ECON 573 Problem Set 5

Harvey Duperier

2022-11-15

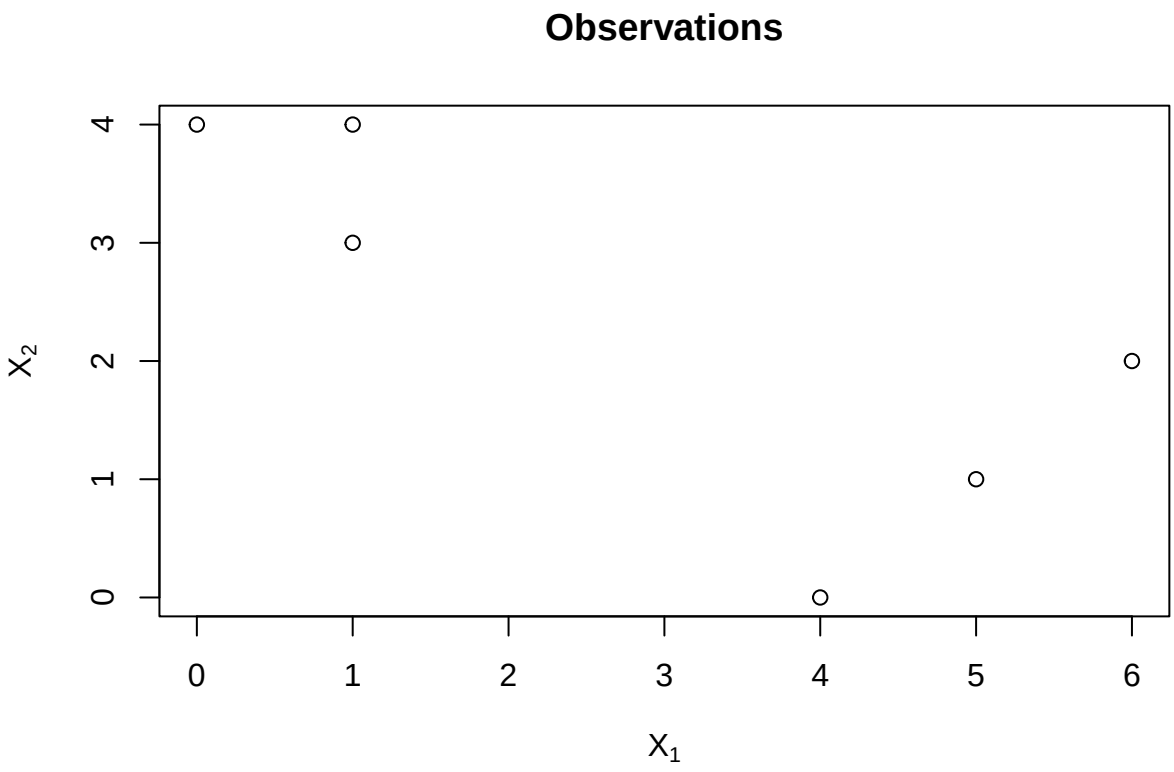
Part I

Chapter 12: Exercise 3

In this problem you will perform K-means clustering manually with K = 2, on a small example with n=6 observations and p = 2 features. The observations are as follows.

a). Plot the observations.

```
obs <- cbind(c(1, 1, 0, 5, 6, 4), c(4, 3, 4, 1, 2, 0))
plot(obs[,1], obs[,2], main='Observations', xlab=expression(X[1]), ylab=expression(X[2]))
```

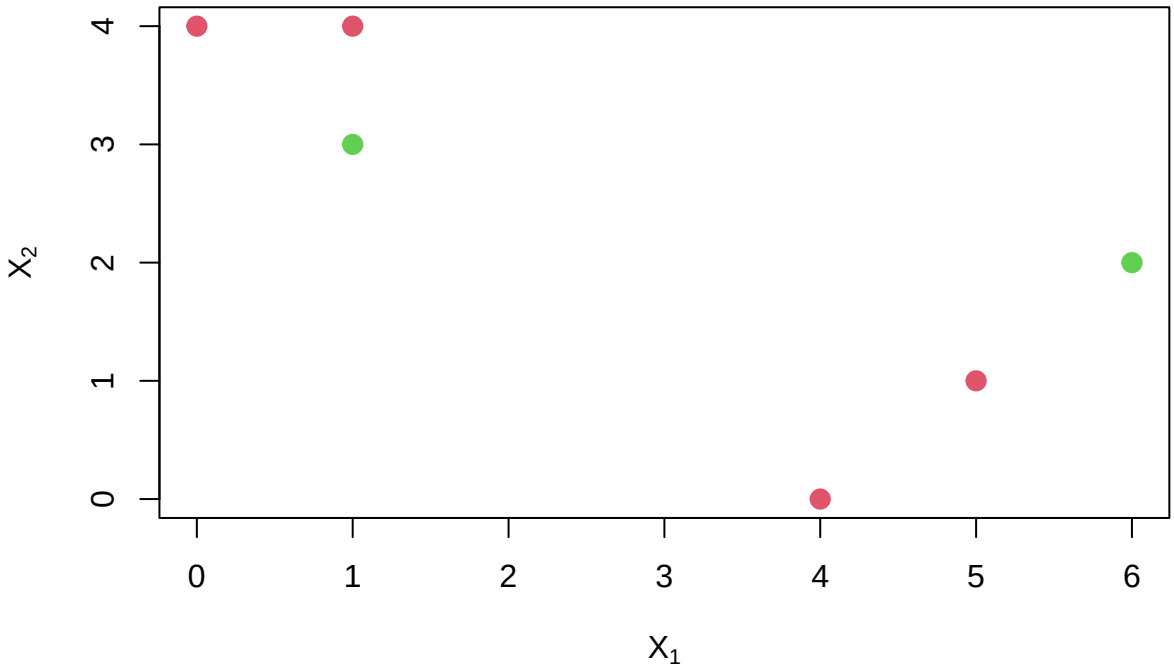


b). Randomly assign a cluster label to each observation. You can use the sample() command in R to do this. Report the cluster labels for each observation.

```r
set.seed(1)
c.labels<- sample(2, nrow(obs), replace=T)
c.labels
```

