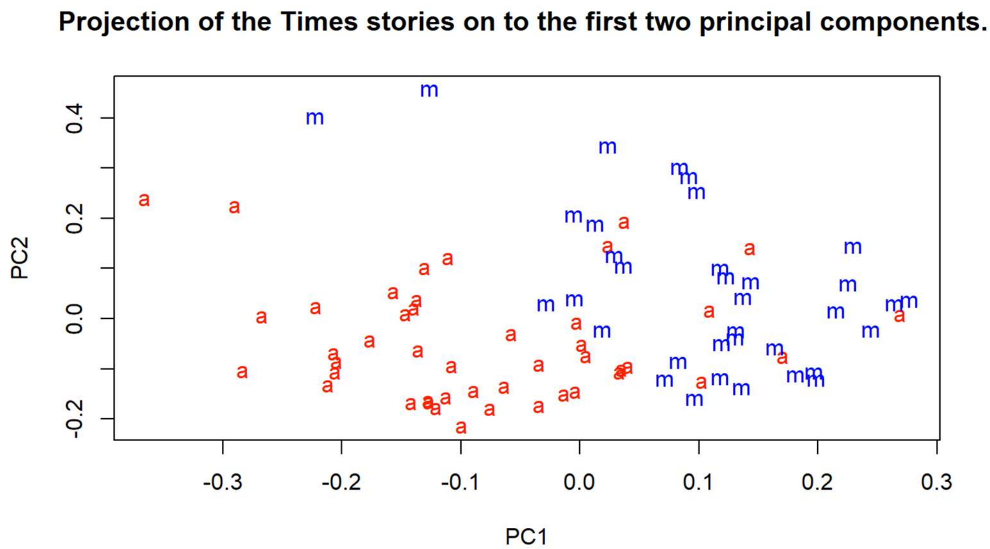
**STAT 6340 Mini Project 5**

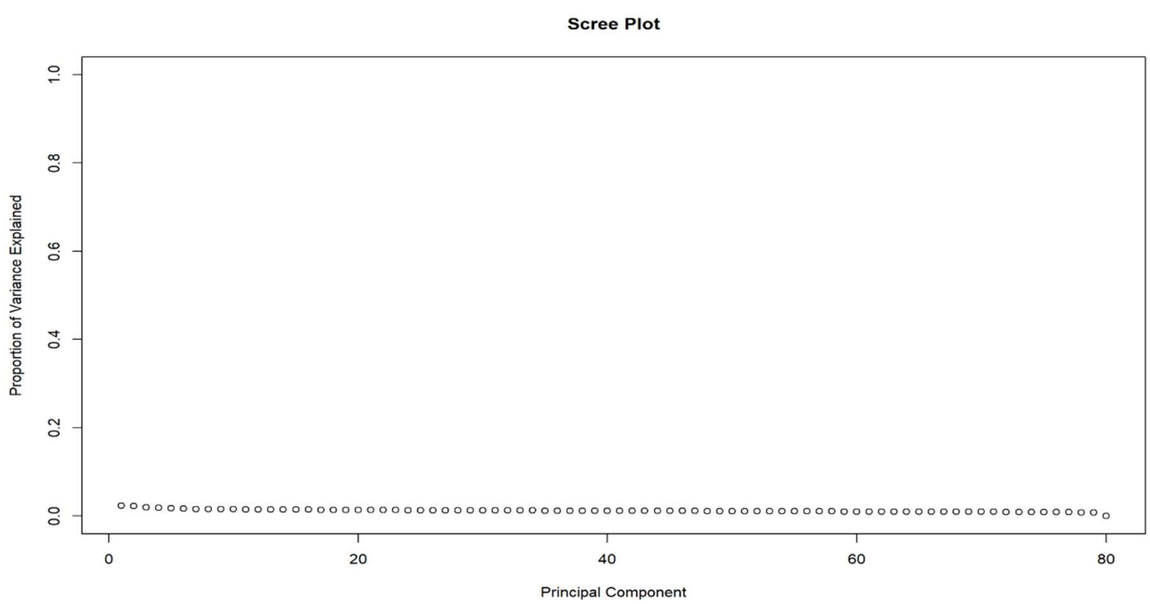
Lakshmipriya Narayanan

**SECTION 1: Observations and Answers**

**1**

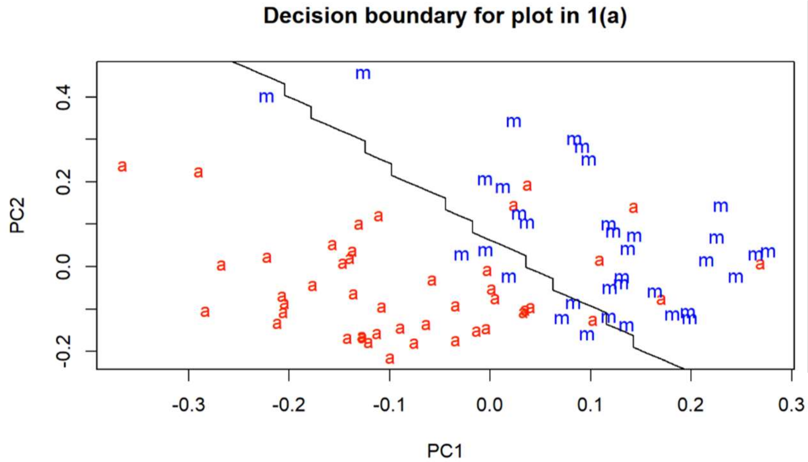
**(a):** The first set of values represent the top 30 features with the highest positive loadings on the first PC. These features with positive projections are most associated with performing music because it has words like "theater", "music", "composer", etc. The second set of values represents the top 30 features with the highest negative loading on PC1. These negative projections are most likely associated with visual arts or personal preferences of arts because of the presence of words like "paintings", "artists", "sculpture", etc. From the paper, the total number of PCs to consider was 2, and from the scree plot we can confirm the consistency because of the sharp decline observed after two principal components.




**(b):** Cumulative PVE for the first two PCs are 2.37% and 4.62% respectively implying, they explain a very small proportion of the variance. But, as stated in the "Latent Semantic Analysis" paper, and from the project plot in (a), we were able to achieve an almost-perfect classification between the art and music classes. In conclusion, since the PVE is small, we might need to consider more PCs to account for this issue (about 62 PCs because PVE = 82.76%).

**(c):**




Training error rate = 83.75%. Since the training error rate is very high we can infer that the current model is not performing well because high training error rate implies increased misclassification of a large majority of the training data. Therefore, we may have to include more PCs to fit our model or use a different model. This issue was reflected in the Cumulative PVEs in (b).

**(d):**

Test error rate = 0.3181 => In general, 31.81% of the predictions made by the logistic regression model on the test data are incorrect.

Class specific error rates computed using the confusion matrix obtained:

Class 0 (Art) = $\frac{5+2}{12} = \frac{7}{12} = 0.5833$ Class 1 (Music) = $\frac{5+2}{10} = \frac{7}{10} = 0.700$

The model incorrectly predicts 58.33% of the Art stories and 70% of the music stories. This indicates that the model has a higher error rate for music than art => this model struggles to correctly identify music stories. There is an imbalance in the overall test error rate and the class-specific error rates; the former is relatively smaller compared to the latter.

**(e):**

- Yes, we can work with M PCs in general to obtain more accurate models to fix the issues in (b)-(d) above.
- Choosing the optimal number of PCs can be done by looking at the scree plot, the cumulative PVEs or cross-validation to choose the best M.
- Both PCA and PCR involve reducing the dimensionality of the data, but the difference is in the response variable. PCA has a qualitative response variable, and we did logistic regression to achieve this. Whereas PCR uses a qualitative response variable (namesake). But in this context since our response is a categorical (art/music), PCA is more appropriate than PCR since our response variable is not continuous.

---

**2**

**(a):** Yes, because when we look at the mean and standard deviation of all the variables (in the updated dataset with dummy variables) we see that they are in vastly different scales. For instance, the mean for variable 'AtBat' is 403.64 whereas, its standard deviation is 147.30. There are many such variables like this. So, we must standardize them all.

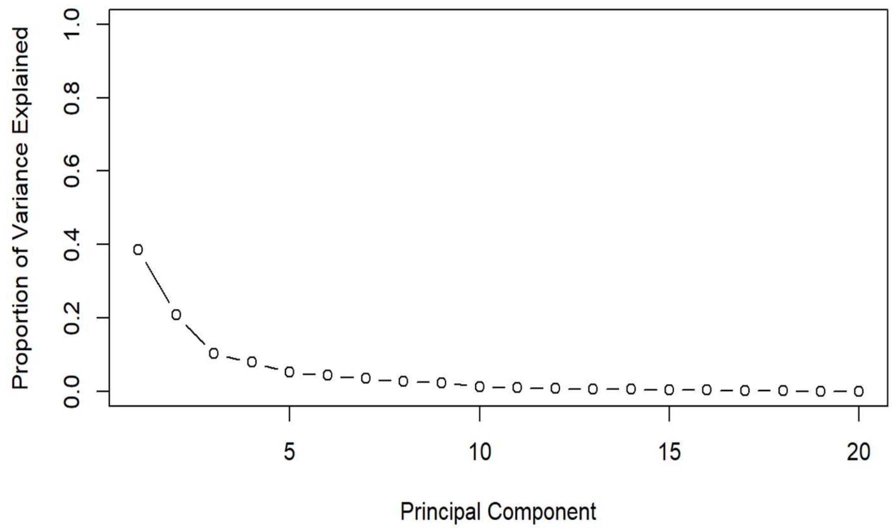
**(b):** Summary of Proportion of variance explained by each principal component and scree plot:

| PC | Proportion of Variation Explained (PVE) | Cumulative PVE |
|---|---|---|
| 1 | 0.386 | 0.386 |
| 2 | 0.207 | 0.594 |
| 3 | 0.101 | 0.695 |
| 4 | 0.078 | 0.774 |

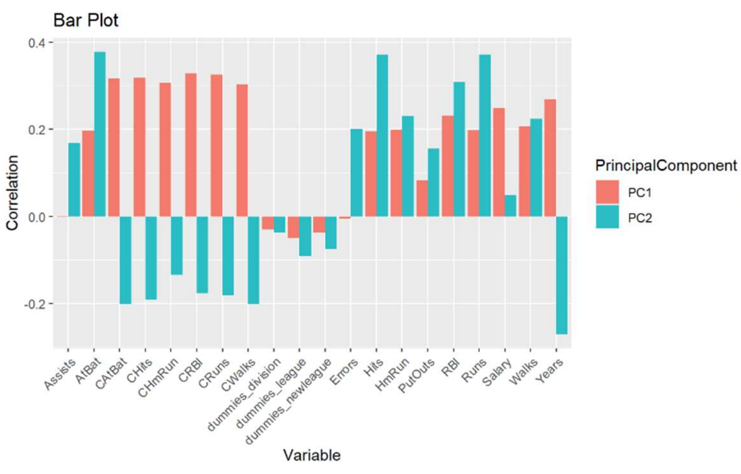
Interpretation of Scree Plot below:

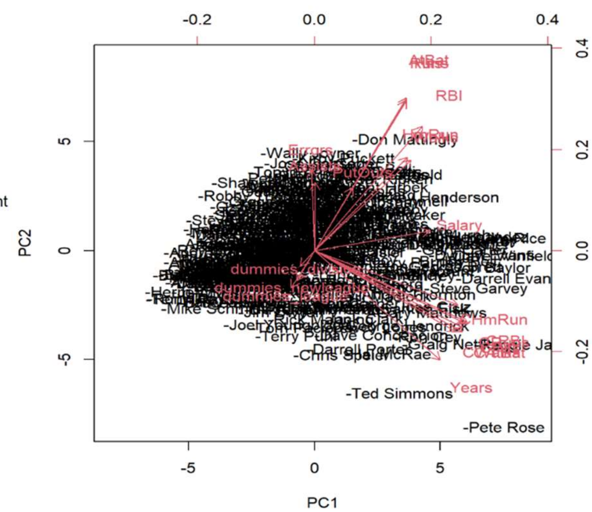
From the Scree plot above for PVE we can see that there is a sharp decline in the plot after the first 5 or 6 PCs. Hence, it is recommended to use the first 6 PCs.

| | | |
|---|---|---|
| 5 | 0.051 | 0.826 |
| 6 | 0.042 | 0.868 |
| 7 | 0.034 | 0.903 |
| 8 | 0.025 | 0.929 |
| 9 | 0.023 | 0.952 |
| 10 | 0.012 | 0.965 |
| 11 | 0.009 | 0.974 |
| 12 | 0.0068 | 0.981 |
| 13 | 0.0063 | 0.987 |
| 14 | 0.0047 | 0.992 |
| 15 | 0.0029 | 0.995 |
| 16 | 0.0025 | 0.997 |
| 17 | 0.0013 | 0.999 |
| 18 | 0.0006 | 0.9997 |
| 19 | 0.0002 | 0.9999 |
| 20 | 0.00005 | 1.0000 |




**(c):** Results of correlations of the standardized response variable using a bar plot and scores on the two components and the loadings on them using a Biplot *(For bigger and clearer images see Code in Section 2)*




From the biplot we can see that variables like Years, CAtBat, HmRun, Chits, etc. determine the first PC. The second PC is determined by variables like DivisionsN, NewLeagueN, LeagueN. These variables look like they are all performance based on the current season. Therefore, the correlation between the first PCs are more heavier than those in the second PC.

