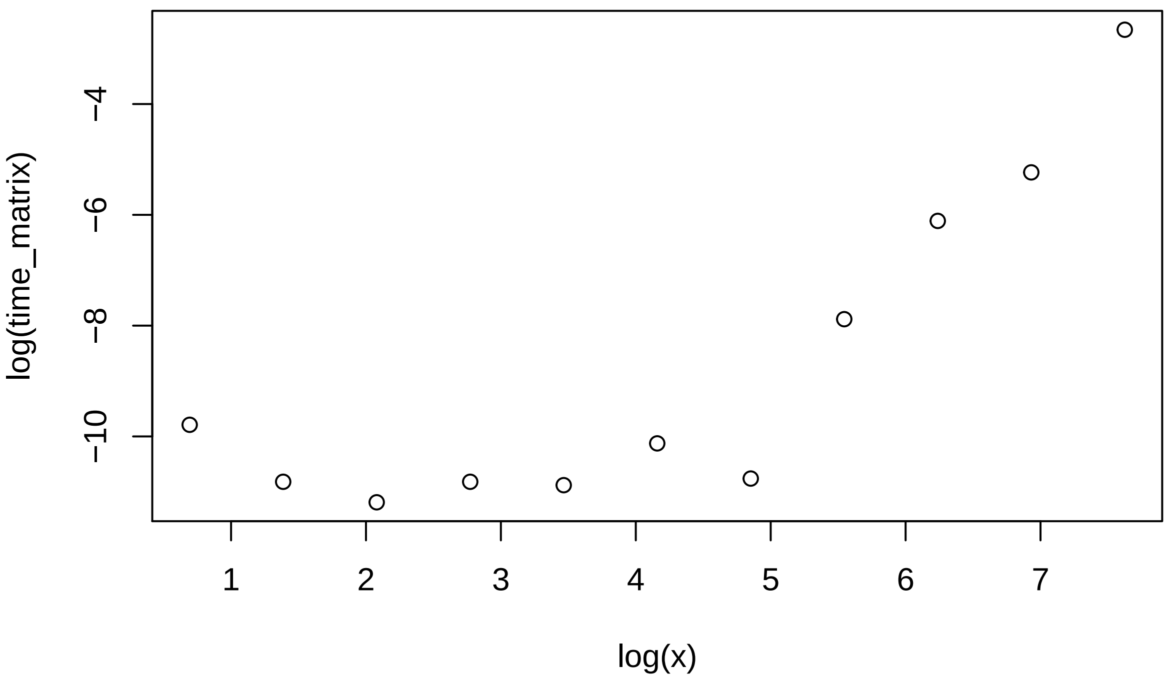
Stat154_hw2

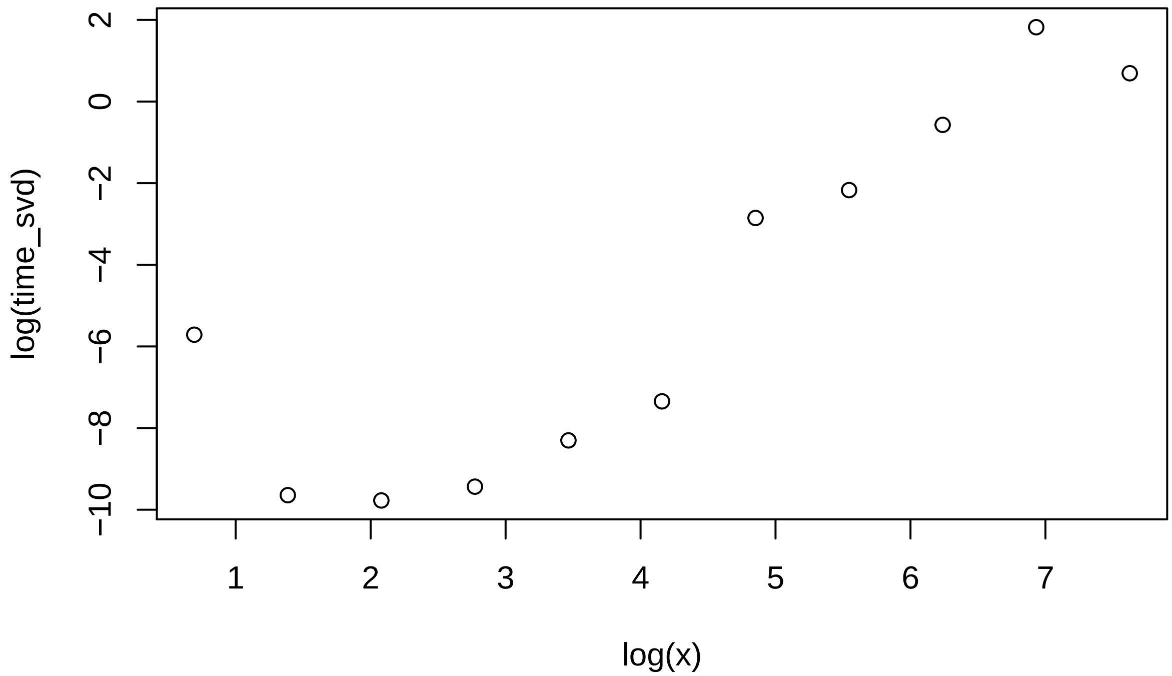
1.A few basics of SVD

(c)

```
x=c(2,4,8,16,32,64,128,256,512,1024,2048)
time_matrix=c()
time_svd=c()
for(i in x){
start1=Sys.time()
M=matrix(rnorm(1),i,i)
end1=Sys.time()
time_matrix=c(time_matrix,(end1-start1))
start2=Sys.time()
svd(M)
end2=Sys.time()
time_svd=c(time_svd,(end2-start2))
}

plot(y=log(time_matrix),x=log(x))
```




```
plot(y=log(time_svd),x=log(x))
```




```
#Observation
#It seems that running time almost stay the same at first, it has the biggest increasing rate when n=4.
#Startig from n=4, running time approximately linearly scaled with n.
#It makes perfect sense, because when n is smaller than 4,
#the biggest matrix we got is 4*4, constructing time would not be so much of a difference,
#and u,v from its svd is not too large dimension. After n reaches 4,
#svd calculation scales fast.Thus running time scales fast.
```