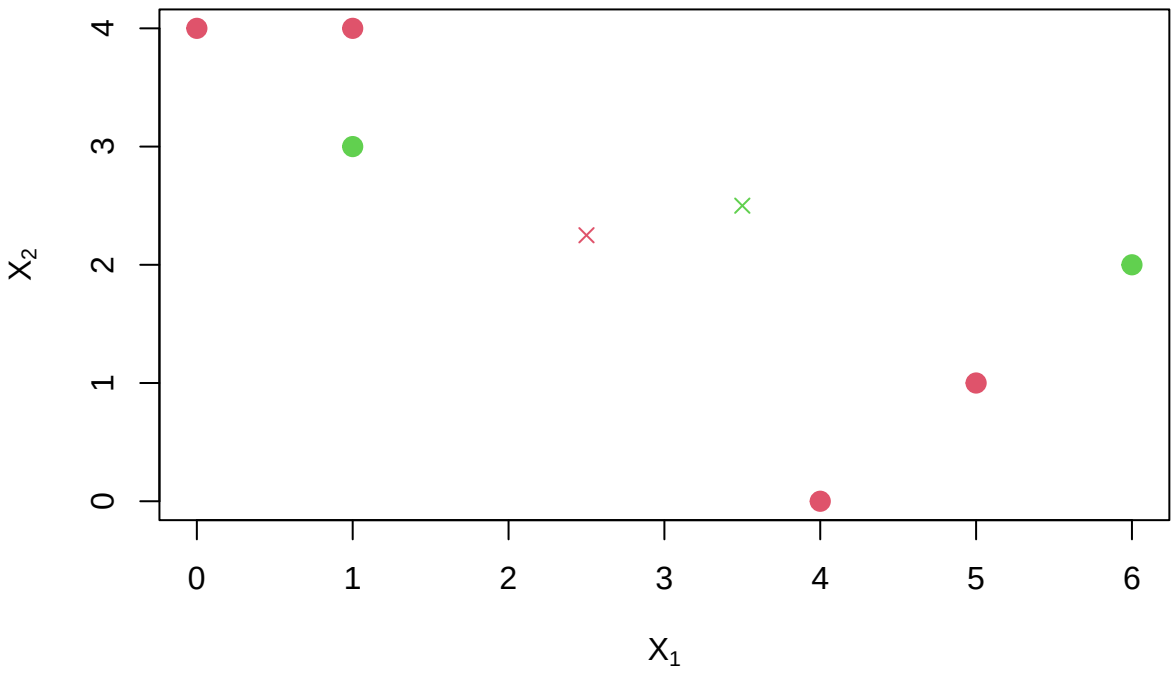
```
## [1] 1 2 1 1 2 1
```

```r
plot(obs[,1], obs[,2], col=(c.labels+1), pch = 20, cex=2, xlab=expression(X[1]), ylab=expression(X[2]))
```



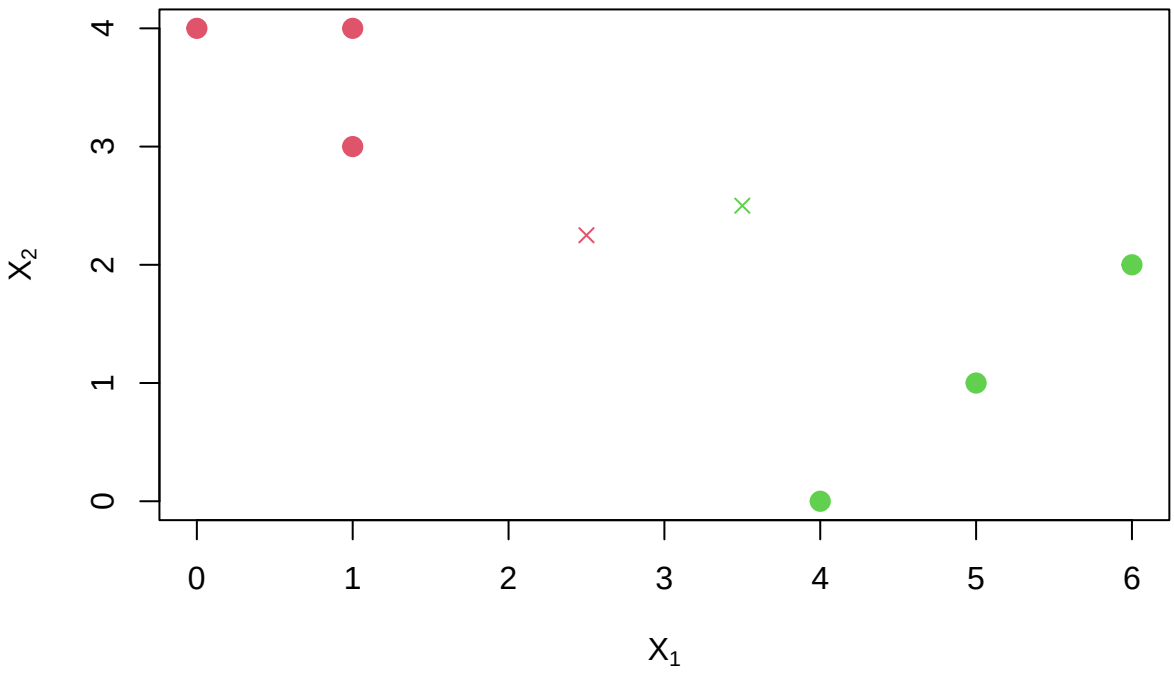
c). Compute the centroid for each cluster

```r
centroid1<- c(mean(obs[c.labels == 1, 1]), mean(obs[c.labels == 1, 2]))
centroid2<- c(mean(obs[c.labels == 2, 1]), mean(obs[c.labels == 2, 2]))
plot(obs[,1], obs[,2], col=(c.labels + 1), pch = 20, cex = 2, xlab=expression(X[1]), ylab=expression(X[
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



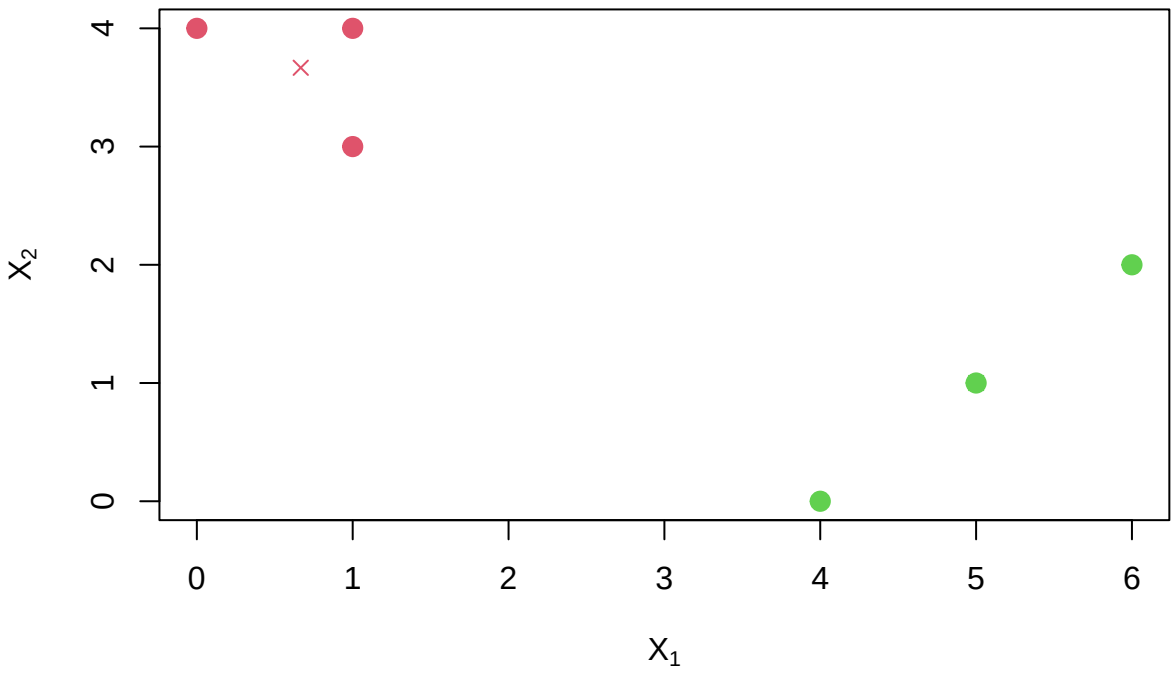
d). Assign each observation to the centroid to which it is closest, in terms of Euclidean distance. Report the cluster labels for each observation.