---

**3**

**(a):** Yes, because standardizing ensures that each variable is on the same scale for distance calculations in clustering. Without standardizing, variables with larger scales can dominate the distance measures and this will lead to biased clusters.

**(b):** Since our goal is to cluster players, we would rather use metric-based distances (Euclidean distance), because this would cluster players based on overall performance. Correlation-based distance is not appropriate in this context because we are not interested in clustering players based on the similarity of their performance patterns.

**(c):** Cluster-specific means of the variables:

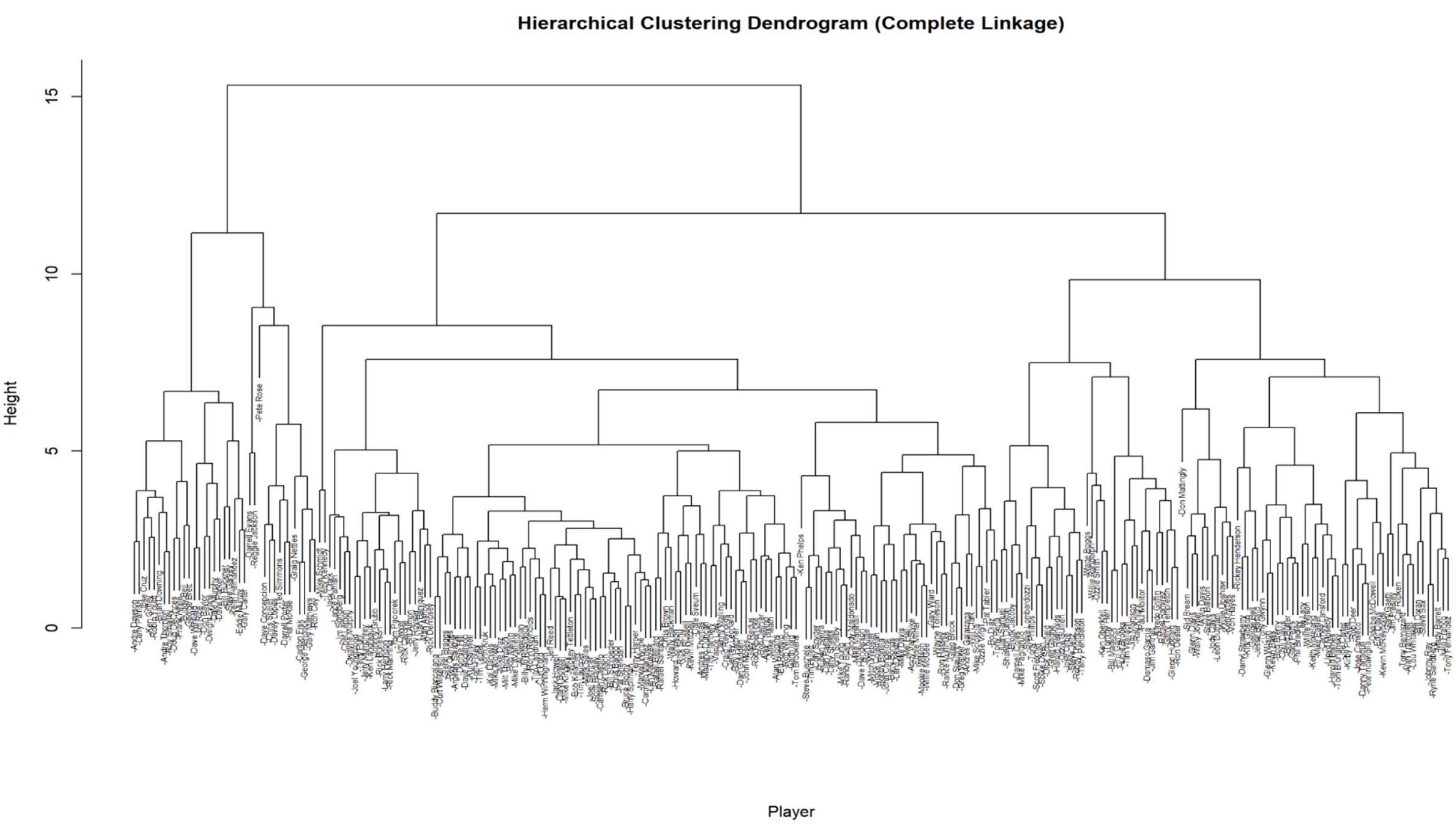
| Cluster <int> | AtBat <dbl> | Hits <dbl> | HmRun <dbl> | Runs <dbl> | RBI <dbl> | Walks <dbl> | Years <dbl> | CAtBat <dbl> | CHits <dbl> | CHmRun <dbl> | CRuns <dbl> | CRBI <dbl> | CWalks <dbl> | PutOuts <dbl> | Assists <dbl> | Errors <dbl> | Salary <dbl> |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 397.4027 | 106.1018 | 10.65929 | 53.59292 | 48.64602 | 39.08407 | 6.00000 | 1946.588 | 522.3097 | 42.76549 | 257.1947 | 221.3805 | 179.5619 | 288.1903 | 126.51327 | 9.026549 | 466.8170 |
| 2 | 441.7568 | 118.3784 | 17.48649 | 61.78378 | 68.83784 | 53.51351 | 15.32432 | 7000.135 | 1943.0541 | 230.94595 | 996.6216 | 996.4324 | 753.2162 | 306.1081 | 71.40541 | 5.945946 | 958.0504 |

rows

Mean salaries of the players in the two clusters:

| Cluster <int> | Salary <dbl> |
|---|---|
| 1 | 466.8170 |
| 2 | 958.0504 |

**Interpretation:** Cluster 1 contains 226 players and Cluster 2 contains 37 players. The larger cluster likely represents a group of players with more common characteristics based on the standardized variables and this also reflects in their salary of $466,817.These players may be less experienced or may have lower performance metrics. Whereas the smaller cluster likely represents a group of players with more unique characteristics that set them apart from the majority and the reason for their higher salary of $958,050. These players may be more experienced, have high performances, or play important roles in their respective teams.




**(d):** Cluster-specific means of the variables and mean salaries (standardized and unstandardized):

| **Variables** | **Cluster 1: 193 players** | | **Cluster 2: 70 players** | |
|---|---|---|---|---|
| | Standardized | Unstandardized | Standardized | Unstandardized |
| AtBat | 0.133 | 423.31 | -0.047 | 396.64 |
| Hits | 0.135 | 113.95 | -0.048 | 105.64 |
| HmRun | 0.280 | 14.07 | -0.099 | 10.74 |
| Runs | 0.134 | 58.18 | -0.047 | 53.52 |

| | | | | |
|---|---|---|---|---|
| RBI | 0.272 | 58.55 | -0.097 | 48.97 |
| Walks | 0.287 | 47.36 | -0.102 | 38.89 |
| Years | 1.325 | 13.66 | -0.471 | 5.05 |
| CAtBat | 1.403 | 5867.37 | -0.499 | 1515.92 |
| Chits | 1.388 | 1622.14 | -0.493 | 402.09 |
| CHmRun | 1.151 | 163.86 | -0.409 | 35.58 |
| CRuns | 1.394 | 823.0 | -0.495 | 196.97 |
| CRBI | 1.338 | 763.20 | -0.476 | 176.48 |
| CWalks | 1.302 | 604.14 | -0.463 | 137.95 |
| PutOuts | -0.045 | 277.86 | 0.016 | 295.27 |
| Assists | -0.129 | 99.92 | 0.046 | 125.45 |
| Errors | -0.168 | 7.47 | 0.060 | 8.98 |
| Salary | 0.711 | 857.02 | -0.253 | 421.71 |

From the above results we can see that cluster 1 players generally have below average values including salaries and vice versa for cluster 2. Looking at the mean salaries, players in cluster 1 generally earn more than players in cluster 2 because clearly $857,029 > $421,719. Therefore, K-means clustering clusters players where high-performance players have higher salaries and low performance players have lower salaries.

**(e):**

| Algorithm | **Hierarchical Clustering** | **K-Means Clustering** |
|---|---|---|
| **Cluster 1** | Players = 226 | Players = 193 |
| | Salary = $466,817 | Salary = $857,029 |
| **Cluster 2** | Players = 37 | Players = 70 |
| | Salary = $958,050 | Salary = $421,719 |

K-means has more balanced cluster sizes and clearly distinguishes between high and low performance and/or salary groups. On the other hand, hierarchical clustering has a very large cluster and a very small cluster which might be an indication of a potential outlier group. Here, the large cluster most likely contains a mix of players whereas the small cluster might only contain outliers or a specific subgroup of players. Hence, we should use the K-means clustering algorithm because it is straightforward to interpret and is more practical because it provides clear and distinct clusters based on performance and salaries.

--------------------------------------------------------------------------------------------------------------------------------

**4**

The table below summarizes the results of (a)-(d):

| **Part** | **Model** | **Optimal M / Penalty parameter** | **Test MSE using LOOCV** |
|---|---|---|---|
| (a) | Multiple Regression | -- | 0.4214 |
| (b) | PCR(Principal Components Regression) | M = 16 | 0.4104 |
| (c) | PLS (Partial Least Squares Regression) | M = 12 | 0.4203 |
| (d) | Ridge Regression | Penalty Parameter = 0.0533 | 0.4082 |

**(e):** Looking at the table above, we can conclude that **Ridge regression** should be preferred over the rest because of its lowest test error rate amongst the other methods. But using the other methods above would not be non-ideal because the test error rates of all these models are remarkably similar to each other.

# Section 2: Code

```
#Load Required Libraries

library(ggplot2)
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```
library(plyr)
library(dplyr)
```

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

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

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

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

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(ISLR)
library(reshape2)
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:caret':
##
##     R2
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```r
set.seed(1)
```

### Problem 1:

## (a):

```r
nyt_train <- read.csv("C:/Users/lakpr/Documents/Working Directory STAT ML 6340/nyt.train.csv")
nyt_test <- read.csv("C:/Users/lakpr/Documents/Working Directory STAT ML 6340/nyt.test.csv")

#PCA for the training data
pca_train <- prcomp(nyt_train[,-1])
nyt_latent_sem <- pca_train$rotation
#nyt_latent_sem

#FIRST PCAs
#music components
signif(sort(nyt_latent_sem[,1], decreasing = TRUE)[1:30], 2)
```

```
##      theater        music            m     theaters    composers      matinee
##        0.180        0.130        0.080        0.079        0.078        0.077
##        opera       sunday      musical       jersey            p    orchestra
##        0.075        0.067        0.065        0.064        0.064        0.062
##         band    committee  performance performances         east        organ
##        0.061        0.060        0.059        0.056        0.056        0.053
##        dance         hour      program       events    yesterday         will
##        0.052        0.051        0.051        0.050        0.049        0.049
##     recitals       ballet     purchase          X.d    guitarist        calif
##        0.048        0.048        0.048        0.047        0.045        0.044
```

```r
#art components
signif(sort(nyt_latent_sem[,1], decreasing = FALSE)[1:30], 2)
```