2.1.Power Method

```
A=cbind(c(1,2,3),c(2,-1,4),c(3,4,-5))
w0=c(1,1,1)

powerMethod=function(v,M){
s0=max(abs(M%*%v))
v=M%*%v/s0
s1=max(abs(M%*%v))

while((s1-s0)/s0>0.01){
  s0=max(abs(M%*%v))
  v=M%*%v/s0
  s1=max(abs(M%*%v))
}
return(c(v,s1))
}


#using power method to compute first eigen value
powerMethod(w0,A)
```

```
## [1] 1.0000000 0.8333333 0.3333333 4.6666667
```

```r
eigen(A)
```

```
## eigen() decomposition
## $values
## [1]  4.610843 -1.842654 -7.768189
##
## $vectors
##            [,1]       [,2]       [,3]
## [1,] -0.6890036  0.6985555 -0.1931172
## [2,] -0.5672220 -0.6856083 -0.4562899
## [3,] -0.4511466 -0.2048451  0.8686226
```

```r
#conclusion
#By comparing the two eigen values obtained with power method
#and eigen value functions. The results are pretty close.
#We can manually adjust the threhold of sk/sk+1 measurement
#to make the result more accruate.
#Thus power method is a pretty good way to get the max eigenvalue.
```

2.2.Deflation and more eigenvectors

```r
B=cbind(c(5,1,0),c(1,4,0),c(0,0,1))
#(a)
first=powerMethod(w0,B)
#first eigenvector
first[1:3]
```

```
## [1] 1.0000000 0.8333333 0.1666667
```

```r
#first eigenvalue
first[4]
```

```
## [1] 5.833333
```

```r
#(b)
B1=B-first[4]*first[1:3]%*%t(first[1:3])
second=powerMethod(w0,B1)
#second eigenvector
second[1:3]
```

```
## [1] -1.0000000 -0.8333333 -0.1666667
```

```r
#second eigenvalue
second[4]
```

```
## [1] 4.212963
```

```r
#(c)
B2=B-second[4]*second[1:3]%*%t(second[1:3])
third=powerMethod(w0,B2)
#third eigenvector
third[1:3]
```

```
## [1] -1.0000000 -0.8333333 -0.1666667
```

```
#third eigenvalue
third[4]
```

```
## [1] 1.713049
```

3.PCA

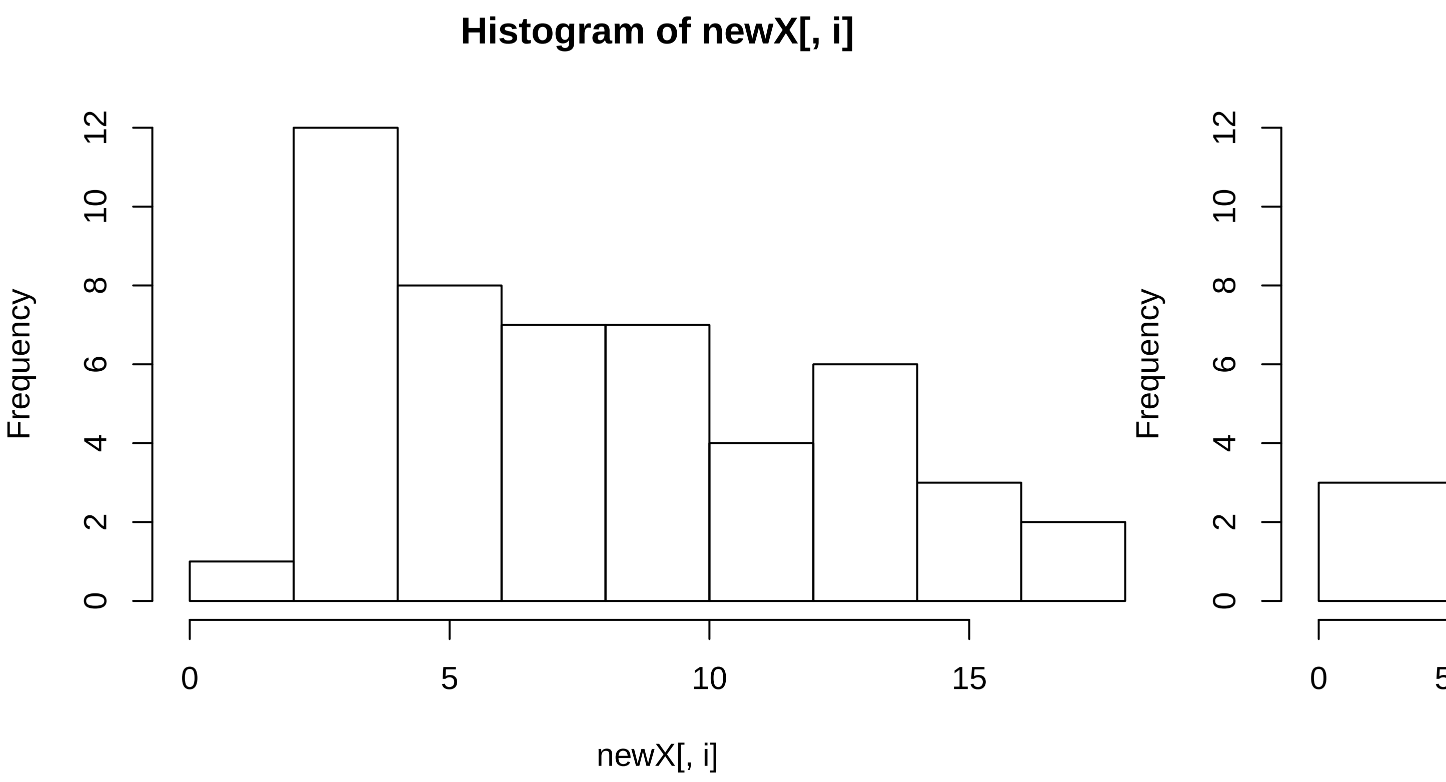
```
dat=USArrests
#(a)mean and variance
apply(dat,2,mean)
```