```
c.labels <- c(1, 1, 1, 2, 2, 2)
plot(obs[, 1], obs[, 2], col = (c.labels + 1), xlab=expression(X[1]), ylab=expression(X[2]), pch = 20, 
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



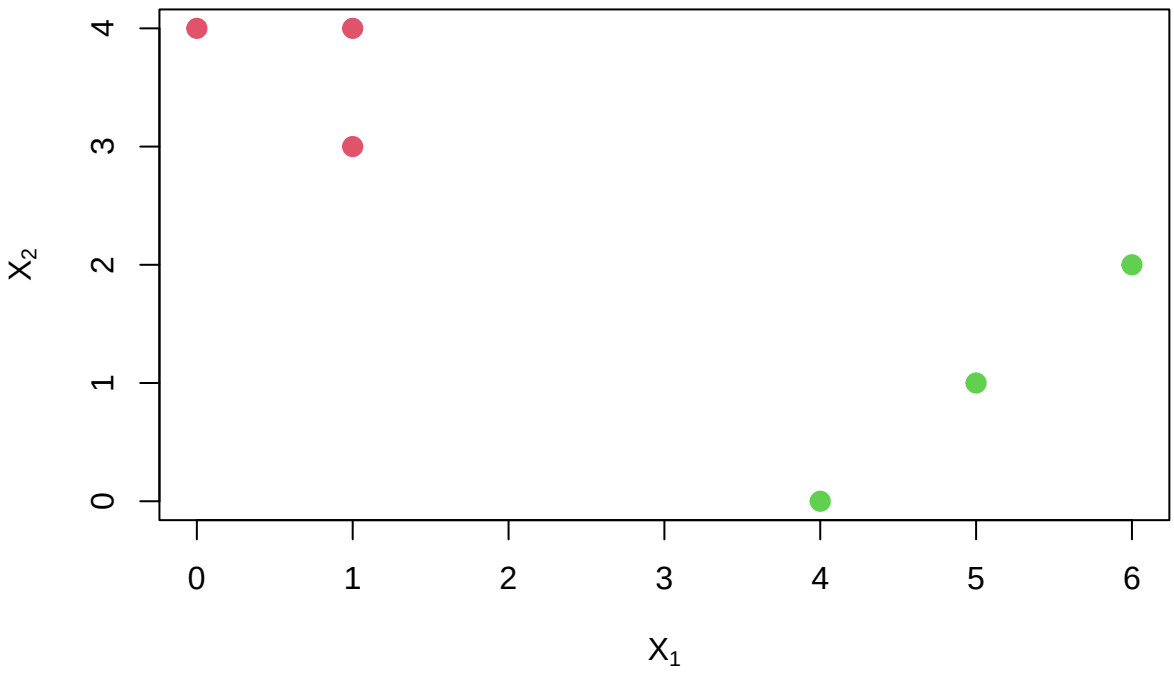
e). Repeat (c) and (d) until the answers obtained stop changing.

```
centroid1 <- c(mean(obs[c.labels == 1, 1]), mean(obs[c.labels == 1, 2]))
centroid2 <- c(mean(obs[c.labels == 2, 1]), mean(obs[c.labels == 2, 2]))
plot(obs[,1], obs[,2], col=(c.labels + 1), xlab=expression(X[1]), ylab=expression(X[2]), pch = 20, cex =
points(centroid1[1], centroid1[2], col = 2, pch = 4)
points(centroid2[1], centroid2[2], col = 3, pch = 4)
```



f). In your plot from (a), color the observations according to the cluster labels obtained.

```
plot(obs[, 1], obs[, 2], col=(c.labels + 1), xlab=expression(X[1]), ylab=expression(X[2]), pch = 20, ce
```



Chapter 12: Exercise 10

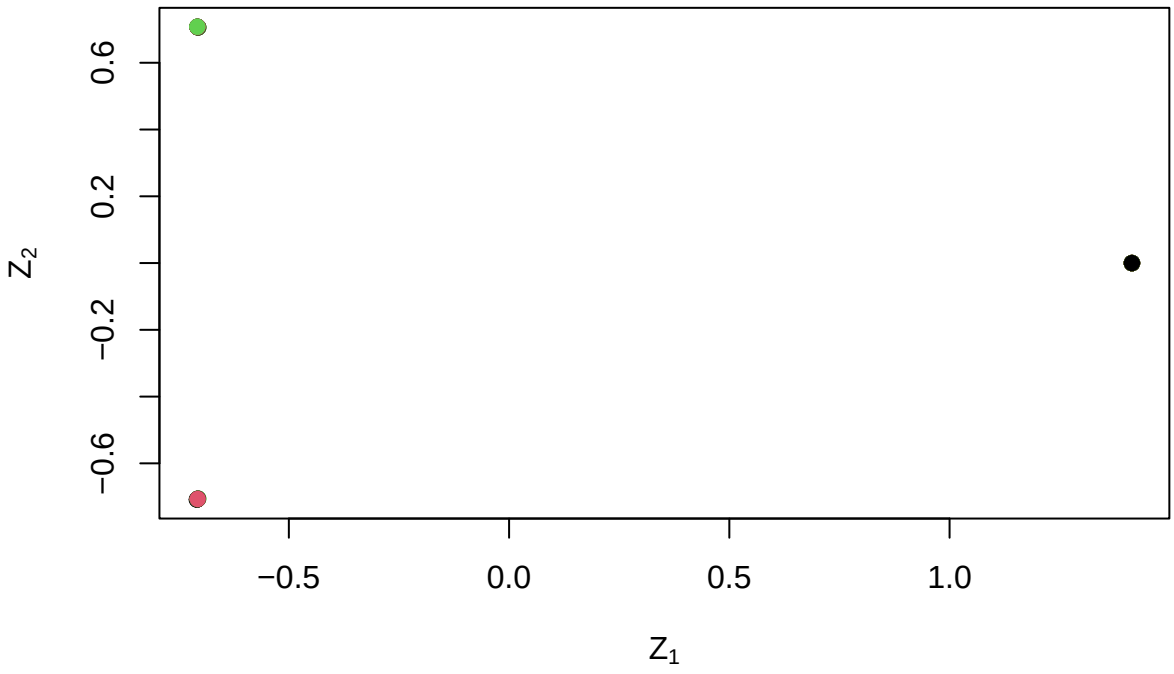
In this problem, you will generate simulated data, and then perform PCA and K-means clustering on the data.

a). Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables.

```
set.seed(2)
x <- matrix(rnorm(20 * 3 * 50, mean = 0, sd = 0.001), ncol = 50)
x[1:20, 2] <- 1
x[21:40, 1] <- 2
x[21:40, 2] <- 2
x[41:60, 1] <- 1
true.labels <- c(rep(1, 20), rep(2, 20), rep(3, 20))
```

b). Perform PCA on the 60 observations and plot the first two principal component score vectors. Use a different color to indicate the observations in each of the three classes. If the three classes appear separated in this plot, then continue on to part (c). If not, then return to part (a) and modify the simulation so that there is greater separation between the three classes. Do not continue to part (c) until the three classes show at least some separation in the first two principal component score vectors.

```
pr.out <- prcomp(x)
plot(pr.out$x[, 1:2], col = 1:3, xlab=expression(Z[1]), ylab=expression(Z[2]), pch = 19)
```



c). Perform K-means clustering of the observations with K = 3. How well do the clusters that you obtained in K-means clustering compare to the true class labels?

```
km.out <- kmeans(x, 3, nstart = 20)
table(true.labels, km.out$cluster)
```

```
##
## true.labels  1  2  3
##           1  0  0 20
##           2 20  0  0
##           3  0 20  0
```

The clusters obtained in K-means clustering are perfectly clustered.

d). Perform K-means clustering with K = 2. Describe your results.

```
km.out <- kmeans(x, 2, nstart = 20)
table(true.labels, km.out$cluster)
```

```
##
## true.labels  1  2
##           1 20  0
```

```
##            2  0 20
##            3 20  0
```

K-means clustering with K=2, compared to K=3, means that one of the three clusters will be absorbed into two clusters.

e). Now perform K-means clustering with K = 4, and describe your results.

```
km.out <- kmeans(x, 4, nstart = 20)
table(true.labels, km.out$cluster)
```

```
##
## true.labels  1  2  3  4
##            1 11  9  0  0
##            2  0  0 20  0
##            3  0  0  0 20
```

K-means clustering with K=4, compared to K=3, means that the first cluster will be split into two.

f). Now perform K-means clustering with K = 3 on the first two principal component score vectors, rather than on the raw data. That is, perform K-means clustering on the 60 × 2 matrix of which the first column is the first principal component score vector, and the second column is the second principal component score vector. Comment on the results.

```
km.out <- kmeans(pr.out$x[, 1:2], 3, nstart = 20)
table(true.labels, km.out$cluster)
```

```
##
## true.labels  1  2  3
##            1  0  0 20
##            2  0 20  0
##            3 20  0  0
```

Similarly to performing with K=3, all observations are perfectly clustered.

g). Using the scale() function, perform K-means clustering with K = 3 on the data after scaling each variable to have standard deviation one. How do these results compare to those obtained in (b)? Explain.

```
km.out <- kmeans(scale(x), 3, nstart = 20)
table(true.labels, km.out$cluster)
```

```
##
## true.labels  1  2  3
##            1  9  2  9
##            2  2 18  0
##            3  7  1 12
```

Unscaled data clearly produces better results than scaled data, this is due to the fact that scaling affects the distance between observations.

Part II

1). Discuss correlation amongst variables of fx. How does this relate to the applicability of PCA? The definition of PCA application is that PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated. Due to this definition, the different variables need to be distinct from each other to be interpretable, otherwise they only represent random directions, therefore if correlation amongst variables increases, the applicability of PCA decreases.

2). Fit, plot, and interpret principal components (PCs).