```
##        her         she          ms    painting   paintings      mother
##     -0.150      -0.140      -0.130      -0.110      -0.100      -0.092
##     cooper     artists       white      images           i        said
##     -0.090      -0.086      -0.078      -0.077      -0.071      -0.070
##    process   sculpture     picasso    gagosian         art          my
##     -0.070      -0.070      -0.068      -0.065      -0.064      -0.064
##     nature       image       color  sculptures        work         red
##     -0.064      -0.061      -0.061      -0.059      -0.059      -0.058
##     artist      rothko       paint photographs       paper      figure
##     -0.056      -0.055      -0.055      -0.055      -0.054      -0.054
```

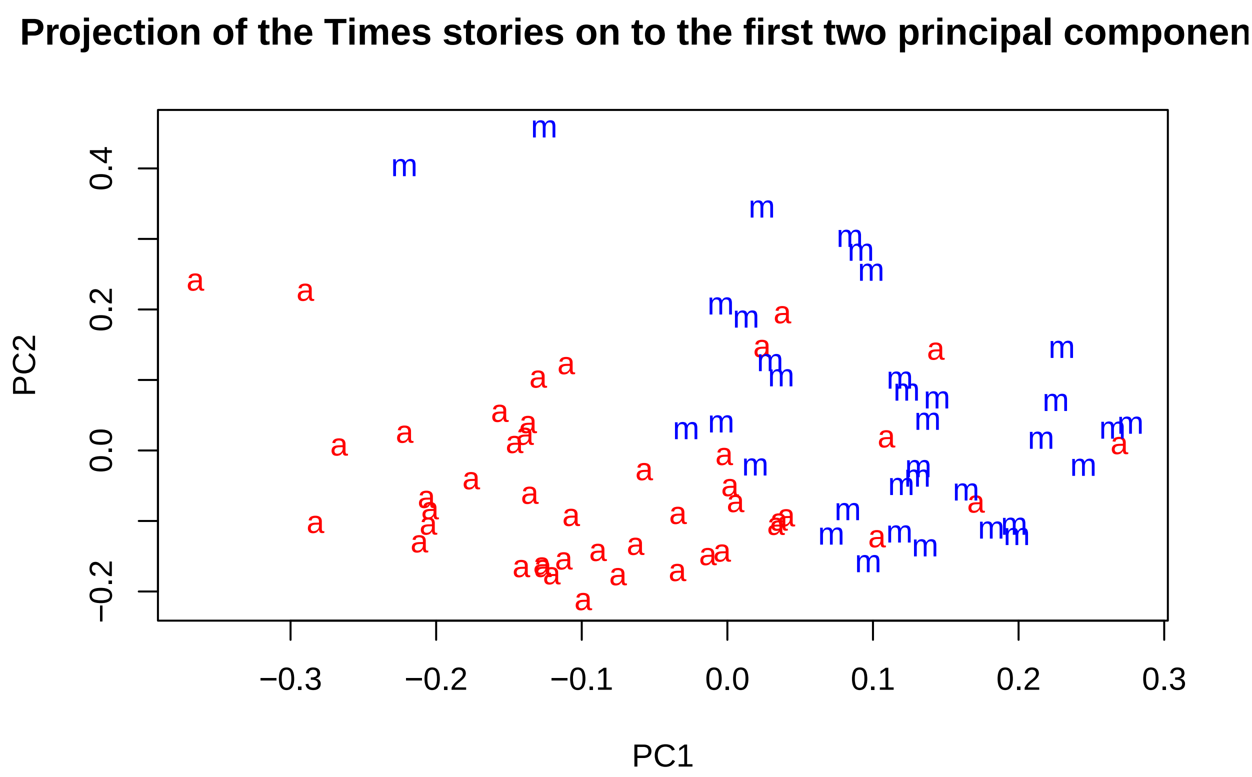
```r
#SECOND PCAs
#music components
signif(sort(nyt_latent_sem[,2], decreasing = TRUE)[1:30], 2)
```

```
##        she        her    theater       said          i         ms     mother
##      0.240      0.240      0.200      0.170      0.120      0.110      0.110
##     cooper       says      opera         my       hour         id         im
##      0.110      0.089      0.084      0.084      0.082      0.081      0.079
## production        was        mrs       play        sir   broadway     awards
##      0.075      0.075      0.074      0.074      0.071      0.070      0.066
##        you   national      garde         me     season   jonathan       week
##      0.066      0.065      0.063      0.062      0.062      0.062      0.060
##       baby   networks
##      0.059      0.059
```

```r
#art components
signif(sort(nyt_latent_sem[,2], decreasing = FALSE)[1:30], 2)
```

```
##     patterns      chinese          feb      chelsea    computers      diamond
##       -0.065       -0.051       -0.046       -0.046       -0.046       -0.045
##    europeans      gallery       museum          art        heads        white
##       -0.044       -0.042       -0.041       -0.040       -0.039       -0.039
##        stone        views      painted    recalling         soho      artists
##       -0.039       -0.039       -0.039       -0.039       -0.039       -0.038
##        pills       statue       newman     computer compositions         grid
##       -0.037       -0.037       -0.037       -0.037       -0.037       -0.037
##   landscapes      spatial       images         wood   technology     personal
##       -0.037       -0.037       -0.036       -0.035       -0.035       -0.035
```

```r
plot(pca_train$x[, 1:2],
     pch = ifelse(nyt_train[, "class.labels"] == "music", "m", "a"),
     col = ifelse(nyt_train[, "class.labels"] == "music", "blue", "red"),
     main = "Projection of the Times stories on to the first two principal components. ")
```




```r
options(scipen = 999)

# Compute the proportion of variance explained (PVE)
nyt_var <- pca_train$sdev^2
pve_nyt <- nyt_var/sum(nyt_var)
pve_nyt
```

```
##  [1] 0.02377051183951651644465918877813237486
##  [2] 0.02248964972474734591578737763484241441
##  [3] 0.01992767564370755487002107031457853736
##  [4] 0.01917766388827963144891874947006726870
##  [5] 0.01760773100580938915760143004263227340
##  [6] 0.01669014124144367616775497253911453299
##  [7] 0.01641740634216232175290883787965867668
##  [8] 0.01621192691600482557734430599793995498
##  [9] 0.01564125142345171998181641015435161535
## [10] 0.01551349812708615640011178271606695489
## [11] 0.01532650711998104729738567897356915637
## [12] 0.01516716484969267073057430650351307122
## [13] 0.01498063788849865289598550788241482223
## [14] 0.01495482409874299319085810822116400232
## [15] 0.01476828811142036346204520214087096974
## [16] 0.01467814263032670915598565386517293518
## [17] 0.01449994272336672072876151418086010381
## [18] 0.01435995505635463166671517143413439044
## [19] 0.01419065705519465848960525278243949288
```

```
## [20] 0.01405111455515966820128959113844757667
## [21] 0.01397934242840760676873657075702794828
## [22] 0.01382576539385220459887904098650324158
## [23] 0.01362624017500337764152718023069610354
## [24] 0.01347467174373183189384217683937094989
## [25] 0.01346174709696753776788646916884317761
## [26] 0.01339565588751251076771175974045036128
## [27] 0.01325615177457603426069976393364413525
## [28] 0.01320062916016997989210324249143013731
## [29] 0.01312758290058467329497471354216031614
## [30] 0.01303496786866309778418759890428191284
## [31] 0.01298823565105808666708675502832193160
## [32] 0.01294589721094153815517113770283685881
## [33] 0.01276235104684166933586286774016116397
## [34] 0.01267745993811579541254985770137864165
## [35] 0.01254309998758530893259166560937956092
## [36] 0.01245797499042419938242165500241753762
## [37] 0.01239791527344517169106907772402337287
## [38] 0.01228591998193288847840420885404455476
## [39] 0.01216227223548485905646465710105985636
## [40] 0.01214716319247320132357348398954854929
## [41] 0.01209859883687047950318049771567530115
## [42] 0.01199278181396379837353460828808238148
## [43] 0.01191640668405511208072233841903653229
## [44] 0.01179037288561738880399154538736183895
## [45] 0.01173762036382239130893889722528911079
## [46] 0.01171018885007273427345531047194526764
## [47] 0.01165047144200739154262791430483048316
## [48] 0.01158662491915890373206377006454204093
## [49] 0.01147655530732475878674669900192384375
## [50] 0.01138779434066434538552758937157705077
## [51] 0.01137891291556434279763632133608552977
## [52] 0.01132132940830031173073955130803369684
## [53] 0.01122904899494338304410945283962064423
## [54] 0.01111273440321451644319239449032465927
## [55] 0.01108575086808477258570881929244933417
## [56] 0.01086703798356661830604164009628220811
## [57] 0.01084160850521589773698405423374424572
## [58] 0.01067703449419818949317484424454960390
## [59] 0.01060539273005544833039515140171715757
## [60] 0.01051323688451219311168483017127073254
## [61] 0.01045356028807349797749015607450928655
## [62] 0.01039925575681297821584170293363058590
## [63] 0.01037490936698380258884633065008529229
## [64] 0.01024564897400079450939802683251400595
## [65] 0.01015635790155510788379533693159828545
## [66] 0.01002926511862338269931438361481923494
## [67] 0.01002768590633102557818645550469227601
## [68] 0.00991861337433307121980163856278522871
## [69] 0.00985075022969556825247483544671922573
```

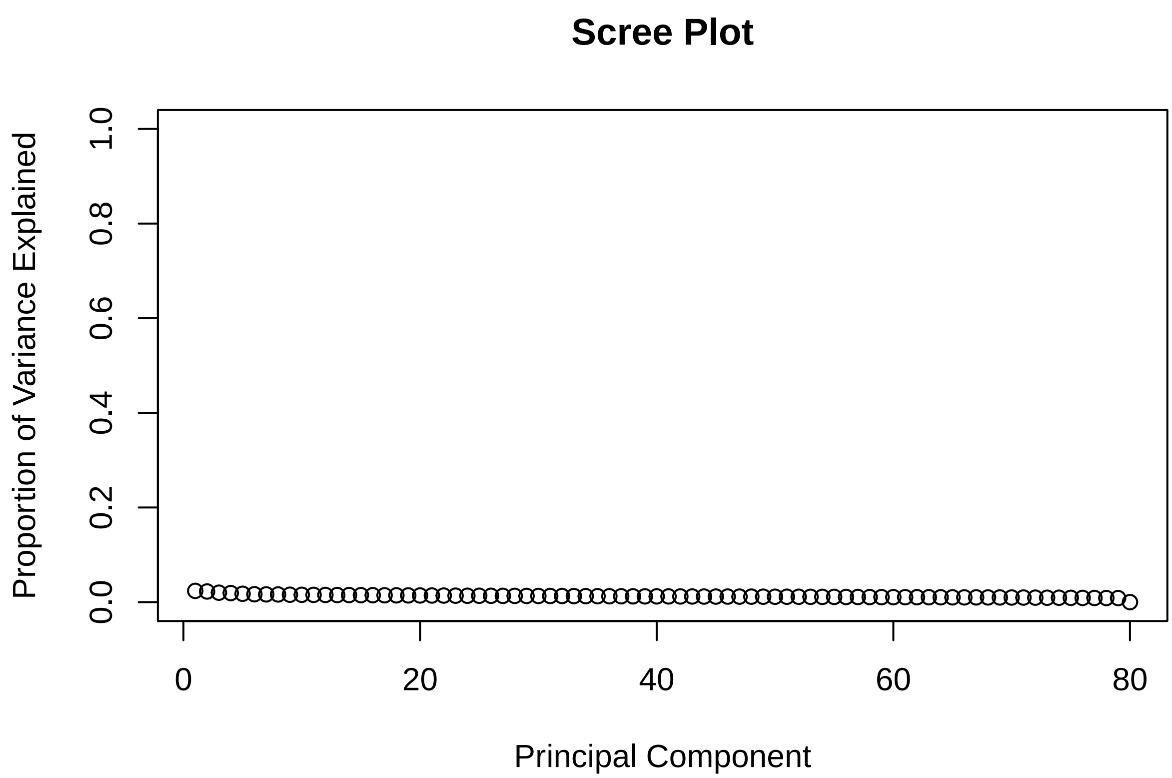
```
## [70] 0.009725216706360438959078074105946143 39
```

```
## [70] 0.00972521670636043895907807410594614339
## [71] 0.00967953960893166119594521745739257312
## [72] 0.00944698820510544469419844659796581254
## [73] 0.00929666061210158468919306784528089338
## [74] 0.00926787245068065590469785774985211901
## [75] 0.00907152462356310834712402879631554242
## [76] 0.00889110444198921816283398555924577522
## [77] 0.00875976354081213666458172184547947836
## [78] 0.00866248577795118712696620377755607478
## [79] 0.00858756030616347927619802504750623484
## [80] 0.00000000000000000000000000000004985691
```

```r
cumsum(pve_nyt)
```

```
##  [1] 0.02377051 0.04626016 0.06618784 0.08536550 0.10297323 0.11966337
##  [7] 0.13608078 0.15229271 0.16793396 0.18344746 0.19877396 0.21394113
## [13] 0.22892177 0.24387659 0.25864488 0.27332302 0.28782296 0.30218292
## [19] 0.31637358 0.33042469 0.34440403 0.35822980 0.37185604 0.38533071
## [25] 0.39879246 0.41218811 0.42544426 0.43864489 0.45177248 0.46480744
## [31] 0.47779568 0.49074158 0.50350393 0.51618139 0.52872449 0.54118246
## [37] 0.55358038 0.56586630 0.57802857 0.59017573 0.60227433 0.61426711
## [43] 0.62618352 0.63797389 0.64971151 0.66142170 0.67307218 0.68465880
## [49] 0.69613536 0.70752315 0.71890206 0.73022339 0.74145244 0.75256518
## [55] 0.76365093 0.77451796 0.78535957 0.79603661 0.80664200 0.81715524
## [61] 0.82760880 0.83800805 0.84838296 0.85862861 0.86878497 0.87881423
## [67] 0.88884192 0.89876053 0.90861128 0.91833650 0.92801604 0.93746303
## [73] 0.94675969 0.95602756 0.96509909 0.97399019 0.98274995 0.99141244
## [79] 1.00000000 1.00000000
```

```r
#SCREE PLOT for PVE
plot(pve_nyt, xlab = "Principal Component", ylab = "Proportion of Variance Explained",
     ylim = c(0,1), type = 'b', main = "Scree Plot")
```




### (c):

```
#extract scores of first two PCs
pca_train_score12 <- pca_train$x[, 1:2]
#pca_train_score12