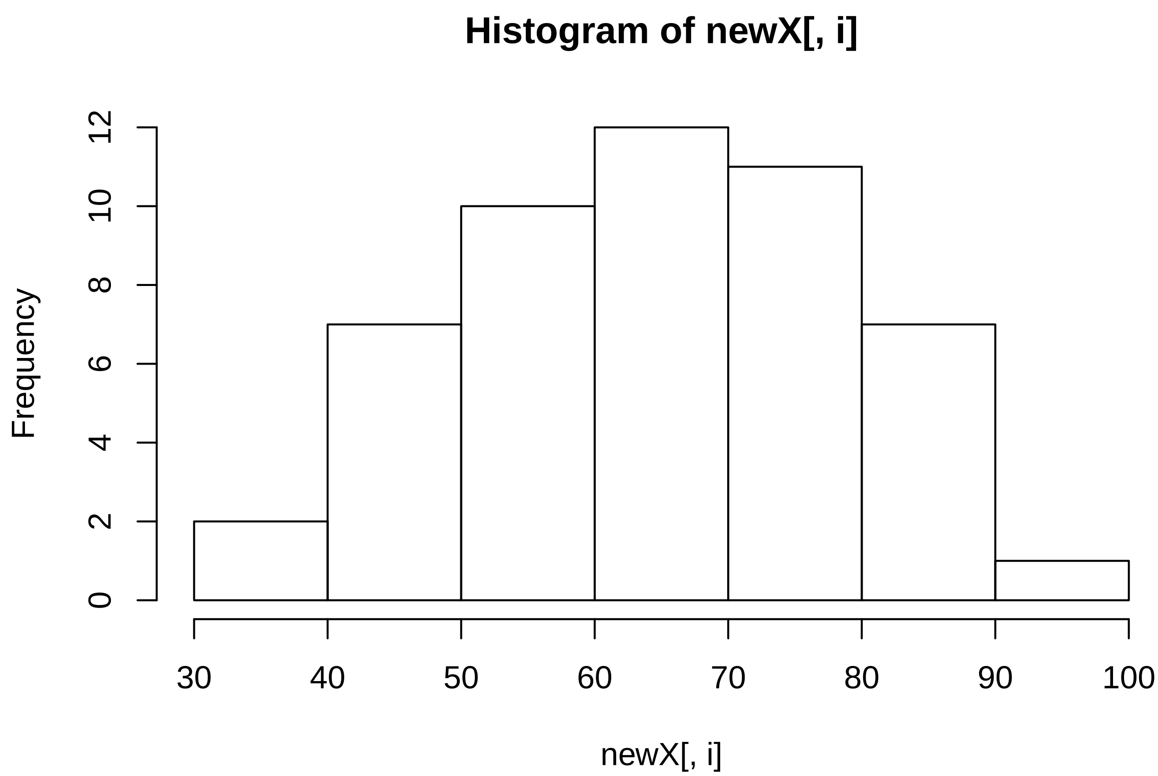
```
##   Murder  Assault UrbanPop     Rape
##    7.788  170.760   65.540   21.232
```

```
apply(dat,2,var)
```

```
##     Murder    Assault   UrbanPop       Rape
##   18.97047 6945.16571  209.51878   87.72916
```

```
#(b)histogram
apply(dat,2,hist)
```






```
## $Murder
## $breaks
##  [1]  0  2  4  6  8 10 12 14 16 18
##
## $counts
## [1]  1 12  8  7  7  4  6  3  2
##
## $density
## [1] 0.01 0.12 0.08 0.07 0.07 0.04 0.06 0.03 0.02
##
## $mids
## [1]  1  3  5  7  9 11 13 15 17
##
## $xname
## [1] "newX[, i]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## $Assault
## $breaks
## [1]   0  50 100 150 200 250 300 350
##
## $counts
## [1]  3  7 12  9  7 10  2
##
```

```
## $density
## [1] 0.0012 0.0028 0.0048 0.0036 0.0028 0.0040 0.0008
##
## $mids
## [1]  25  75 125 175 225 275 325
##
## $xname
## [1] "newX[, i]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## $UrbanPop
## $breaks
## [1]  30  40  50  60  70  80  90 100
##
## $counts
## [1]  2  7 10 12 11  7  1
##
## $density
## [1] 0.004 0.014 0.020 0.024 0.022 0.014 0.002
##
## $mids
## [1] 35 45 55 65 75 85 95
##
## $xname
## [1] "newX[, i]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
##
## $Rape
## $breaks
##  [1]  5 10 15 20 25 30 35 40 45 50
##
## $counts
## [1]  5  8 12  9  8  3  2  2  1
##
## $density
## [1] 0.020 0.032 0.048 0.036 0.032 0.012 0.008 0.008 0.004
##
## $mids
## [1]  7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5 47.5
##
## $xname
## [1] "newX[, i]"
##
## $equidist
```

```
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

```r
#(c)correlation
cor.test(dat$Murder,dat$Assault)
```

```
##
##  Pearson's product-moment correlation
##
## data:  dat$Murder and dat$Assault
## t = 9.2981, df = 48, p-value = 2.596e-12
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6739512 0.8831110
## sample estimates:
##       cor
## 0.8018733
```

```r
cor.test(dat$Rape,dat$Assault)
```

```
##
##  Pearson's product-moment correlation
##
## data:  dat$Rape and dat$Assault
## t = 6.173, df = 48, p-value = 1.364e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4748141 0.7961645
## sample estimates:
##       cor
## 0.6652412
```

```r
cor.test(dat$Rape,dat$Murder)
```

```
##
##  Pearson's product-moment correlation
##
## data:  dat$Rape and dat$Murder
## t = 4.7267, df = 48, p-value = 2.031e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3383006 0.7277619
## sample estimates:
##       cor
## 0.5635788
```

```r
#We can see from the pearson correlation test,
#that these 3 criminal types are somehow correlated,
#especially murder and assualt, correlation as high as 0.8018.

#(d)
pca1=princomp(dat,cor=TRUE)
summary(prcomp(dat))
```

```
## Importance of components%s:
```

```
##                            PC1      PC2    PC3     PC4
## Standard deviation     83.7324 14.21240 6.4894 2.48279
## Proportion of Variance  0.9655  0.02782 0.0058 0.00085
## Cumulative Proportion   0.9655  0.99335 0.9991 1.00000
```