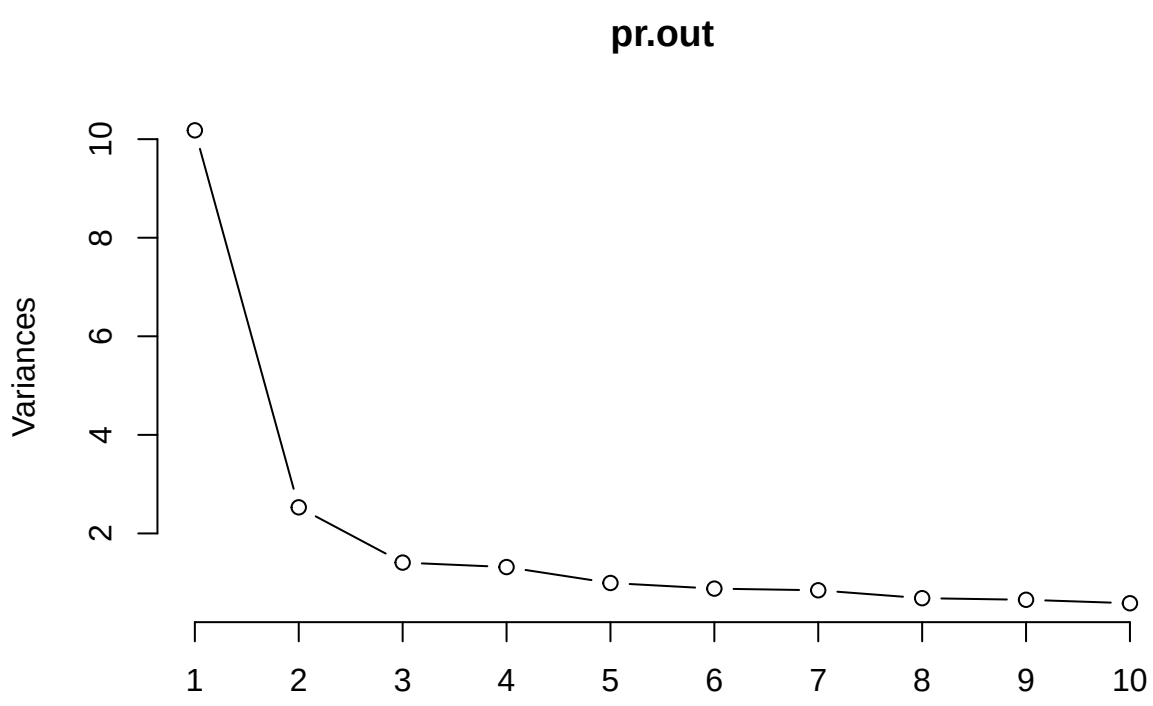
```
WD<-getwd()
if (!is.null(WD)) setwd(WD)
fx<- read.csv("FXmonthly.csv")
fx<- (fx[2:120,]-fx[1:119,])/(fx[1:119,])
pr.out <- prcomp(fx, retx=TRUE, center=TRUE, scale.= TRUE, tol=NULL)
print(pr.out)
```

```
## Standard deviations (1, .., p=23):
##  [1] 3.19041033 1.59045449 1.18680366 1.14792414 0.99740480 0.93815170
##  [7] 0.92009355 0.82835449 0.80840658 0.76389669 0.69184572 0.65916991
## [13] 0.58024379 0.56011701 0.55254251 0.50189768 0.44624080 0.41833998
## [19] 0.38807591 0.33723734 0.30770693 0.25804092 0.01556967
##
## Rotation (n x k) = (23 x 23):
##                PC1         PC2         PC3           PC4          PC5
## exalus -0.27546307  0.14479617 -0.12228468  0.0217819919 -0.007801149
## exbzus -0.15853455  0.32942973 -0.03056533 -0.2898382828 -0.156807395
## excaus -0.22144219  0.17928800 -0.22127964  0.0196985460 -0.105703079
## exchus -0.06278227 -0.07998548  0.12806680 -0.6360500274  0.297181357
## exdnus -0.27559139 -0.20539756 -0.11144766 -0.0301583530  0.162067676
## exhkus -0.03365354 -0.26187043 -0.41635552 -0.0511642205 -0.443654802
## exinus -0.20354162  0.23338613  0.19362879  0.0511177968 -0.267705691
## exjpus -0.09001974 -0.46262858  0.12181056  0.0310283474 -0.329748091
## exkous -0.25300324  0.19827588  0.10846851  0.1842564229 -0.067277121
## exmaus -0.21139752  0.11342703  0.17375865 -0.1434866772 -0.098178628
## exmxus -0.16007639  0.43007644 -0.08772852 -0.1262327052  0.095059835
## exnzus -0.24517875  0.08032252 -0.06208289  0.1110244112  0.064942337
## exnous -0.26564269 -0.05262103 -0.10635087  0.0425693024  0.190248746
## exsius -0.26460130 -0.11924763  0.17990630 -0.0703816060 -0.080900901
## exsfus -0.18697409  0.11982916 -0.03745871  0.2033496930 -0.140405916
## exslus -0.01275494 -0.06461653  0.49619725 -0.2471249938  0.058810550
## exsdus -0.28516719 -0.07813666 -0.10657794 -0.0007402518  0.161682137
## exszus -0.24343260 -0.32873137 -0.06637626 -0.0321441994  0.152882495
## extaus -0.21404796 -0.06904283  0.28627494  0.0030857827 -0.133255240
## exthus -0.16920700 -0.07899087  0.34541114  0.0683071257 -0.339018391
## exukus -0.24171405 -0.07811805 -0.10858297  0.0826302415  0.177219157
## exvzus  0.01107475  0.01040368 -0.30832567 -0.5458427935 -0.372688401
## exeuus -0.27509742 -0.20543403 -0.10937298 -0.0316181989  0.162867979
##                PC6          PC7         PC8           PC9          PC10
## exalus  0.03215871 -0.044288886 -0.16338984 -0.1429058177  0.008025892
## exbzus  0.10112085 -0.058579708 -0.33040687 -0.2228143390 -0.120329386
## excaus  0.15934195 -0.277029140 -0.04146645 -0.2984174002  0.055761979
## exchus -0.40250468 -0.045080839 -0.03145013 -0.3101650652  0.286746360
```

```
## exdnus  0.07398911  0.092062880  0.05194045  0.0788155739 -0.024496638
## exhkus -0.05037683 -0.576631337 -0.04268456  0.0722800464  0.118111278
## exinus -0.15722012 -0.050373474  0.22137800  0.2592382913  0.151009908
## exjpus -0.18546223  0.068925390  0.03245952 -0.2990550187 -0.204406901
## exkous -0.10620128  0.054041138  0.08757345  0.0426141115 -0.185594730
## exmaus -0.23077745 -0.181042017  0.15710985  0.4805951716  0.305777190
## exmxus -0.12840199 -0.021249786  0.05832373  0.0189103890 -0.293207045
## exnzus  0.12278598  0.064269289 -0.34185810 -0.0450008456  0.129437454
## exnous  0.05731097 -0.073748346  0.14082262  0.0007473203  0.097888247
## exsius -0.07664338  0.040765379  0.04396424 -0.0021830511 -0.041421881
## exsfus  0.05404218  0.189239654  0.58932421 -0.4755126101  0.230947626
## exslus  0.66757228 -0.374266627  0.20978288 -0.0392503525 -0.068965587
## exsdus  0.02695619  0.001947436 -0.01868591  0.0178879308  0.022352919
## exszus -0.01082180  0.039385994  0.05167607  0.0635126959 -0.055412995
## extaus -0.22321532 -0.103068729 -0.07458315 -0.0109473806 -0.581237192
## exthus  0.13629604  0.312756208 -0.42784082 -0.0151855460  0.391919597
## exukus  0.20432583  0.055671928 -0.10964111  0.2484124337 -0.049828271
## exvzus  0.25242911  0.472199995  0.18626381  0.2020995772 -0.150778205
## exeuus  0.07527599  0.090711742  0.05293872  0.0807211837 -0.026490373
##                PC11          PC12         PC13         PC14         PC15
## exalus -0.244114132  0.1749280638 -0.029362691 -0.082532097  0.12223967
## exbzus  0.418272914  0.0954361637 -0.396064322 -0.414847305 -0.10951785
## excaus -0.030605114  0.3459230716  0.292856289  0.340400368 -0.25555647
## exchus -0.034359010 -0.1919890512  0.245634754 -0.011463375 -0.01710255
## exdnus  0.134316933  0.0683741025  0.053357349 -0.033888529  0.22159502
## exhkus  0.005662667 -0.3649818337 -0.006741833  0.022296404  0.14304379
## exinus  0.271585051  0.1313835635  0.544848293 -0.332182230  0.14233834
## exjpus  0.003834740  0.3598301666  0.105967160 -0.056823592 -0.23402343
## exkous  0.114006859 -0.0318817332  0.031148805  0.194315364  0.15572137
## exmaus -0.254570115  0.2128603221 -0.395024360 -0.025597423 -0.21648556
## exmxus -0.017258931 -0.1602094069  0.109098987  0.354220172 -0.03709317
## exnzus -0.579434255  0.0169399986  0.096147523 -0.250219530  0.25710002
## exnous  0.302601460 -0.0797283345 -0.250958077  0.227026906  0.07134271
## exsius -0.175888966  0.1084893855 -0.188812255  0.321761170 -0.12021718
## exsfus -0.128343244 -0.3141438328 -0.174969350 -0.208196103 -0.08060569
## exslus -0.069072852 -0.0113364772  0.060321717 -0.001541858  0.08784520
## exsdus  0.087415543 -0.0265911475 -0.007336338  0.021236185  0.12586201
## exszus  0.085862319  0.1344172270 -0.097933120 -0.119830884 -0.02034380
## extaus -0.164102278 -0.3521865840 -0.041445585 -0.102357778  0.08461241
## exthus  0.183155759 -0.2639261235  0.015819207  0.311320981  0.02958741
## exukus -0.006118648 -0.3360624137  0.239995917 -0.191965068 -0.71483403
## exvzus -0.145630506 -0.0002939743  0.050175315  0.065595102  0.05346178
## exeuus  0.131013195  0.0672049087  0.049150252 -0.039879154  0.22887886
##                PC16          PC17         PC18         PC19          PC20
## exalus  0.072350096  0.0522036929  0.033664233 -0.150476472  0.1304089052
## exbzus -0.114287306  0.0950343355 -0.034144708  0.013882222 -0.0349172736
## excaus  0.352855176 -0.0787399215 -0.239498868  0.199676680  0.0435634615
## exchus  0.053538525  0.1123854762  0.089354032  0.115788742  0.0128504533
## exdnus -0.163231661  0.0362712898 -0.262261745  0.086362643  0.1635808997
## exhkus -0.186945059  0.0846564876  0.026854951  0.031591105 -0.0113178037
## exinus  0.055059226  0.0491936744 -0.021368033 -0.275256975  0.0205077645
## exjpus -0.243033680 -0.1862950831  0.372891982 -0.083635972 -0.0143333579
## exkous -0.057885412  0.4006139853  0.386349944  0.613732934 -0.1334771138
## exmaus  0.016934150 -0.2232275019  0.005568094  0.185482779  0.0213799306
```

```
## exmxus -0.526434357 -0.3986103462  0.035971153 -0.157346237  0.1254894789
## exnzus -0.094579917  0.0066172291  0.268297084 -0.046740500  0.0514695407
## exnous  0.352748663 -0.0004480539  0.501512045 -0.385324096  0.2783989367
## exsius -0.167246974  0.5698646757 -0.274643524 -0.386764253 -0.1390947042
## exsfus -0.057774611 -0.0662227361 -0.128617114  0.049781013  0.0024452249
## exslus -0.116485162 -0.0392655119  0.107951181  0.031966215 -0.0205783021
## exsdus  0.088353214 -0.2760360030  0.010086428 -0.108747144 -0.8659826957
## exszus  0.006767012 -0.2383167662 -0.060223339  0.264248428  0.1645885233
## extaus  0.440869528 -0.1360929350 -0.204629833 -0.014674522  0.0877380231
## exthus -0.014907593 -0.2180430046 -0.082729761  0.061913225  0.0819565239
## exukus -0.037627230  0.1510918203  0.091532710 -0.000154865  0.0006261504
## exvzus  0.195008057  0.0157393567  0.112266093  0.017177910 -0.0229464215
## exeuus -0.172426073  0.0417619031 -0.268714987  0.084236775  0.1621378628
##                PC21         PC22          PC23
## exalus  0.786137214 -0.240674373  0.0007358666
## exbzus -0.134445934  0.045702510 -0.0027754502
## excaus -0.188947355  0.095733472  0.0054518819
## exchus  0.031007463  0.010470933  0.0009983123
## exdnus  0.054431953  0.314050732 -0.7144510236
## exhkus  0.005145551 -0.056706492 -0.0011804208
## exinus -0.080360083 -0.137902261  0.0017414794
## exjpus  0.056305374  0.200031093 -0.0001401379
## exkous  0.054316910 -0.027713456  0.0011431844
## exmaus  0.051300931  0.213322588 -0.0028290719
## exmxus -0.044886714 -0.074301702  0.0007559027
## exnzus -0.439294682  0.073540979 -0.0021162469
## exnous -0.108950247  0.088673430  0.0039482145
## exsius -0.179376734 -0.187388010 -0.0014148206
## exsfus -0.017395065  0.007749965 -0.0011427060
## exslus  0.059549330 -0.029461460 -0.0024543446
## exsdus  0.049615484  0.012743646  0.0001171944
## exszus -0.207801727 -0.736156487  0.0111483512
## extaus -0.032488463  0.108068663 -0.0014261045
## exthus  0.069281502 -0.041590219  0.0032267978
## exukus  0.106124740  0.045667829  0.0017776514
## exvzus -0.007562794 -0.015821450  0.0014644597
## exeuus  0.057815999  0.333012820  0.6995248348
```