# Put the scores of the first two PCs into a data frame to use as predictor
#variables for logistic regression
pca_train_12 <- data.frame(PC1 = pca_train_score12[,1], PC2 = pca_train_score12[,2],
            class = as.factor(ifelse(nyt_train$class.labels == "music", 1, 0)))

# Fit the logistic regression model on the first two PCs as the predictors and
#the class of a story(art or music) as the response.
lg_model <- glm(class ~ PC1 + PC2, data = pca_train_12, family = binomial)
summary(lg_model)
```

```
##
## Call:
## glm(formula = class ~ PC1 + PC2, family = binomial, data = pca_train_12)
##
## Coefficients:
##             Estimate Std. Error z value   Pr(>|z|)
## (Intercept)  -0.5736     0.3519  -1.630    0.10307
## PC1          14.4540     3.2589   4.435 0.00000919 ***
```

```
## PC2           8.6093     2.5624   3.360    0.00078 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 109.650  on 79  degrees of freedom
## Residual deviance:  58.331  on 77  degrees of freedom
## AIC: 64.331
##
## Number of Fisher Scoring iterations: 5
```

```r
# Mke predictions on the training data
pca_train_pred <- predict(lg_model, type = "response")
train_pred_class <- ifelse(pca_train_pred > 0.5, 1, 0)

# Evaluate the model
table(Predicted = train_pred_class, Actual = pca_train_12$class)
```

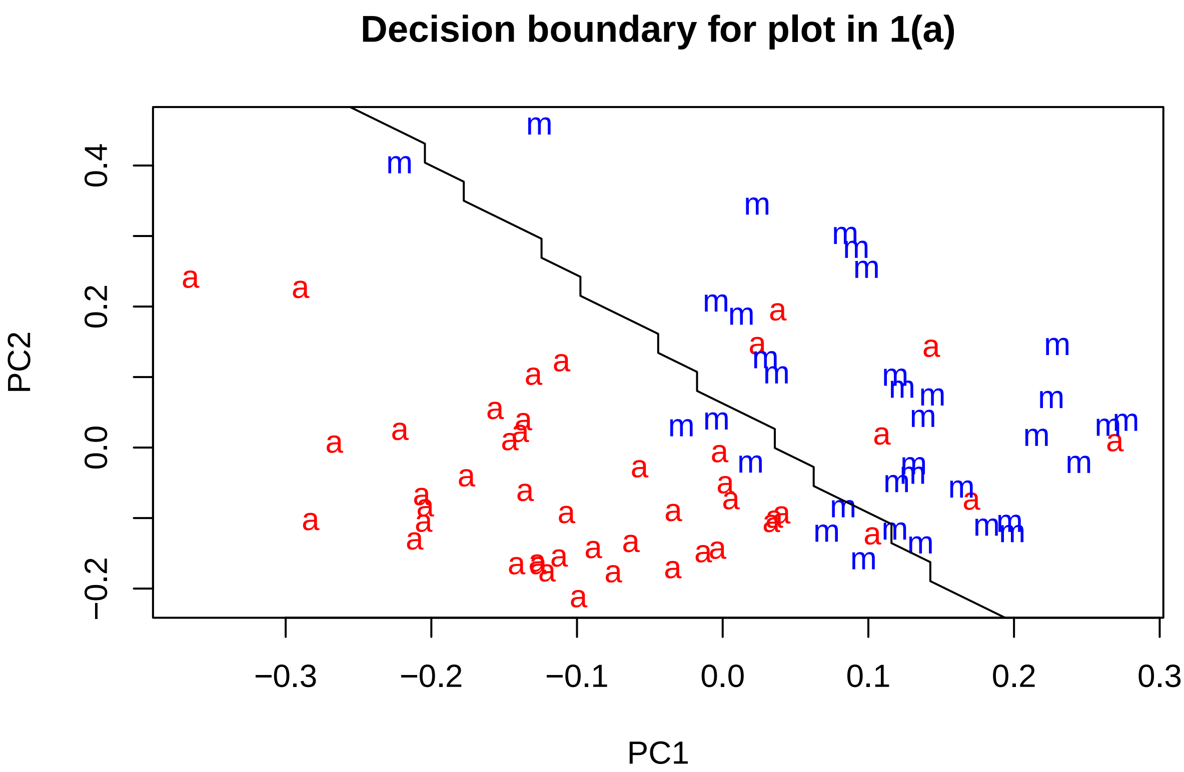
```
##          Actual
## Predicted  0  1
##         0 39  7
##         1  6 28
```

```r
accuracy <- mean(train_pred_class == pca_train_12$class)
print(paste("Training Error rate:", accuracy))
```

```
## [1] "Training Error rate: 0.8375"
```

```r
# Plot the PCA scores with the decision boundary same as done in (a) above
plot(pca_train_score12[,1], pca_train_score12[,2],
     col = ifelse(pca_train_12$class == 1, "blue", "red"),
     pch = ifelse(pca_train_12$class == 1, "m", "a"), xlab = "PC1", ylab = "PC2",
     main = "Decision boundary for plot in 1(a)")

# Create a grid of values to add the decisin boundary as a contour plot using
#the "contour" function. Set the range of the axes of the grid as minimum and
#maximum values of the first two PCs
x_min <- min(pca_train_score12[,1]) - 1
x_max <- max(pca_train_score12[,1]) + 1
y_min <- min(pca_train_score12[,2]) - 1
y_max <- max(pca_train_score12[,2]) + 1

# Build the grid
grid <- expand.grid(PC1 = seq(x_min, x_max, length.out = 100),
                    PC2 = seq(y_min, y_max, length.out = 100))

# Make predictions on the grid using the logistic regression model to predict
#the probability of each data point on the grid belonging to either of the
#classes. Here, music is class 1 and art is class 0.
```

```r
grid_pred <- predict(lg_model, newdata = grid, type = "response")
grid_pred_class <- ifelse(grid_pred > 0.5, 1, 0)

# Add the decision boundary to the above plot
contour(seq(x_min, x_max, length.out = 100), seq(y_min, y_max, length.out = 100),
        matrix(grid_pred_class, 100, 100), add = TRUE, drawlabels = FALSE,
        levels = 0.5, col = "black")
```




## (d):

```r
# Perform predictions on the test data using the same PCA model fitted on the
#training data above
pca_test_scores <- predict(pca_train, newdata = nyt_test[,-1])
pca_test_scores <- pca_test_scores[, 1:2]  # Extract the scores for the first two PCs

# Prepare the test data for prediction just like (c) but use test data and
#corresponding scores
pca_test_data <- data.frame(PC1 = pca_test_scores[,1], PC2 = pca_test_scores[,2],
            class = as.factor(ifelse(nyt_test$class.labels == "music", 1, 0)))

# Make predictions on the test data using the same logistic regression model in (c)
pca_test_pred <- predict(lg_model, newdata = pca_test_data, type = "response")
test_pred_class <- ifelse(pca_test_pred > 0.5, 1, 0)
```

```r
# Compute the test error rate = 1 - accuracy
pca_test_acc <- mean(test_pred_class == pca_test_data$class)
pca_test_err <- 1 - pca_test_acc
print(paste("Test Error Rate:", pca_test_err))
```

```
## [1] "Test Error Rate: 0.318181818181818"
```

```r
# Compute class-specific error rates using the 'caret' library and
#obtain the confusion matrix
pca_cm <- confusionMatrix(as.factor(test_pred_class), pca_test_data$class)
pca_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 10  5
##          1  2  5
##
##                Accuracy : 0.6818
##                  95% CI : (0.4513, 0.8614)
##     No Information Rate : 0.5455
##     P-Value [Acc > NIR] : 0.1419
##
##                   Kappa : 0.3419
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.8333
##             Specificity : 0.5000
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.7143
##              Prevalence : 0.5455
##          Detection Rate : 0.4545
##    Detection Prevalence : 0.6818
##       Balanced Accuracy : 0.6667
##
##        'Positive' Class : 0
##
```

## Problem 2:

```r
Hitters <- na.omit(Hitters)
players <- row.names(Hitters) #names of beaseball players
str(Hitters)
```

```
## 'data.frame':    263 obs. of  20 variables:
##  $ AtBat    : int  315 479 496 321 594 185 298 323 401 574 ...
```

```
##  $ Hits     : int  81 130 141 87 169 37 73 81 92 159 ...
##  $ HmRun    : int  7 18 20 10 4 1 0 6 17 21 ...
##  $ Runs     : int  24 66 65 39 74 23 24 26 49 107 ...
##  $ RBI      : int  38 72 78 42 51 8 24 32 66 75 ...
##  $ Walks    : int  39 76 37 30 35 21 7 8 65 59 ...
##  $ Years    : int  14 3 11 2 11 2 3 2 13 10 ...
##  $ CAtBat   : int  3449 1624 5628 396 4408 214 509 341 5206 4631 ...
##  $ CHits    : int  835 457 1575 101 1133 42 108 86 1332 1300 ...
##  $ CHmRun   : int  69 63 225 12 19 1 0 6 253 90 ...
##  $ CRuns    : int  321 224 828 48 501 30 41 32 784 702 ...
##  $ CRBI     : int  414 266 838 46 336 9 37 34 890 504 ...
##  $ CWalks   : int  375 263 354 33 194 24 12 8 866 488 ...
##  $ League   : Factor w/ 2 levels "A","N": 2 1 2 2 1 2 1 2 1 1 ...
##  $ Division : Factor w/ 2 levels "E","W": 2 2 1 1 2 1 2 2 1 1 ...
##  $ PutOuts  : int  632 880 200 805 282 76 121 143 0 238 ...
##  $ Assists  : int  43 82 11 40 421 127 283 290 0 445 ...
##  $ Errors   : int  10 14 3 4 25 7 9 19 0 22 ...
##  $ Salary   : num  475 480 500 91.5 750 ...
##  $ NewLeague: Factor w/ 2 levels "A","N": 2 1 2 2 1 1 1 2 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:59] 1 16 19 23 31 33 37 39 40 42 ...
##   ..- attr(*, "names")= chr [1:59] "-Andy Allanson" "-Billy Beane" "-Bruce Bochte" "-Bob Boone" ...
```

```r
#-------------------Dummy representation-----------------------------------------

#Creating dummy variables for categorical variables: League, Division, and
# NewLeague, dropping the reference category to avoid multicollinearity issue
dummies_league <- model.matrix(~ League, data = Hitters)[, -1]
dummies_division <- model.matrix(~ Division, data = Hitters)[, -1]
dummies_newleague <- model.matrix(~ NewLeague, data = Hitters)[, -1]

#Combine the dummy variables with the original Hitters dataset excluding the
# original categorical variables and use this for all analysis further.

Hitters_dummies <- cbind(Hitters[, !names(Hitters) %in%
                  c("League", "Division", "NewLeague")],dummies_league,
                  dummies_division, dummies_newleague)

# Inspect the new dataset #league and newleague N1 A0 division W1 E0
str(Hitters_dummies)
```

```
## 'data.frame':    263 obs. of  20 variables:
##  $ AtBat            : int  315 479 496 321 594 185 298 323 401 574 ...
##  $ Hits             : int  81 130 141 87 169 37 73 81 92 159 ...
##  $ HmRun            : int  7 18 20 10 4 1 0 6 17 21 ...
##  $ Runs             : int  24 66 65 39 74 23 24 26 49 107 ...
##  $ RBI              : int  38 72 78 42 51 8 24 32 66 75 ...
##  $ Walks            : int  39 76 37 30 35 21 7 8 65 59 ...
##  $ Years            : int  14 3 11 2 11 2 3 2 13 10 ...
##  $ CAtBat           : int  3449 1624 5628 396 4408 214 509 341 5206 4631 ...
##  $ CHits            : int  835 457 1575 101 1133 42 108 86 1332 1300 ...
```

```
##  $ CHmRun           : int  69 63 225 12 19 1 0 6 253 90 ...
##  $ CRuns            : int  321 224 828 48 501 30 41 32 784 702 ...
##  $ CRBI             : int  414 266 838 46 336 9 37 34 890 504 ...
##  $ CWalks           : int  375 263 354 33 194 24 12 8 866 488 ...
##  $ PutOuts          : int  632 880 200 805 282 76 121 143 0 238 ...
##  $ Assists          : int  43 82 11 40 421 127 283 290 0 445 ...
##  $ Errors           : int  10 14 3 4 25 7 9 19 0 22 ...
##  $ Salary           : num  475 480 500 91.5 750 ...
##  $ dummies_league   : num  1 0 1 1 0 1 0 1 0 0 ...
##  $ dummies_division : num  1 1 0 0 1 0 1 1 0 0 ...
##  $ dummies_newleague: num  1 0 1 1 0 0 0 1 0 0 ...
```

## (a):

```r
# Get mean and sd to check for scales of variables
apply(Hitters_dummies, 2, mean)
```