```r
#(e)
pca1$loadings[,1:3]
```

```
##              Comp.1     Comp.2     Comp.3
## Murder   -0.5358995  0.4181809 -0.3412327
## Assault  -0.5831836  0.1879856 -0.2681484
## UrbanPop -0.2781909 -0.8728062 -0.3780158
## Rape     -0.5434321 -0.1673186  0.8177779
```

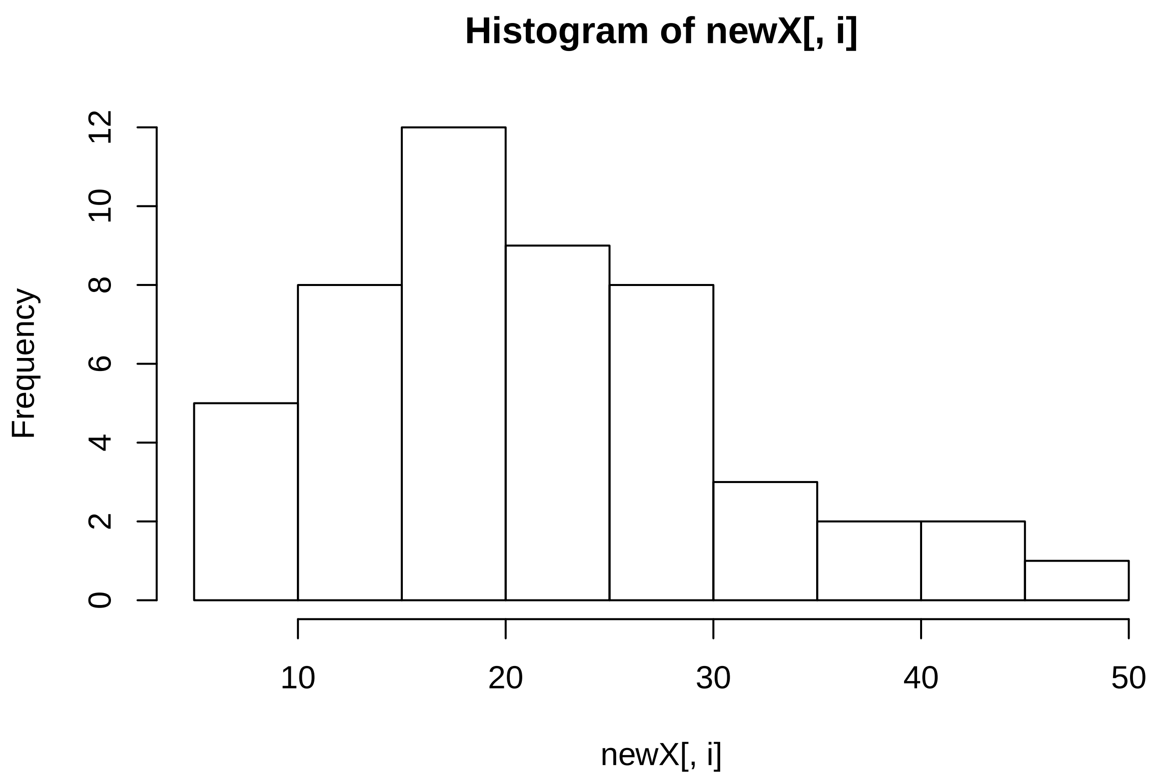
```r
#(f)PCs aka Scores
head(pca1$scores[,1:3])
```

```
##                Comp.1     Comp.2      Comp.3
## Alabama    -0.9855659  1.1333924 -0.44426879
## Alaska     -1.9501378  1.0732133  2.04000333
## Arizona    -1.7631635 -0.7459568  0.05478082
## Arkansas    0.1414203  1.1197968  0.11457369
## California -2.5239801 -1.5429340  0.59855680
## Colorado   -1.5145629 -0.9875551  1.09500699
```

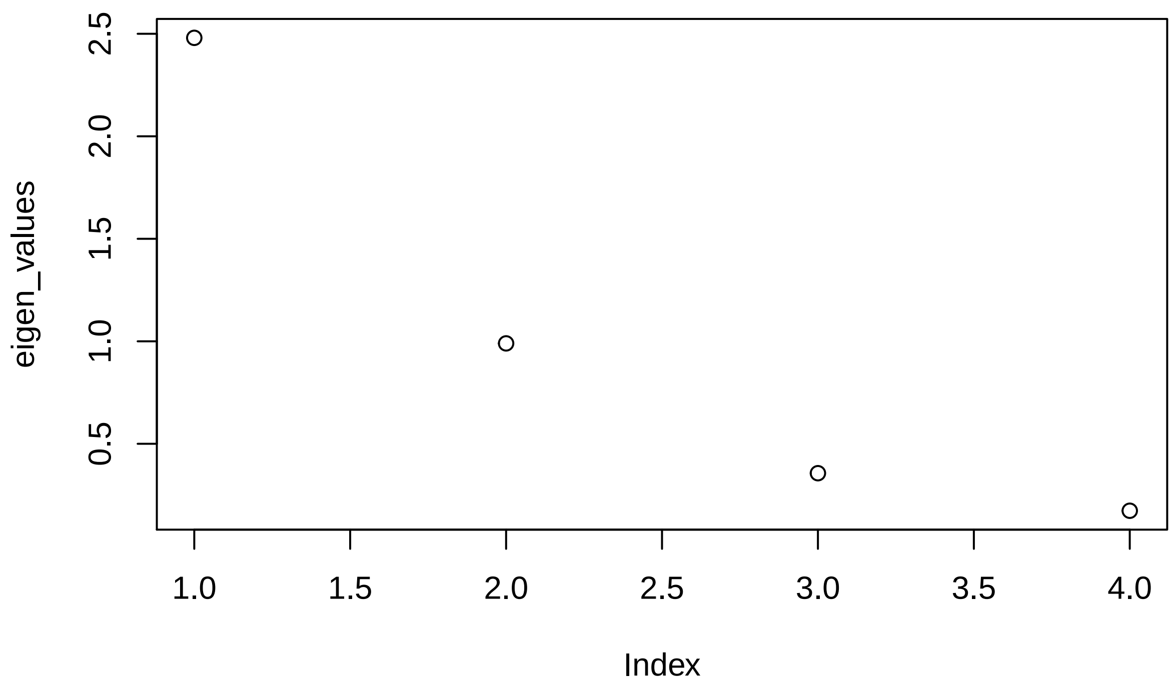
```r
#(g)eigen_values and sum
eigen_values=pca1$sdev^2
sum(eigen_values)
```