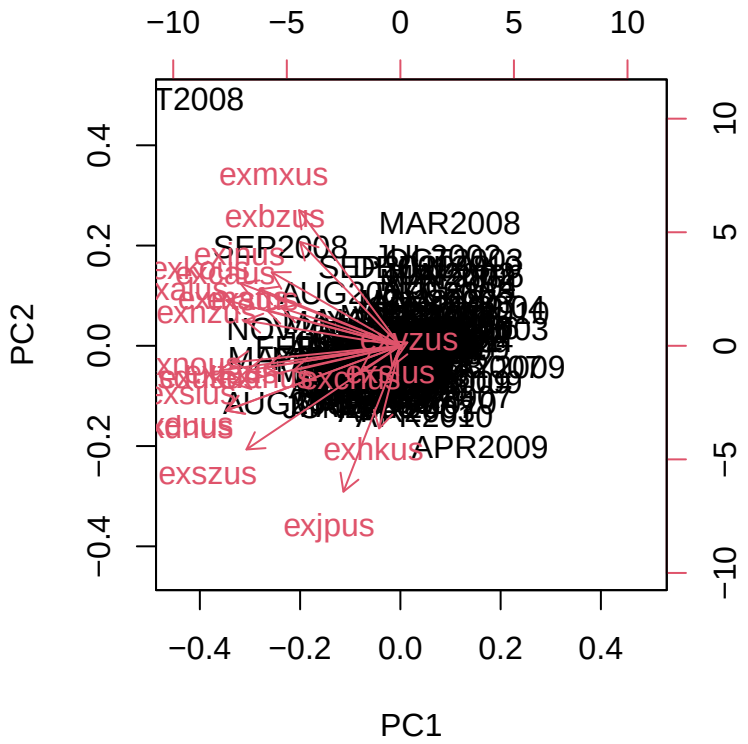
```
plot(pr.out, type="lines")
```




```
summary(pr.out)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.1904 1.5905 1.18680 1.14792 0.99740 0.93815 0.92009
## Proportion of Variance 0.4425 0.1100 0.06124 0.05729 0.04325 0.03827 0.03681
## Cumulative Proportion  0.4425 0.5525 0.61377 0.67107 0.71432 0.75258 0.78939
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.82835 0.80841 0.76390 0.69185 0.65917 0.58024 0.56012
## Proportion of Variance 0.02983 0.02841 0.02537 0.02081 0.01889 0.01464 0.01364
## Cumulative Proportion  0.81923 0.84764 0.87301 0.89382 0.91271 0.92735 0.94099
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.55254 0.50190 0.44624 0.41834 0.38808 0.33724 0.30771
## Proportion of Variance 0.01327 0.01095 0.00866 0.00761 0.00655 0.00494 0.00412
## Cumulative Proportion  0.95427 0.96522 0.97388 0.98149 0.98803 0.99298 0.99709
##                           PC22    PC23
## Standard deviation     0.2580 0.01557
## Proportion of Variance 0.0029 0.00001
## Cumulative Proportion  1.0000 1.00000
```

```
biplot(pr.out)
```