```
##            AtBat              Hits             HmRun              Runs
##      403.6425856       107.8288973        11.6197719        54.7452471
##              RBI             Walks             Years            CAtBat
##       51.4866920        41.1140684         7.3117871      2657.5437262
##            CHits            CHmRun             CRuns              CRBI
##      722.1863118        69.2395437       361.2205323       330.4182510
##           CWalks           PutOuts           Assists            Errors
##      260.2661597       290.7110266       118.7604563         8.5931559
##           Salary    dummies_league  dummies_division dummies_newleague
##      535.9258821         0.4714829         0.5095057         0.4638783
```

```r
apply(Hitters_dummies, 2, sd)
```

```
##            AtBat              Hits             HmRun              Runs
##      147.3072088        45.1253259         8.7571077        25.5398156
##              RBI             Walks             Years            CAtBat
##       25.8827143        21.7180561         4.7936159      2286.5829295
##            CHits            CHmRun             CRuns              CRBI
##      648.1996437        82.1975815       331.1985706       323.3676682
##           CWalks           PutOuts           Assists            Errors
##      264.0558680       279.9345755       145.0805766         6.6065742
##           Salary    dummies_league  dummies_division dummies_newleague
##      451.1186807         0.5001378         0.5008628         0.4996443
```

```r
# Scales vary so standardization required.
```

## (b):

```r
#PCA
pca_hit <- prcomp(Hitters_dummies, center = T, scale = T)
pca_hit$rotation
```

```
##                            PC1         PC2         PC3         PC4
## AtBat              0.196678381  0.37766222 -0.07950035  0.04753091
## Hits               0.195849755  0.37181804 -0.06717444  0.02880365
## HmRun              0.199198197  0.23043572  0.22071502 -0.22175643
## Runs               0.197453222  0.37187245  0.02332956 -0.03934903
## RBI                0.231499181  0.30748571  0.07933006 -0.12446295
## Walks              0.207011159  0.22430997 -0.04251274 -0.12118727
## Years              0.268911448 -0.26999928 -0.02714668  0.10463925
## CAtBat             0.317265037 -0.20148582 -0.07529413  0.10259542
## CHits              0.318348055 -0.19124937 -0.07873693  0.09386601
## CHmRun             0.307132296 -0.13472065  0.09144641 -0.06470546
## CRuns              0.325668794 -0.18087877 -0.04569223  0.07853630
## CRBI               0.327764402 -0.17678264 -0.00847891  0.01695478
## CWalks             0.303668081 -0.20065558 -0.03507286  0.04104438
## PutOuts            0.082484758  0.15557561 -0.05890109 -0.29346349
## Assists            0.001291828  0.16837290 -0.38656482  0.53039738
## Errors            -0.005848194  0.20034975 -0.37181458  0.43277231
## Salary             0.248582079  0.04867015 -0.07016748 -0.11742620
## dummies_league    -0.049877967 -0.09148431 -0.55447341 -0.37752787
## dummies_division  -0.030663000 -0.03813223  0.02511029  0.06791102
## dummies_newleague -0.037900419 -0.07437035 -0.55073305 -0.39715798
##                             PC5          PC6          PC7           PC8
## AtBat             -0.0579118154 -0.077197421 -0.104071513 -0.2258050429
## Hits              -0.0159800857 -0.080055909 -0.135082357 -0.3597548587
## HmRun             -0.1406550838 -0.180665031  0.495312789  0.1948009204
## Runs               0.0018271225 -0.144513914 -0.202773908 -0.0864349514
## RBI               -0.0751942856 -0.140897154  0.313571360 -0.0128054163
## Walks              0.0119070167 -0.024596694 -0.534453135  0.6461163155
## Years             -0.0042779303  0.008865588  0.026274755 -0.0911604482
## CAtBat            -0.0242340087  0.010196863 -0.002765204 -0.1149149431
## CHits             -0.0162918232  0.019606139 -0.013209899 -0.1677676181
## CHmRun            -0.0603084814 -0.057993556  0.226556523  0.2171395808
## CRuns              0.0051076618 -0.006535707 -0.055283201 -0.0613025989
## CRBI              -0.0432748343 -0.001740281  0.121417970 -0.0002413993
## CWalks             0.0142493730  0.006196948 -0.154101621  0.3013749056
## PutOuts            0.0006044522  0.874417435  0.112590530  0.0381021731
## Assists            0.0370429454  0.045418108  0.010720176  0.0561434154
## Errors            -0.0356237656  0.131104898  0.383018721  0.2805318868
## Salary             0.2518480006  0.234338400 -0.114147582 -0.2681879712
## dummies_league    -0.0645754330 -0.152468355  0.069547009  0.0186030860
## dummies_division  -0.9425445551  0.131176232 -0.170548944 -0.0803595365
## dummies_newleague -0.0746224053 -0.174386546  0.017555208 -0.0557960009
##                            PC9         PC10        PC11         PC12
## AtBat              0.181089850 -0.004082554  0.180638892  0.085563873
## Hits               0.143635874 -0.064054268  0.128193633  0.013836247
```

```
## HmRun            -0.098649257  0.137981701 -0.346185943 -0.200343570
## Runs              0.101565453 -0.168245750  0.032671188 -0.319358084
## RBI              -0.004908478  0.136467313 -0.170492558  0.254633228
## Walks            -0.052527652 -0.015205601 -0.118949260  0.183017186
## Years             0.178990134  0.015237584 -0.485741212  0.199779552
## CAtBat            0.123001436 -0.046791734 -0.095228309 -0.029270985
## CHits             0.109829483 -0.091910643 -0.085975495 -0.025054017
## CHmRun           -0.140102470  0.147894575  0.661003127  0.062717070
## CRuns             0.079711572 -0.092423595  0.004662174 -0.152829194
## CRBI             -0.011305131  0.042113125  0.278794601  0.094420057
## CWalks            0.073487683  0.011567923 -0.051191231 -0.200878325
## PutOuts           0.287556236  0.106819332  0.020335023 -0.041212159
## Assists          -0.080095578  0.705032032 -0.046411871 -0.088454931
## Errors           -0.091959292 -0.610502114  0.009974338  0.077907965
## Salary           -0.827432446 -0.056547220 -0.115718520  0.024240781
## dummies_league   -0.023682176 -0.005133581 -0.001971908 -0.574874165
## dummies_division -0.217123184 -0.007124841 -0.032289752 -0.009172775
## dummies_newleague  0.055644784  0.044952387  0.002071878  0.536030451
##                          PC13         PC14        PC15         PC16
## AtBat            -0.110463894 -0.013343862  0.20236215  0.543068901
## Hits             -0.121941884  0.006207128  0.25180349  0.016467880
## HmRun             0.314965275 -0.114364878 -0.01438779  0.362200840
## Runs              0.315188875 -0.387348858 -0.22495721 -0.498625455
## RBI              -0.339847063  0.438371445  0.01374409 -0.469402215
## Walks            -0.180120261  0.032618789 -0.25052305  0.152834228
## Years            -0.352056160 -0.606670511  0.11959143 -0.049002813
## CAtBat            0.063226271  0.146977934 -0.20368884  0.132031304
## CHits             0.087430105  0.256217388 -0.30637365  0.105334892
## CHmRun           -0.083543697 -0.316962141  0.03234810 -0.008100151
## CRuns             0.228793705  0.198176752 -0.14030418  0.034092922
## CRBI             -0.121148293  0.031056276 -0.20967487 -0.093415209
## CWalks            0.207667963  0.194081260  0.74167216 -0.171784261
## PutOuts           0.037798334 -0.032213916 -0.02230247 -0.033017510
## Assists           0.096916652 -0.036107046 -0.07995292 -0.050106893
## Errors           -0.007025259 -0.037727081  0.03361010 -0.008772512
## Salary            0.015302699 -0.033484036  0.07659545  0.025354029
## dummies_league   -0.415154689  0.016649431 -0.03224059  0.037221557
## dummies_division  0.003145716 -0.004867908  0.02347587 -0.044852773
## dummies_newleague  0.436186778 -0.067635411  0.04444432 -0.056395335
##                          PC17         PC18          PC19          PC20
## AtBat             0.5260358241  0.171791349  0.0938602632 -0.0540856600
## Hits             -0.7021869211 -0.200397690 -0.0302283063  0.0981518417
## HmRun            -0.1877151457  0.043652152  0.0585697921  0.0248569345
## Runs              0.1863900452  0.142176928 -0.0489535507 -0.0592218478
## RBI               0.2249622282 -0.101548193 -0.0635379740 -0.0193906972
## Walks            -0.1146527522 -0.052376342 -0.0018835917  0.0181783041
## Years             0.0460230768 -0.087005414  0.0841469312 -0.0203068123
## CAtBat            0.0240938073  0.168020690 -0.7218471414  0.4086645317
## CHits            -0.1335085027  0.018783992  0.0089468300 -0.7701032153
## CHmRun            0.0312545429 -0.297265518 -0.2482511882 -0.1661448627
```

```
## CRuns              0.1855616328 -0.611813229  0.4095815204  0.3447254454
## CRBI              -0.1878692276  0.601841938  0.4713954213  0.2602776563
## CWalks             0.0227339086  0.180288399 -0.0122037513 -0.0879499151
## PutOuts            0.0005307823 -0.019811307 -0.0020717207  0.0023944715
## Assists           -0.0237853428 -0.016110820  0.0150581784 -0.0088085731
## Errors            -0.0177394048 -0.007717020 -0.0001916008  0.0060626102
## Salary             0.0575321212  0.025447710 -0.0136444946 -0.0004903488
## dummies_league     0.0181966431 -0.008726301  0.0026485190 -0.0058405278
## dummies_division   0.0125178020 -0.012008017  0.0017842272  0.0008248913
## dummies_newleague -0.0080677988 -0.004504842  0.0030436126  0.0075417681
```

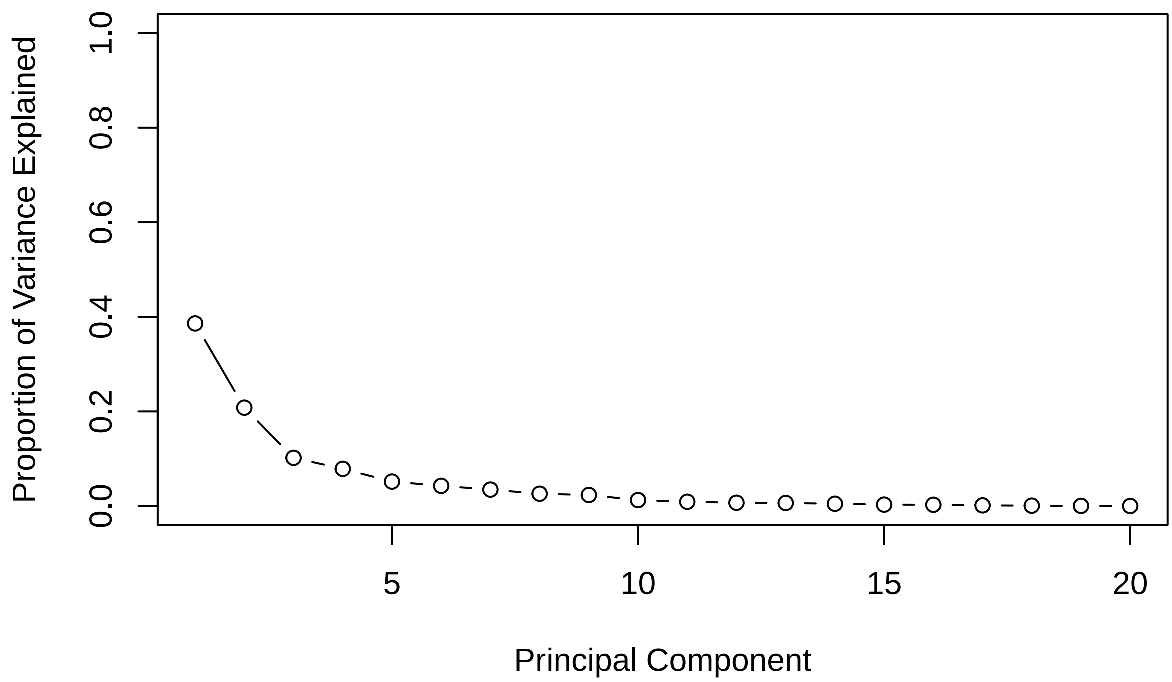
```r
# Set the scientific penalty option to a high value to get pve's in decimal
#notation rather than scientific for convenience in interpretation
options(scipen = 999)

# Compute the proportion of variance explained (PVE)
pc_var <- pca_hit$sdev^2
pve_hit <- pc_var/sum(pc_var)
pve_hit
```

```
##  [1] 0.38605622062 0.20797550557 0.10190591783 0.07858612006 0.05164621136
##  [6] 0.04268820510 0.03468525898 0.02590627443 0.02327670467 0.01248996704
## [11] 0.00900191100 0.00685110822 0.00636943838 0.00475354855 0.00290639903
## [16] 0.00258787853 0.00132420774 0.00069078264 0.00023898089 0.00005935935
```

```r
cumsum(pve_hit)
```

```
##  [1] 0.3860562 0.5940317 0.6959376 0.7745238 0.8261700 0.8688582 0.9035434
##  [8] 0.9294497 0.9527264 0.9652164 0.9742183 0.9810694 0.9874388 0.9921924
## [15] 0.9950988 0.9976867 0.9990109 0.9997017 0.9999406 1.0000000
```