```
## [1] 4
```

```r
#(h)
library(ggplot2)
```

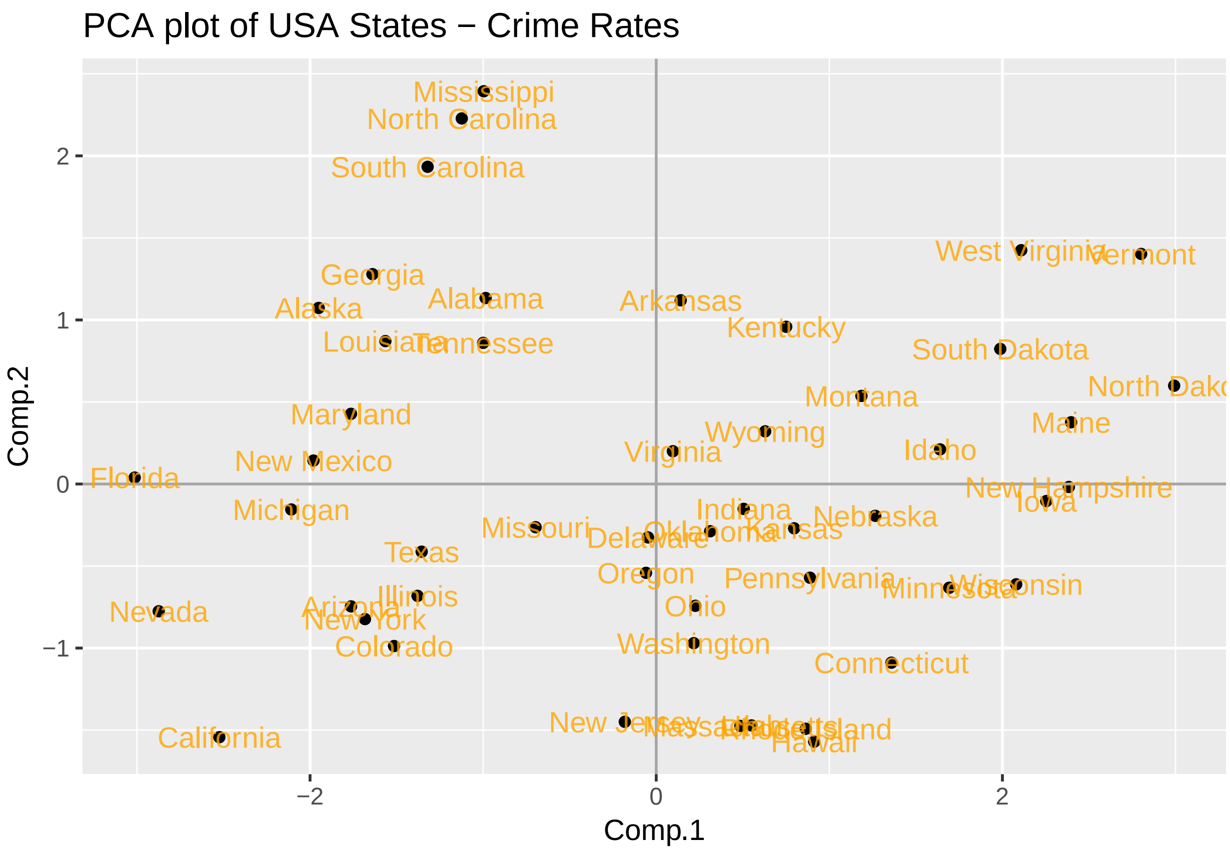



```r
plot(eigen_values)
```

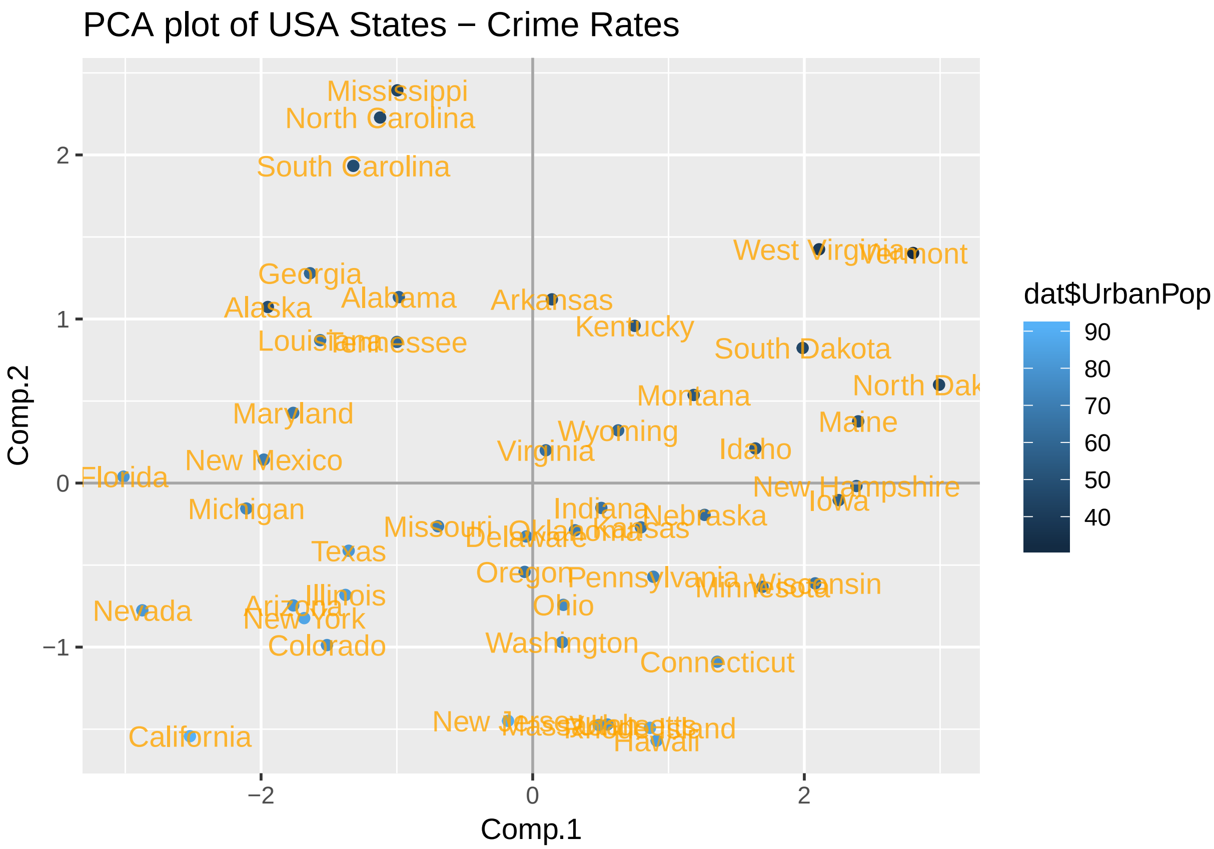



```r
#The eigen values descend from 2.5 to 0.Eigen valeus descend,
#which means that PCs are ordered in an descending order,
#aka, the first pc captures the most variability.