Analyzing the results from the code above, printing (pr.out) shows the standard deviation of all 23 PCs, and their loadings, the coefficients of the linear combinations of the continuous variables. Then, the plot function that I used shows a plot of the variances associated with the PCs. It is useful to decide how many PCs to retain for further analysis: In this case, we can see that the first two PCs explain most of the variability in the data. I then printed a summary to describe the importance of the PCs. Row 1 describes the standard deviation associated with each PC, row 2 describes the proportion of variance in the data explained by each component, and row 3 describes the cumulative proportion of explained variance. In this case, we can see that the first two PCs account for about 55% of the variance of the data and the first three PCs account for about 61% of the variance in the data.

3). Regress SP500 returns onto currency movement PCs, using both "glm on first K" and lasso techniques. Use the results to add to your factor interpretation.

```
sp<-read.csv("sp500.csv")
y<-sp$sp500
x<- data.matrix(pr.out$x)
glm.reg<-glm(y~x)
summary(glm.reg)
```

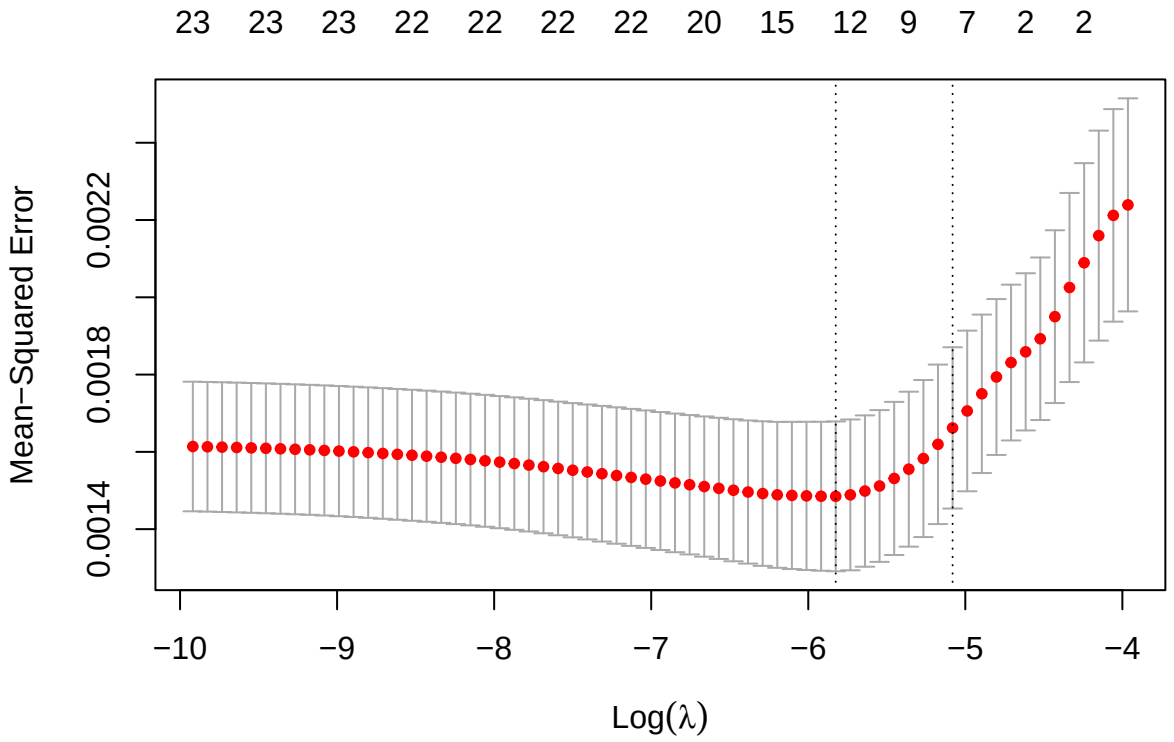
```
##
## Call:
## glm(formula = y ~ x)
##
## Deviance Residuals:
```

```
##       Min         1Q     Median         3Q        Max
## -0.099622  -0.017730   0.003774   0.019975   0.075296
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0004431  0.0032321   0.137  0.89125
## xPC1         0.0059741  0.0010174   5.872 6.30e-08 ***
## xPC2        -0.0111795  0.0020408  -5.478 3.51e-07 ***
## xPC3         0.0082100  0.0027349   3.002  0.00343 **
## xPC4         0.0024263  0.0028275   0.858  0.39300
## xPC5         0.0042391  0.0032542   1.303  0.19584
## xPC6         0.0002007  0.0034598   0.058  0.95387
## xPC7        -0.0012710  0.0035277  -0.360  0.71942
## xPC8        -0.0022731  0.0039183  -0.580  0.56320
## xPC9        -0.0014776  0.0040150  -0.368  0.71367
## xPC10        0.0056542  0.0042490   1.331  0.18646
## xPC11       -0.0020874  0.0046915  -0.445  0.65738
## xPC12        0.0029706  0.0049240   0.603  0.54776
## xPC13        0.0049291  0.0055938   0.881  0.38045
## xPC14        0.0045013  0.0057948   0.777  0.43922
## xPC15       -0.0130368  0.0058742  -2.219  0.02885 *
## xPC16        0.0099206  0.0064670   1.534  0.12834
## xPC17        0.0212105  0.0072736   2.916  0.00442 **
## xPC18        0.0098580  0.0077587   1.271  0.20698
## xPC19       -0.0051704  0.0083638  -0.618  0.53793
## xPC20        0.0279026  0.0096246   2.899  0.00465 **
## xPC21       -0.0208193  0.0105483  -1.974  0.05132 .
## xPC22        0.0061814  0.0125785   0.491  0.62426
## xPC23       -0.5630594  0.2084675  -2.701  0.00819 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.001243133)
##
##     Null deviance: 0.26463  on 118  degrees of freedom
## Residual deviance: 0.11810  on  95  degrees of freedom
## AIC: -435.22
##
## Number of Fisher Scoring iterations: 2
```

```
cv_model <- cv.glmnet(x, y, alpha = 1)
best_lambda <- cv_model$lambda.min
best_lambda
```

```
## [1] 0.002952617
```

```
plot(cv_model)
```




```
best_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)
coef(best_model)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  0.0004430924
## PC1          0.0050447396
## PC2         -0.0093151486
## PC3          0.0057116149
## PC4          .
## PC5          0.0012662751
## PC6          .
## PC7          .
## PC8          .
## PC9          .
## PC10         0.0017726582
## PC11         .
## PC12         .
## PC13         .
## PC14         .
## PC15        -0.0076704673
## PC16         0.0040128248
## PC17         0.0145658872
## PC18         0.0027702351
## PC19         .
```

```
## PC20          0.0191102423
## PC21         -0.0111831618
## PC22          .
## PC23         -0.3726185271
```