```r
#SCREE PLOT for PVE
plot(pve_hit, xlab = "Principal Component", ylab = "Proportion of Variance Explained",
     ylim = c(0,1), type = 'b')
```




### (c):

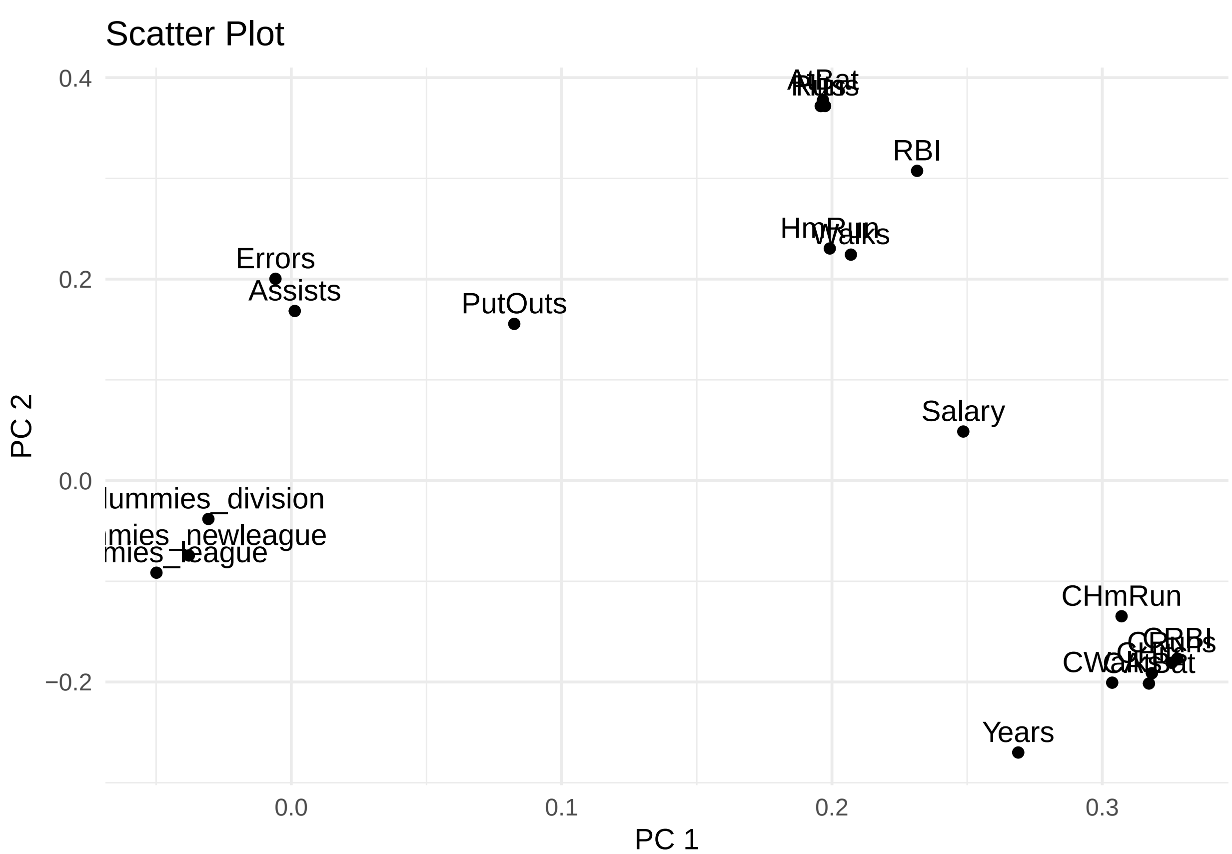
```
# Loadings matrix gives the correlations of the original variables with the
#first two PCs
loadings_hitters <- pca_hit$rotation

# Correlation of the standardized variables with first two PCs are obtained from
#the values in the rotation matrix
corr_pc1 <- loadings_hitters[, 1]
corr_pc1
```

```
##           AtBat            Hits           HmRun            Runs
##     0.196678381     0.195849755     0.199198197     0.197453222
##             RBI           Walks           Years          CAtBat
##     0.231499181     0.207011159     0.268911448     0.317265037
##           CHits          CHmRun           CRuns            CRBI
##     0.318348055     0.307132296     0.325668794     0.327764402
##          CWalks         PutOuts         Assists          Errors
##     0.303668081     0.082484758     0.001291828    -0.005848194
##          Salary  dummies_league dummies_division dummies_newleague
##     0.248582079    -0.049877967    -0.030663000    -0.037900419
```

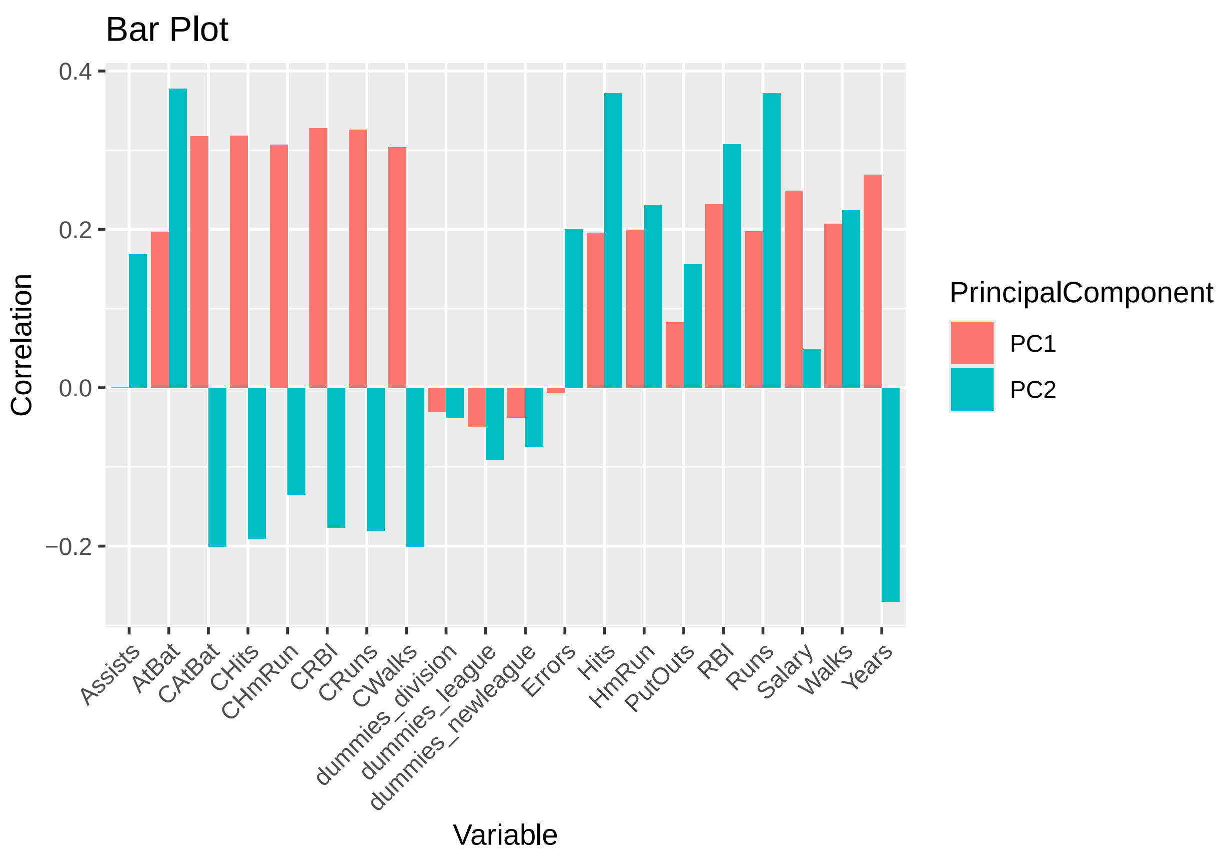
```
corr_pc2 <- loadings_hitters[, 2]
corr_pc2
```

```
##            AtBat             Hits            HmRun             Runs
##       0.37766222       0.37181804       0.23043572       0.37187245
##              RBI            Walks            Years           CAtBat
##       0.30748571       0.22430997      -0.26999928      -0.20148582
##            CHits           CHmRun            CRuns             CRBI
##      -0.19124937      -0.13472065      -0.18087877      -0.17678264
##           CWalks          PutOuts          Assists           Errors
##      -0.20065558       0.15557561       0.16837290       0.20034975
##           Salary    dummies_league dummies_division dummies_newleague
##       0.04867015      -0.09148431      -0.03813223      -0.07437035
```

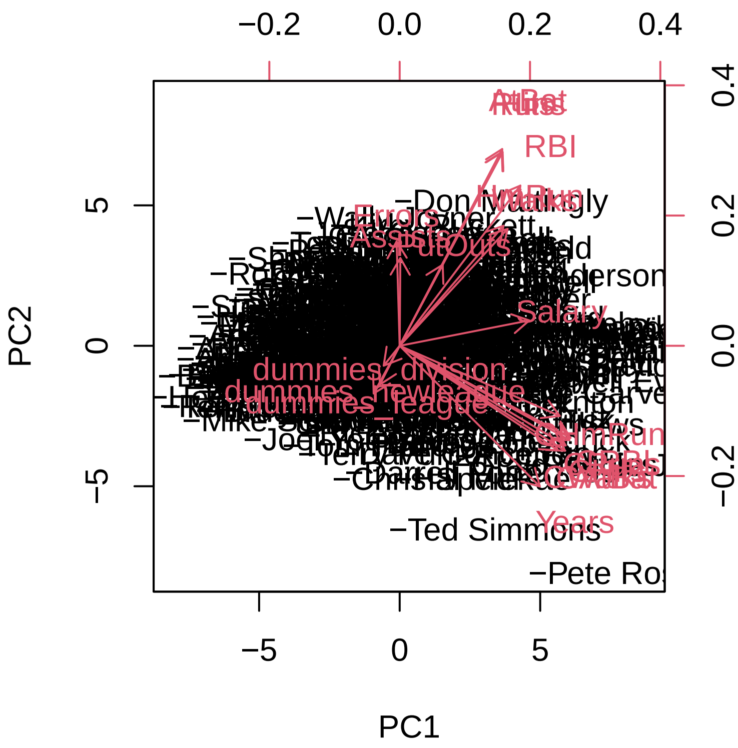
```r
# Creating a scatter plot with the variables and the correlations of the first
#two PCs above helps visualize the above correlations

# Create a data frame for making the scatter plot with the PCs
corr_df <- data.frame(Variable = rownames(loadings_hitters),
                              PC1 = corr_pc1, PC2 = corr_pc2)

# Plot the correlations as a scatter plot
ggplot(corr_df, aes(x = PC1, y = PC2, label = Variable)) +
  geom_point() +
  geom_text(vjust = -0.5, hjust = 0.5) +
  labs(title = "Scatter Plot",
       x = "PC 1",
       y = "PC 2") +
  theme_minimal()
```



```r
#Scatter plot is not easy to interpret so use a barplot.

