	-0.011992	0.136837	0.076509	-0.026194	-0.001727	0.015921	-0.580759	-0.406626	-0.009116	-0.016491	-0.042854	0.023716	0.007534	0.0017	0.005876
DivisionW	-0.025725	-0.03668	0.016201	0.042747	-0.985726	0.090885	-0.11339	0.002935	0.00347	-0.0115	-0.013549	0.000624	-0.000723	0.006168	0.049761	0.003533	0.015916	0.003545	-0.000887
PutOuts	0.077697	0.155737	-0.051274	-0.287591	-0.105866	-0.92409	0.019082	0.065108	0.094082	-0.005917	-0.035528	0.041542	0.038175	-0.011982	0.027451	0.010897	0.015599	-0.004609	-0.002309
Assists	-0.000842	0.168652	-0.397871	0.524053	0.011125	-0.035223	0.013229	-0.0756	0.706801	-0.06031	-0.087099	0.09812	0.033609	-0.085674	0.041749	-0.017497	0.01309	0.013444	0.008872
Errors	-0.007859	0.20076	-0.38285	0.421917	-0.05529	-0.148183	0.372539	-0.301207	-0.609276	0.027996	0.075912	-0.009193	0.035014	0.023559	0.014239	-0.023906	0.008841	0.000428	-0.006087
NewLeague	-0.04191	-0.077584	-0.544582	-0.417718	-0.014458	0.156965	0.022636	0.067734	0.038457	-0.023255	0.54397	0.429403	0.062339	0.029395	0.062473	-0.012527	0.005491	0.003484	-0.007555