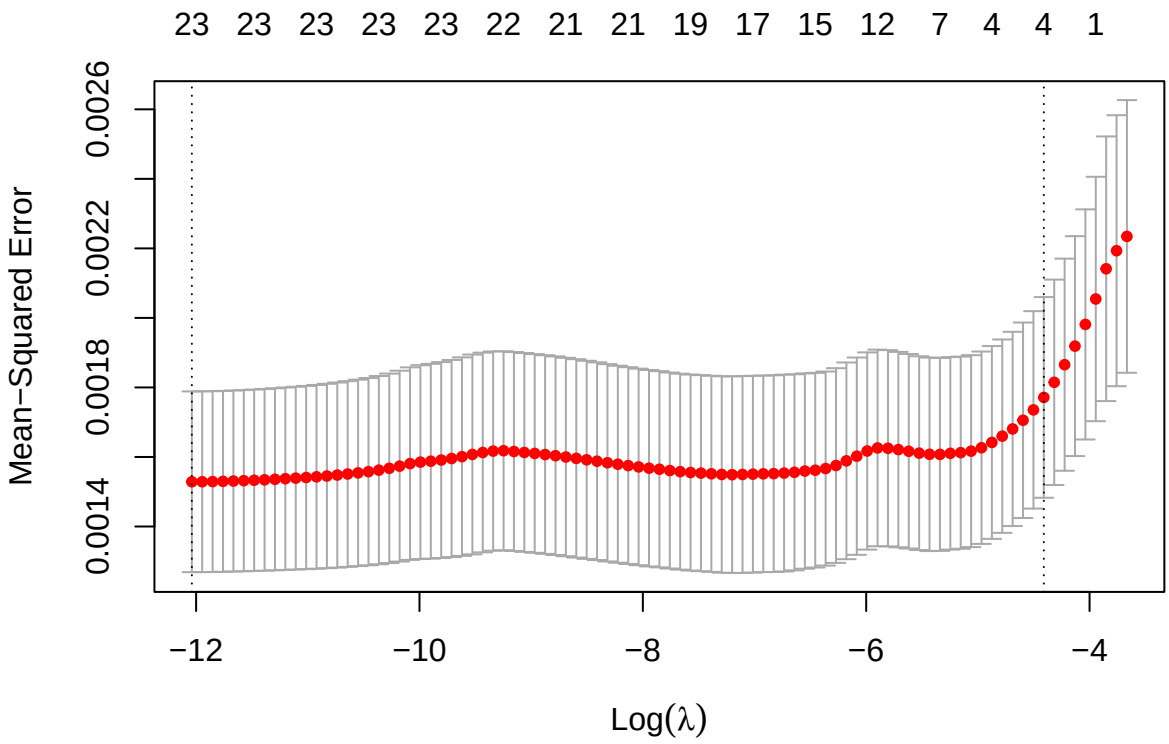
The Lambda for the lasso regression that minimized the test MSE is 0.00168. In the 24x1 sparse Matrix, if the PC has a "." next to it, that means that the lasso regression shrunk the coefficient all the way to zero because it wasn't influential enough. PC 1 and 2 are clearly the most influential again.

4). Fit lasso to the original covariates and describe how it differs from the principal components regression here.

```
x<- data.matrix(fx)
cv_model <- cv.glmnet(x, y, alpha = 1)
best_lambda <- cv_model$lambda.min
best_lambda
```

```
## [1] 5.914108e-06
```

```
plot(cv_model)
```




```
best_model <- glmnet(x, y, alpha = 1, lambda = best_lambda)
coef(best_model)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)   0.006710625
## exalus       -0.475831498
## exbzus       -0.111458351
## excaus        0.213767585
## exchus        1.889991119
## exdnus       13.570494919
## exhkus       -3.446710643
## exinus       -0.046807128
## exjpus        0.224457907
## exkous       -0.086792908
## exmaus       -0.228310659
## exmxus       -0.575033264
## exnzus        0.329710765
## exnous        0.568251756
## exsius        0.979260145
## exsfus       -0.139568564
## exslus        0.149361240
## exsdus       -1.077679063
## exszus       -0.278806675
## extaus       -0.012881891
## exthus       -0.160571985
## exukus        0.360509971
## exvzus       -0.070412226
## exeuus      -13.356607534
```

The original covariates are pretty similarly with a slightly higher best lambda value.