#Melt data frame to reshape into long format for plotting a bar plot so that the bars are grouped by the PCs.
corr_melted <- melt(corr_df, id.vars = "Variable",
                    variable.name = "PrincipalComponent",
                    value.name = "Correlation")

ggplot(corr_melted, aes(x = Variable, y = Correlation, fill = PrincipalComponent)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Bar Plot",
       x = "Variable",
       y = "Correlation") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Create the biplot
biplot(pca_hit, scale = 0)
```




**Problem 3:**

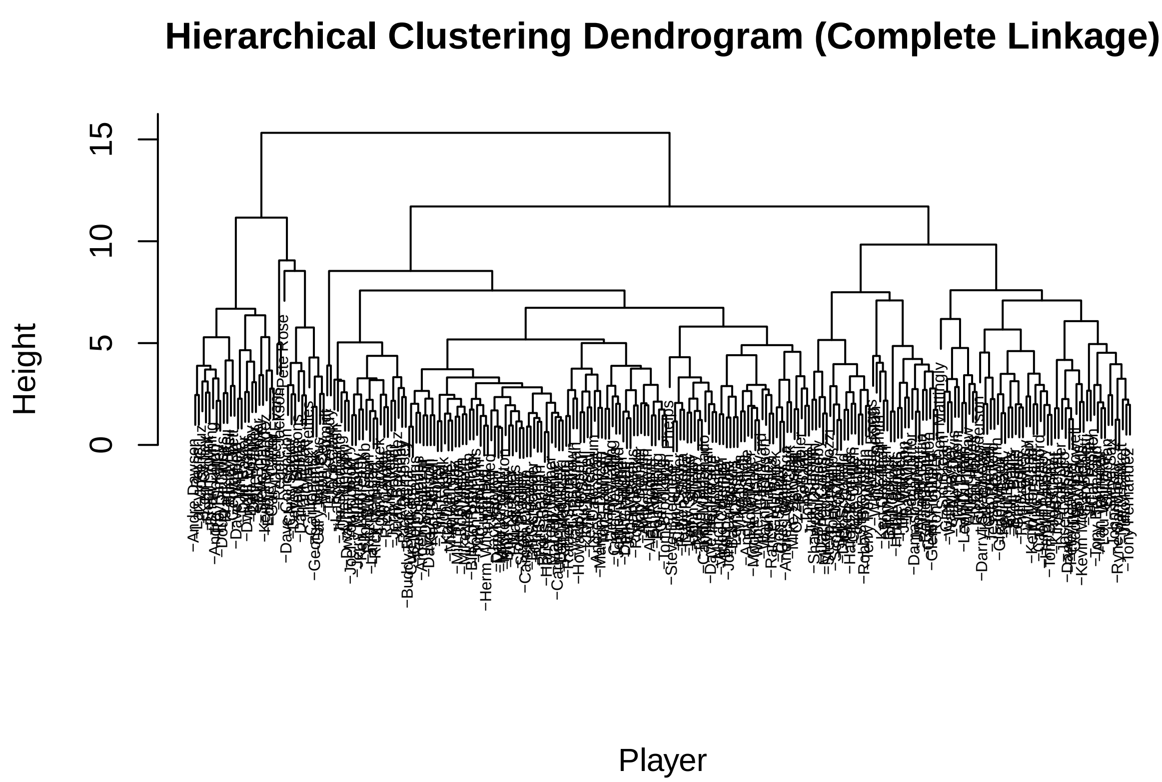
## (c):

```
# Identify the numeric variables (excluding the dummy variables)
numeric_vars <- names(Hitters_dummies)[!names(Hitters_dummies) %in% c("dummies_league", "dummies_division", "d

# Standardize the numeric variables
Hitters_std <- Hitters_dummies
Hitters_std[numeric_vars] <- scale(Hitters_dummies[numeric_vars])
str(Hitters_std)
```

```
## 'data.frame':    263 obs. of  20 variables:
##  $ AtBat            : num  -0.602 0.512 0.627 -0.561 1.292 ...
##  $ Hits             : num  -0.595 0.491 0.735 -0.462 1.356 ...
##  $ HmRun            : num  -0.528 0.729 0.957 -0.185 -0.87 ...
##  $ Runs             : num  -1.204 0.441 0.402 -0.616 0.754 ...
##  $ RBI              : num  -0.5211 0.7925 1.0244 -0.3665 -0.0188 ...
##  $ Walks            : num  -0.0973 1.6063 -0.1894 -0.5117 -0.2815 ...
##  $ Years            : num  1.395 -0.899 0.769 -1.108 0.769 ...
##  $ CAtBat           : num  0.346 -0.452 1.299 -0.989 0.766 ...
##  $ CHits            : num  0.174 -0.409 1.316 -0.958 0.634 ...
##  $ CHmRun           : num  -0.00291 -0.07591 1.89495 -0.69637 -0.6112 ...
##  $ CRuns            : num  -0.121 -0.414 1.409 -0.946 0.422 ...
```

```
##  $ CRBI             : num  0.2585 -0.1992 1.5697 -0.8796 0.0173 ...
##  $ CWalks           : num  0.4345 0.0104 0.355 -0.8607 -0.251 ...
##  $ PutOuts          : num  1.2192 2.1051 -0.324 1.8372 -0.0311 ...
##  $ Assists          : num  -0.522 -0.253 -0.743 -0.543 2.083 ...
##  $ Errors           : num  0.213 0.818 -0.847 -0.695 2.483 ...
##  $ Salary           : num  -0.1351 -0.124 -0.0796 -0.9852 0.4745 ...
##  $ dummies_league   : num  1 0 1 1 0 1 0 1 0 0 ...
##  $ dummies_division : num  1 1 0 0 1 0 1 1 0 0 ...
##  $ dummies_newleague: num  1 0 1 1 0 0 0 1 0 0 ...
```

```r
# Compute the distance matrix using Euclidean distance
dist_matrix <- dist(Hitters_std, method = "euclidean")

# Perform hierarchical clustering using complete linkage
hc_complete <- hclust(dist_matrix, method = "complete")

# Plot the dendrogram
plot(hc_complete, main = "Hierarchical Clustering Dendrogram (Complete Linkage)",
     xlab = "Player", sub = "", cex = 0.5)
```




```r
# Cut the dendrogram at a height that results in two distinct clusters
clusters <- cutree(hc_complete, k = 2)

# Add cluster assignments to the original data. We do means on unstandardized
#data for easy interpretation
Hitters_dummies$Cluster <- clusters
Hitters_std$Cluster <- clusters
```

```r
# Display the cluster assignments
table(clusters)
```

```
## clusters
##   1   2
## 226  37
```

```r
#c1 contains 226 players and c2 contains 37 players.

# Summarize the cluster-specific means of the variables
c_means <- aggregate(Hitters_dummies[, numeric_vars],
                     by = list(Cluster = clusters), mean)
c_means
```

```
##   Cluster    AtBat     Hits    HmRun     Runs      RBI    Walks    Years
## 1       1 397.4027 106.1018 10.65929 53.59292 48.64602 39.08407  6.00000
## 2       2 441.7568 118.3784 17.48649 61.78378 68.83784 53.51351 15.32432
##     CAtBat     CHits    CHmRun    CRuns      CRBI   CWalks  PutOuts   Assists
## 1 1946.588  522.3097  42.76549 257.1947 221.3805 179.5619 288.1903 126.51327
## 2 7000.135 1943.0541 230.94595 996.6216 996.4324 753.2162 306.1081  71.40541
##     Errors   Salary
## 1 9.026549 466.8170
## 2 5.945946 958.0504
```

```r
c_means_std <- aggregate(Hitters_std[, numeric_vars],
                     by = list(Cluster = clusters), mean)
c_means_std
```

```
##   Cluster       AtBat        Hits      HmRun        Runs        RBI      Walks
## 1       1 -0.04235998 -0.03827402 -0.1096800 -0.04511884 -0.1097518 -0.0934705
## 2       2  0.25873935  0.23378182  0.6699375  0.27559074  0.6703758  0.5709279
##         Years     CAtBat      CHits     CHmRun      CRuns       CRBI     CWalks
## 1 -0.2736529 -0.3109248 -0.3083565 -0.3220783 -0.314089 -0.3371943 -0.3056331
## 2  1.6715017  1.8991620  1.8834749  1.9672891  1.918490  2.0596190  1.8668400
##         PutOuts     Assists      Errors     Salary
## 1 -0.009004822  0.05343802  0.06560023 -0.1531944
## 2  0.055002429 -0.32640517 -0.40069329  0.9357283
```

```r
# Summarize the mean salaries of the players in the two clusters
mean_sal <- aggregate(Salary ~ Cluster, data = Hitters_dummies, mean)
mean_sal
```

```
##   Cluster   Salary
## 1       1 466.8170
## 2       2 958.0504
```

```r
mean_sal_std <- aggregate(Salary ~ Cluster, data = Hitters_std, mean)
mean_sal_std
```

```
##   Cluster     Salary
## 1       1 -0.1531944
## 2       2  0.9357283
```

## (d):

```r
# Apply K-means clustering with K = 2
kmeans_result_std <- kmeans(Hitters_std[numeric_vars], centers = 2, nstart = 20)
kmeans_result <- kmeans(Hitters_dummies[numeric_vars], centers = 2, nstart = 20)

# Add the cluster assignments to the original data
Hitters_std$cluster <- kmeans_result$cluster

# Calculate the cluster-specific means of the variables
cluster_means_std_k <- aggregate(Hitters_std[numeric_vars],
                    by = list(cluster = Hitters_std$cluster), FUN = mean)
cluster_means_std_k
```

```
##   cluster      AtBat        Hits       HmRun        Runs         RBI      Walks
## 1       1  0.1335729  0.13579125  0.28008013  0.13481533  0.27292472  0.2876984
## 2       2 -0.0475079 -0.04829689 -0.09961613 -0.04794978 -0.09707116 -0.1023257
##        Years     CAtBat      CHits     CHmRun      CRuns       CRBI     CWalks
## 1  1.3256964  1.4037368  1.3883973  1.1512507  1.3942677  1.3383671  1.3022955
## 2 -0.4715106 -0.4992672 -0.4938114 -0.4094654 -0.4958993 -0.4760172 -0.4631876
##       PutOuts     Assists      Errors     Salary
## 1 -0.04587308 -0.12981007 -0.16875539  0.7117932
## 2  0.01631568  0.04616956  0.06002125 -0.2531636
```

```r
# Calculate the mean salaries of the players in the two clusters
mean_sal_std_k <- aggregate(Hitters_std$Salary,
                   by = list(cluster = Hitters_std$cluster), FUN = mean)
colnames(mean_sal_std_k) <- c("cluster", "mean_salary")
mean_sal_std_k
```

```
##   cluster mean_salary
## 1       1   0.7117932
## 2       2  -0.2531636
```

```r
#---------------UNSTANDARDIZED VERSION FOR INTERPETATION-----------------------
# Add the cluster assignments to the original data
Hitters_dummies$cluster <- kmeans_result$cluster

# Calculate the cluster-specific means of the variables
cluster_means_k <- aggregate(Hitters_dummies[numeric_vars],
```

```
                    by = list(cluster = Hitters_dummies$cluster), FUN = mean)
cluster_means_k
```

```
##   cluster    AtBat     Hits    HmRun     Runs      RBI    Walks     Years
## 1       1 423.3188 113.9565 14.07246 58.18841 58.55072 47.36232 13.666667
## 2       2 396.6443 105.6495 10.74742 53.52062 48.97423 38.89175  5.051546
##     CAtBat     CHits    CHmRun    CRuns     CRBI   CWalks  PutOuts   Assists
## 1 5867.304 1622.1449 163.86957 823.0000 763.2029 604.1449 277.8696  99.92754
## 2 1515.928  402.0979  35.58247 196.9794 176.4897 137.9588 295.2784 125.45876
##     Errors   Salary
## 1 7.478261 857.0291
## 2 8.989691 421.7191
```

```
# Calculate the mean salaries of the players in the two clusters
#unstandardized for easy interpetation
mean_sal_k <- aggregate(Hitters_dummies$Salary,
                   by = list(cluster = Hitters_dummies$cluster), FUN = mean)
colnames(mean_sal_k) <- c("cluster", "mean_salary")
mean_sal_k
```

```
##   cluster mean_salary
## 1       1    857.0291
## 2       2    421.7191
```

## Problem 4:

```
#transformation of Salary variable to Log(Salary) as specified in the question
Hitters_dummies$salary_new <- log(Hitters_dummies$Salary)
str(Hitters_dummies)
```

```
## 'data.frame':    263 obs. of  23 variables:
##  $ AtBat            : int  315 479 496 321 594 185 298 323 401 574 ...
##  $ Hits             : int  81 130 141 87 169 37 73 81 92 159 ...
##  $ HmRun            : int  7 18 20 10 4 1 0 6 17 21 ...
##  $ Runs             : int  24 66 65 39 74 23 24 26 49 107 ...
##  $ RBI              : int  38 72 78 42 51 8 24 32 66 75 ...
##  $ Walks            : int  39 76 37 30 35 21 7 8 65 59 ...
##  $ Years            : int  14 3 11 2 11 2 3 2 13 10 ...
##  $ CAtBat           : int  3449 1624 5628 396 4408 214 509 341 5206 4631 ...
##  $ CHits            : int  835 457 1575 101 1133 42 108 86 1332 1300 ...
##  $ CHmRun           : int  69 63 225 12 19 1 0 6 253 90 ...
##  $ CRuns            : int  321 224 828 48 501 30 41 32 784 702 ...
##  $ CRBI             : int  414 266 838 46 336 9 37 34 890 504 ...
##  $ CWalks           : int  375 263 354 33 194 24 12 8 866 488 ...
##  $ PutOuts          : int  632 880 200 805 282 76 121 143 0 238 ...
##  $ Assists          : int  43 82 11 40 421 127 283 290 0 445 ...
##  $ Errors           : int  10 14 3 4 25 7 9 19 0 22 ...
##  $ Salary           : num  475 480 500 91.5 750 ...
```

```
##  $ dummies_league   : num  1 0 1 1 0 1 0 1 0 0 ...
##  $ dummies_division : num  1 1 0 0 1 0 1 1 0 0 ...
##  $ dummies_newleague: num  1 0 1 1 0 0 0 1 0 0 ...
##  $ Cluster          : int  1 1 2 1 1 1 1 1 2 1 ...
##  $ cluster          : int  2 2 1 2 1 2 2 2 1 1 ...
##  $ salary_new       : num  6.16 6.17 6.21 4.52 6.62 ...
```

## (a):

```r
train_ctrl <- trainControl(method = "LOOCV")