#(i)
scores=as.data.frame(pca1$scores[,c(1,2)])
```

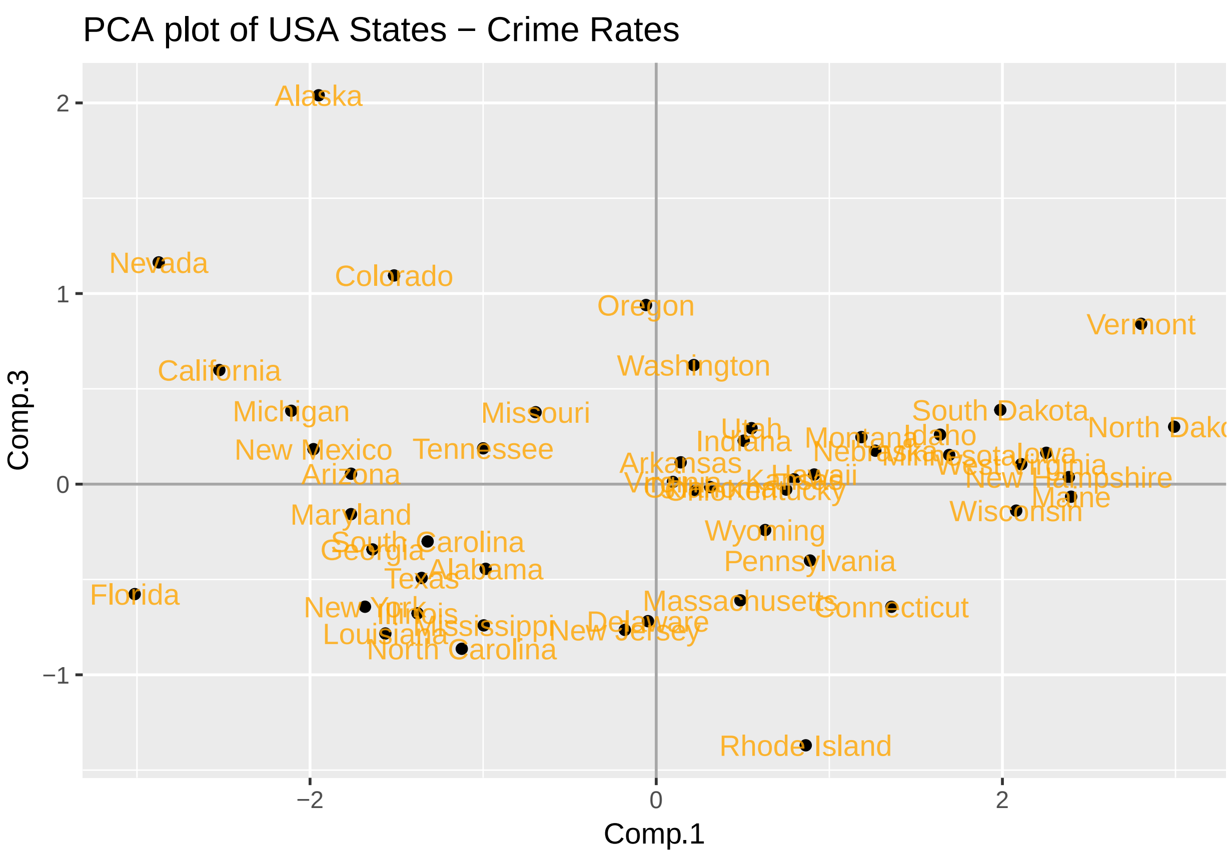
```r
ggplot(data=scores,aes(x=Comp.1,y=Comp.2,label=rownames(scores)))+
  geom_point()+
  geom_hline(yintercept = 0, colour = "gray65") +
  geom_vline(xintercept = 0, colour = "gray65") +
  geom_text(colour = "orange", alpha = 0.8, size = 4) +
  ggtitle("PCA plot of USA States - Crime Rates")
```




```r
#which state stands out?
#Mississippi stands out. Without PCA, it is hard to say
#which state stands out regarding crime rate.
#However, under PCA(2-dim), Mississipi seems to stand out
#to be have the highest overall crime rate.


#(j)color states according to UrbanPop
scores=as.data.frame(pca1$scores[,c(1,2)])
scores=cbind(scores,dat$UrbanPop)
ggplot(data=scores,aes(x=Comp.1,y=Comp.2,label=rownames(scores)))+
  geom_point(aes(color=dat$UrbanPop))+
  geom_hline(yintercept = 0, colour = "gray65") +
  geom_vline(xintercept = 0, colour = "gray65") +
  geom_text(colour = "orange", alpha = 0.8, size = 4) +
  ggtitle("PCA plot of USA States - Crime Rates")
```




```r
#(k)
scores=as.data.frame(pca1$scores[,c(1,3)])
ggplot(data=scores,aes(x=Comp.1,y=Comp.3,label=rownames(scores)))+
  geom_point()+
  geom_hline(yintercept = 0, colour = "gray65") +
  geom_vline(xintercept = 0, colour = "gray65") +
  geom_text(colour = "orange", alpha = 0.8, size = 4) +
  ggtitle("PCA plot of USA States - Crime Rates")
```




```
#PC3 has lower variance capturebility, pc2 is better as a PC dimension
#compared to PC3 to capture variability.
#In the plot, we observe that y-range is narrower in this plot.
#Accordingly, the stand-out state has changes under different PC,
#Alaska stands out in this plot.
```

4.K-means and PCA

10.7

```
#(a)generate 3 classes students as our simulation data.
class1=rnorm(1000,20,1) #n,mean,sd
class2=rnorm(1000,60,1)
class3=rnorm(1000,90,1)

classes=as.data.frame(matrix(data=c(class1,class2,class3),nrow=60,ncol=50),byrow=T)
group=c(rep(1,20),rep(2,20),rep(3,20))
classes=cbind(classes,group)