# Train the regression model using the train function defined above
multi_lm_hitters <- train(salary_new ~ AtBat + Hits + HmRun + Runs + RBI + Walks +
                  Years + CAtBat + CHits + CHmRun + CRuns + CRBI + CWalks +
                  PutOuts + Assists + Errors + dummies_division + dummies_league +
                  dummies_newleague, data = Hitters_dummies,
                  method = "lm", trControl = train_ctrl)
summary(multi_lm_hitters)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.22870 -0.45350  0.09424  0.40474  2.77223
##
## Coefficients:
##                      Estimate  Std. Error t value             Pr(>|t|)
## (Intercept)        4.61814299  0.17646244  26.171 < 0.0000000000000002 ***
## AtBat             -0.00298384  0.00123238  -2.421              0.01620 *
## Hits               0.01308450  0.00462164   2.831              0.00503 **
## HmRun              0.01179338  0.01205486   0.978              0.32889
## Runs              -0.00141930  0.00579423  -0.245              0.80670
## RBI               -0.00167546  0.00505579  -0.331              0.74063
## Walks              0.01095506  0.00355439   3.082              0.00229 **
## Years              0.05696428  0.02412778   2.361              0.01902 *
## CAtBat             0.00012830  0.00026288   0.488              0.62596
## CHits             -0.00044139  0.00131125  -0.337              0.73670
## CHmRun            -0.00007809  0.00314371  -0.025              0.98020
## CRuns              0.00151293  0.00145880   1.037              0.30072
## CRBI               0.00013118  0.00134637   0.097              0.92246
## CWalks            -0.00146585  0.00063775  -2.298              0.02239 *
## PutOuts            0.00033890  0.00015054   2.251              0.02526 *
## Assists            0.00062139  0.00042998   1.445              0.14970
## Errors            -0.01196690  0.00853680  -1.402              0.16225
## dummies_division  -0.16564350  0.07846845  -2.111              0.03580 *
## dummies_league     0.28247508  0.15407453   1.833              0.06797 .
```

```
## dummies_newleague -0.17416056  0.15357151  -1.134             0.25788
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6135 on 243 degrees of freedom
## Multiple R-squared:  0.5586, Adjusted R-squared:  0.524
## F-statistic: 16.18 on 19 and 243 DF,  p-value: < 0.0000000000000022
```

```r
# Calculate the test MSE = Root MSE * Root MSE
test_mse <- multi_lm_hitters$results$RMSE^2
test_mse
```

```
## [1] 0.4214063
```

# (b):

```r
#PCR model to choose optimal M using LOOCV
pcr_fit <- pcr(salary_new ~ AtBat + Hits + HmRun + Runs + RBI + Walks +
                  Years + CAtBat + CHits + CHmRun + CRuns + CRBI + CWalks +
                  PutOuts + Assists + Errors + dummies_division + dummies_league +
                  dummies_newleague, data = Hitters_dummies, scale = TRUE,
               validation = "LOO", center = TRUE)
summary(pcr_fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 263 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8909   0.6515   0.6535   0.6508   0.6512   0.6475   0.6492
## adjCV       0.8909   0.6515   0.6535   0.6508   0.6512   0.6475   0.6492
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.6484   0.6453   0.6478    0.6466    0.6509    0.6483    0.6535
## adjCV   0.6484   0.6453   0.6477    0.6466    0.6509    0.6483    0.6534
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.6462    0.6490    0.6406    0.6433    0.6462    0.6492
## adjCV    0.6461    0.6489    0.6405    0.6432    0.6461    0.6490
##
## TRAINING: % variance explained
##             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X             38.31    60.16    70.84    79.03    84.29    88.63    92.26
## salary_new    47.29    47.55    48.46    48.70    49.67    49.96    50.50
##             8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X             94.96    96.28     97.26     97.98     98.65     99.15     99.47
## salary_new    51.42    51.53     52.07     52.26     52.85     53.09     54.17
```

```
##             15 comps  16 comps  17 comps  18 comps  19 comps
## X              99.75     99.89     99.97     99.99    100.00
## salary_new     54.18     55.35     55.79     55.79     55.86
```

```r
MSEP(pcr_fit) #MSE of prediction
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.7937   0.4245   0.4271   0.4236    0.424   0.4193   0.4215
## adjCV       0.7937   0.4245   0.4271   0.4236    0.424   0.4192   0.4215
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.4205   0.4164   0.4196    0.4181    0.4237    0.4203    0.4271
## adjCV   0.4204   0.4164   0.4195    0.4180    0.4236    0.4202    0.4269
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.4176    0.4211    0.4104    0.4138    0.4176    0.4214
## adjCV    0.4174    0.4210    0.4103    0.4137    0.4175    0.4213
```

```r
sqrt(MSEP(pcr_fit)$val[1, 1,]) #RMSE for first component
```

```
## (Intercept)     1 comps     2 comps     3 comps     4 comps     5 comps
##   0.8908877   0.6515160   0.6535356   0.6508368   0.6511891   0.6474972
##     6 comps     7 comps     8 comps     9 comps    10 comps    11 comps
##   0.6492347   0.6484410   0.6453110   0.6477616   0.6466231   0.6509002
##    12 comps    13 comps    14 comps    15 comps    16 comps    17 comps
##   0.6483253   0.6534991   0.6461946   0.6489591   0.6406307   0.6432763
##    18 comps    19 comps
##   0.6462255   0.6491581
```

```r
#Identify the number of components that minimize the MSEP
best_m_pcr <- which.min(MSEP(pcr_fit)$val[1, 1,])
best_m_pcr
```

```
## 16 comps
##       17
```

```r
#Fit pcr model with M = 16 to get corresponding test MSE
pcr_fit_best <- pcr(salary_new ~ AtBat + Hits + HmRun + Runs + RBI + Walks +
                  Years + CAtBat + CHits + CHmRun + CRuns + CRBI + CWalks +
                  PutOuts + Assists + Errors + dummies_division + dummies_league +
                  dummies_newleague, data = Hitters_dummies, scale = TRUE,
                  validation = "LOO", center = TRUE, ncomp = best_m_pcr)
summary(pcr_fit_best)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 263 leave-one-out segments.
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8909   0.6515   0.6535   0.6508   0.6512   0.6475   0.6492
## adjCV       0.8909   0.6515   0.6535   0.6508   0.6512   0.6475   0.6492
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.6484   0.6453   0.6478    0.6466    0.6509    0.6483    0.6535
## adjCV   0.6484   0.6453   0.6477    0.6466    0.6509    0.6483    0.6534
##        14 comps  15 comps  16 comps  17 comps
## CV       0.6462    0.6490    0.6406    0.6433
## adjCV    0.6461    0.6489    0.6405    0.6432
##
## TRAINING: % variance explained
##             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X             38.31    60.16    70.84    79.03    84.29    88.63    92.26
## salary_new    47.29    47.55    48.46    48.70    49.67    49.96    50.50
##             8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X             94.96    96.28     97.26     97.98     98.65     99.15     99.47
## salary_new    51.42    51.53     52.07     52.26     52.85     53.09     54.17
##             15 comps  16 comps  17 comps
## X              99.75     99.89     99.97
## salary_new     54.18     55.35     55.79
```

```
#Test MSE
MSEP(pcr_fit_best)$val[1, 1, best_m_pcr]
```

```
## [1] 0.4104077
```

# (c):

```
#PCR model to choose optimal M using LOOCV
pls_fit <- plsr(salary_new ~ AtBat + Hits + HmRun + Runs + RBI + Walks +
                  Years + CAtBat + CHits + CHmRun + CRuns + CRBI + CWalks +
                  PutOuts + Assists + Errors + dummies_division + dummies_league +
                  dummies_newleague, data = Hitters_dummies, scale = TRUE,
               validation = "LOO", center = TRUE)
summary(pls_fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: kernelpls
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 263 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8909   0.6479   0.6477   0.6470   0.6459   0.6462   0.6462
## adjCV       0.8909   0.6479   0.6477   0.6469   0.6459   0.6462   0.6461
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.6474   0.6476   0.6441    0.6438    0.6432    0.6430    0.6440
```

```
## adjCV   0.6473   0.6475   0.6440    0.6437    0.6431    0.6429    0.6439
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.6454    0.6457    0.6453    0.6523    0.6508    0.6492
## adjCV    0.6453    0.6455    0.6452    0.6522    0.6507    0.6490
##
## TRAINING: % variance explained
##             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X             38.21    48.95    64.47    73.93    78.21    83.29    87.67
## salary_new    48.67    51.18    52.11    52.89    53.86    54.48    54.85
##             8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X             90.41    92.39     95.48     96.92     97.69     98.35     98.73
## salary_new    55.17    55.47     55.56     55.68     55.75     55.78     55.80
##             15 comps  16 comps  17 comps  18 comps  19 comps
## X              99.05     99.68     99.71     99.97    100.00
## salary_new     55.81     55.82     55.85     55.86     55.86
```

```
MSEP(pls_fit) #MSE of prediction
```

```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.7937   0.4198   0.4195   0.4186   0.4172   0.4176   0.4176
## adjCV       0.7937   0.4198   0.4195   0.4185   0.4171   0.4175   0.4175
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.4191   0.4193   0.4149    0.4145    0.4138    0.4135    0.4148
## adjCV   0.4189   0.4192   0.4148    0.4144    0.4136    0.4134    0.4146
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV       0.4166    0.4169    0.4165    0.4255    0.4236    0.4214
## adjCV    0.4164    0.4167    0.4163    0.4253    0.4234    0.4213
```

```
sqrt(MSEP(pls_fit)$val[1, 1,]) #RMSE for first component
```

```
## (Intercept)     1 comps     2 comps     3 comps     4 comps     5 comps
##   0.8908877   0.6479324   0.6477197   0.6469999   0.6459317   0.6462446
##     6 comps     7 comps     8 comps     9 comps    10 comps    11 comps
##   0.6462333   0.6473558   0.6475602   0.6441164   0.6438118   0.6432462
##    12 comps    13 comps    14 comps    15 comps    16 comps    17 comps
##   0.6430299   0.6440156   0.6454307   0.6456531   0.6453407   0.6522957
##    18 comps    19 comps
##   0.6508126   0.6491581
```

```
# Identify the number of components that minimize the MSEP
best_m_pls <- which.min(MSEP(pls_fit)$val[1, 1,])
best_m_pls
```

```
## 12 comps
##       13
```

```
#Fit pla model with M = 12 to get corresponding test MSE
pls_fit_best <- pcr(salary_new ~ AtBat + Hits + HmRun + Runs + RBI + Walks +
                  Years + CAtBat + CHits + CHmRun + CRuns + CRBI + CWalks +
```

```r
                  PutOuts + Assists + Errors + dummies_division + dummies_league +
                  dummies_newleague, data = Hitters_dummies, scale = TRUE,
                  validation = "LOO", center = TRUE, ncomp = best_m_pls)
summary(pls_fit_best)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 263 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8909   0.6515   0.6535   0.6508   0.6512   0.6475   0.6492
## adjCV       0.8909   0.6515   0.6535   0.6508   0.6512   0.6475   0.6492
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.6484   0.6453   0.6478    0.6466    0.6509    0.6483    0.6535
## adjCV   0.6484   0.6453   0.6477    0.6466    0.6509    0.6483    0.6534
##
## TRAINING: % variance explained
##             1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X             38.31    60.16    70.84    79.03    84.29    88.63    92.26
## salary_new    47.29    47.55    48.46    48.70    49.67    49.96    50.50
##             8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X             94.96    96.28     97.26     97.98     98.65     99.15
## salary_new    51.42    51.53     52.07     52.26     52.85     53.09
```

```r
#Test MSE
MSEP(pls_fit_best)$val[1, 1, best_m_pls]
```

```
## [1] 0.4203257
```

## (d):

```r
# Set response and predictor matrices. The design matrix X includes 1's in the
# first column. We consider the predictor matrix without these 1's
y_resp <- Hitters_dummies$salary_new
x_pred <- model.matrix(salary_new ~ AtBat + Hits + HmRun + Runs + RBI + Walks +
                  Years + CAtBat + CHits + CHmRun + CRuns + CRBI + CWalks +
                  PutOuts + Assists + Errors + dummies_division + dummies_league +
                  dummies_newleague, data = Hitters_dummies)[, -1]

# Define a grid of lambda values
grid <- 10^seq(10, -2, length = 100)

# Fit ridge regression model for each lambda (penalty parameter) on the grid.
ridge_reg_hitters <- glmnet(x_pred, y_resp, alpha = 0, lambda = grid)
```

```r
# Perform LOOCV to find the penalty parameter lambda
loocv_ridge_hitters <- cv.glmnet(x_pred, y_resp, alpha = 0, nfolds = nrow(Hitters),
                         lambda = grid, grouped = FALSE, type.measure = "mse")

# Penalty parameter = minimum value of lambda
penalty_par_hitters <- loocv_ridge_hitters$lambda.min
penalty_par_hitters
```

```
## [1] 0.05336699
```

```r
# Calculate test MSE using built-in function "cvm" which stands for
# Cross Validation MSE which is very easy to calculate (it is included in the
# glmnet package)
test_err_new <- min(loocv_ridge_hitters$cvm)
test_err_new
```

```
## [1] 0.4082671
```