#(b)
pca2=princomp(classes,cor=TRUE)
summary(pca2)
```

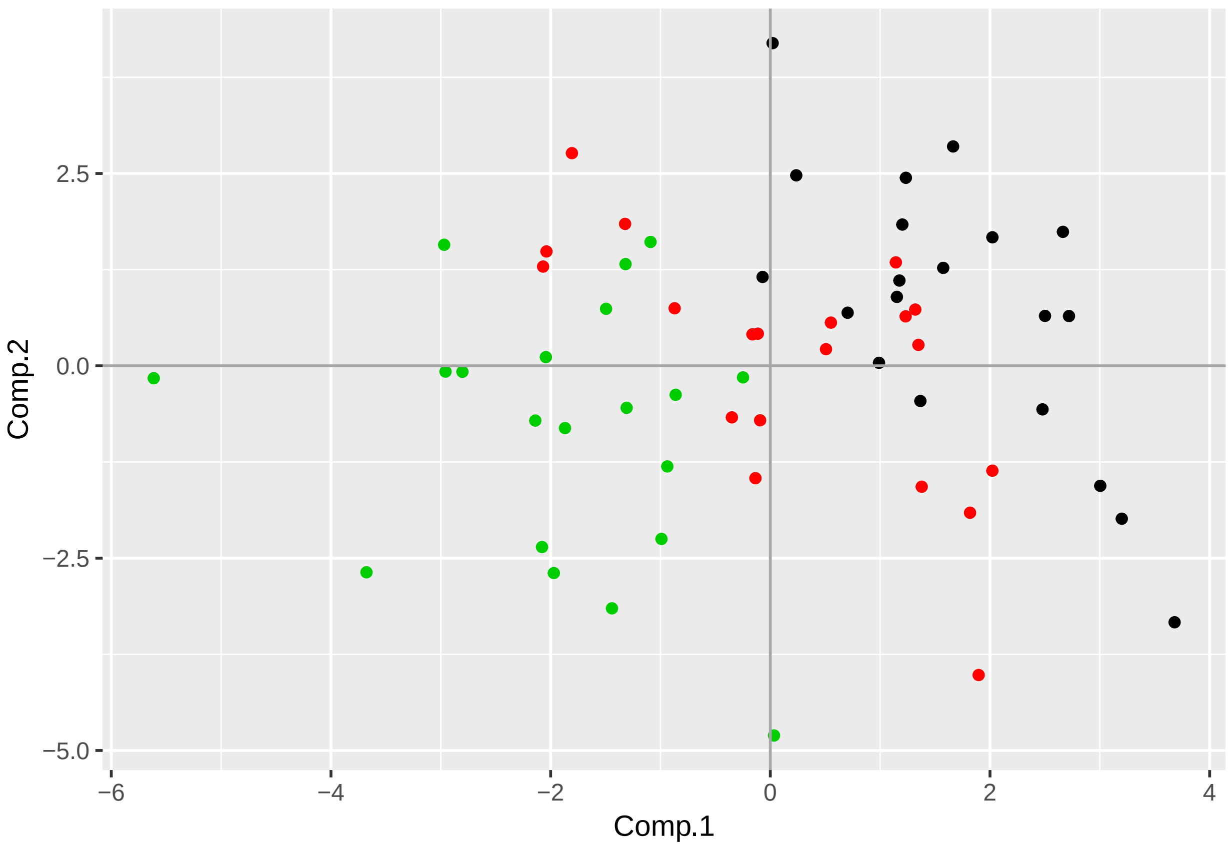
```
## Importance of components:
```

```
##                            Comp.1    Comp.2     Comp.3     Comp.4
## Standard deviation     1.89240382 1.7624403 1.74187275 1.62116629
## Proportion of Variance 0.07021945 0.0609058 0.05949256 0.05153294
## Cumulative Proportion  0.07021945 0.1311253 0.19061782 0.24215076
##                            Comp.5     Comp.6     Comp.7    Comp.8
## Standard deviation     1.59805354 1.55817587 1.49262796 1.4208607
## Proportion of Variance 0.05007402 0.04760612 0.04368506 0.0395852
## Cumulative Proportion  0.29222478 0.33983090 0.38351596 0.4231012
##                            Comp.9    Comp.10    Comp.11    Comp.12
## Standard deviation     1.40892200 1.35450598 1.34114887 1.31746474
## Proportion of Variance 0.03892277 0.03597424 0.03526824 0.03403359
## Cumulative Proportion  0.46202393 0.49799817 0.53326642 0.56730001
##                           Comp.13    Comp.14    Comp.15    Comp.16
## Standard deviation     1.28463712 1.24440102 1.23538366 1.19252071
## Proportion of Variance 0.03235868 0.03036341 0.02992496 0.02788442
## Cumulative Proportion  0.59965869 0.63002210 0.65994705 0.68783148
##                           Comp.17    Comp.18    Comp.19    Comp.20
## Standard deviation     1.16774462 1.10561341 1.07813430 1.02583987
## Proportion of Variance 0.02673779 0.02396826 0.02279164 0.02063426
## Cumulative Proportion  0.71456927 0.73853753 0.76132917 0.78196343
##                           Comp.21    Comp.22    Comp.23    Comp.24
## Standard deviation     1.00408316 0.97087856 0.90827187 0.90384055
## Proportion of Variance 0.01976829 0.01848245 0.01617564 0.01601819
## Cumulative Proportion  0.80173172 0.82021418 0.83638982 0.85240801
##                           Comp.25    Comp.26    Comp.27    Comp.28
## Standard deviation     0.87404660 0.84688459 0.80042425 0.76278836
## Proportion of Variance 0.01497956 0.01406301 0.01256233 0.01140875
## Cumulative Proportion  0.86738757 0.88145058 0.89401291 0.90542166
##                           Comp.29     Comp.30     Comp.31     Comp.32
## Standard deviation     0.71587117 0.699224128 0.686378514 0.647732963
## Proportion of Variance 0.01004846 0.009586556 0.009237558 0.008226627
## Cumulative Proportion  0.91547012 0.925056678 0.934294236 0.942520863
##                            Comp.33     Comp.34     Comp.35     Comp.36
## Standard deviation     0.632511225 0.608504781 0.564327566 0.546759461
## Proportion of Variance 0.007844519 0.007260354 0.006244424 0.005861684
## Cumulative Proportion  0.950365382 0.957625736 0.963870160 0.969731844
##                            Comp.37     Comp.38     Comp.39    Comp.40
## Standard deviation     0.478391428 0.452647272 0.435017962 0.41667830
## Proportion of Variance 0.004487419 0.004017442 0.003710601 0.00340433
## Cumulative Proportion  0.974219263 0.978236705 0.981947306 0.98535164
##                            Comp.41     Comp.42     Comp.43     Comp.44
## Standard deviation     0.396180338 0.390527551 0.344875366 0.273530807
## Proportion of Variance 0.003077625 0.002990427 0.002332138 0.001467041
## Cumulative Proportion  0.988429260 0.991419687 0.993751825 0.995218866
##                            Comp.45     Comp.46      Comp.47      Comp.48
## Standard deviation     0.269218208 0.236228866 0.2136709135 0.1907902518
## Proportion of Variance 0.001421146 0.001094198 0.0008952012 0.0007137435
## Cumulative Proportion  0.996640012 0.997734209 0.9986294104 0.9993431540
##                             Comp.49      Comp.50      Comp.51
## Standard deviation     0.1464944476 0.1085102621 1.624952e-02
## Proportion of Variance 0.0004207965 0.0002308721 5.177392e-06
## Cumulative Proportion  0.9997639505 0.9999948226 1.000000e+00
```

```r
head(pca2$scores[,1:2])
```

```
##         Comp.1    Comp.2
## [1,] 1.5745167 1.2726008
## [2,] 1.6644289 2.8510161
## [3,] 1.2342414 2.4437835
## [4,] 2.0223267 1.6701626
## [5,] 0.2363056 2.4755536
## [6,] 2.7192484 0.6470889
```

```r
scores2=as.data.frame(pca2$scores[,c(1,2)])

ggplot(data=scores2,aes(x=Comp.1,y=Comp.2))+
  geom_point(color=group)+
  geom_hline(yintercept = 0, colour = "gray65") +
  geom_vline(xintercept = 0, colour = "gray65")
```




```r
#Plot shows that there is a visible boundary among three
#clusters, so we continue to conduct kmeans clustering.

#(c)
kmeans(classes,centers=3)
```

```
## K-means clustering with 3 clusters of sizes 20, 20, 20
##
## Cluster means:
##         V1       V2       V3       V4       V5       V6       V7       V8
## 1 19.97240 19.72686 19.97423 20.40341 20.16614 20.27625 19.92674 20.04647
```

```
## 2 20.01819 20.35797 20.11227 19.75259 20.04011 19.39495 20.30645 20.25014
## 3 19.52928 20.18172 20.13095 19.75002 20.02027 19.41301 19.96914 20.02710
##         V9      V10      V11      V12      V13      V14      V15      V16
## 1 20.02338 19.92812 20.00596 20.37454 19.98881 20.01206 20.09069 20.28545
## 2 20.19394 19.79543 20.10252 19.75600 20.27584 19.92263 20.45507 19.90699
## 3 19.71645 20.26300 20.22720 20.01613 20.34132 19.57202 20.14760 20.26945
##        V17      V18      V19      V20      V21      V22      V23      V24
## 1 19.61010 59.59448 60.23318 60.08840 59.79531 60.15502 60.15749 60.14714
## 2 59.73904 60.11128 59.81648 59.98027 60.04586 59.81626 60.02962 60.31988
## 3 20.05247 60.05471 59.68371 60.03961 60.01844 59.80895 60.00245 59.90374
##        V25      V26      V27      V28      V29      V30      V31      V32
## 1 60.18633 59.84731 60.24165 60.03256 59.99404 60.05940 60.17832 59.90069
## 2 59.95927 60.17719 59.95640 59.84882 60.08103 59.69276 59.89471 59.82640
## 3 59.97358 60.15442 60.25507 59.91941 60.04478 60.37201 60.06027 59.67587
##        V33      V34      V35      V36      V37      V38      V39      V40
## 1 59.75988 89.88497 89.44370 89.92288 89.75545 90.01990 89.94869 90.18793
## 2 59.50958 89.61202 89.92403 89.84345 89.81362 89.88088 89.69310 89.61029
## 3 59.83328 60.01865 90.14942 89.88886 90.18902 90.16421 89.71413 89.98663
##        V41      V42      V43      V44      V45      V46      V47      V48
## 1 90.04913 90.11911 89.84545 89.76691 89.94975 89.79951 90.06895 90.08354
## 2 89.73371 90.00322 90.43532 89.38071 89.96887 90.18259 90.51738 89.96952
## 3 90.41274 89.89578 90.05694 89.97381 90.33129 89.90631 90.26019 90.13549
##        V49      V50 group
## 1 90.03556 89.79590     2
## 2 89.76002 89.91204     3
## 3 89.77739 89.78042     1
##
## Clustering vector:
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 981.8135 964.4516 976.9757
##  (between_SS / total_SS =  91.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
#conclusion:kmeans clustering with 3 centers on the raw
#data does not do very good. Data are clustered into group of 9,11,40.
#Comparing to true lable of group 20,20,20.


#(d)
kmeans(classes,centers=2)
```

```
## K-means clustering with 2 clusters of sizes 20, 40
##
## Cluster means:
##         V1       V2       V3       V4       V5       V6       V7       V8
## 1 19.97240 19.72686 19.97423 20.40341 20.16614 20.27625 19.92674 20.04647
## 2 19.77373 20.26985 20.12161 19.75130 20.03019 19.40398 20.13780 20.13862
```

```
##          V9      V10      V11      V12      V13      V14      V15      V16
## 1 20.02338 19.92812 20.00596 20.37454 19.98881 20.01206 20.09069 20.28545
## 2 19.95520 20.02921 20.16486 19.88606 20.30858 19.74733 20.30133 20.08822
##         V17      V18      V19      V20      V21      V22      V23      V24
## 1 19.61010 59.59448 60.23318 60.08840 59.79531 60.15502 60.15749 60.14714
## 2 39.89576 60.08300 59.75010 60.00994 60.03215 59.81260 60.01604 60.11181
##         V25      V26      V27      V28      V29      V30      V31      V32
## 1 60.18633 59.84731 60.24165 60.03256 59.99404 60.05940 60.17832 59.90069
## 2 59.96642 60.16580 60.10574 59.88411 60.06291 60.03238 59.97749 59.75113
##         V33      V34      V35      V36      V37      V38      V39      V40
## 1 59.75988 89.88497 89.44370 89.92288 89.75545 90.01990 89.94869 90.18793
## 2 59.67143 74.81534 90.03672 89.86616 90.00132 90.02254 89.70361 89.79846
##         V41      V42      V43      V44      V45      V46      V47      V48
## 1 90.04913 90.11911 89.84545 89.76691 89.94975 89.79951 90.06895 90.08354
## 2 90.07322 89.94950 90.24613 89.67726 90.15008 90.04445 90.38878 90.05251
##         V49      V50 group
## 1 90.03556 89.79590     2
## 2 89.76871 89.84623     2
##
## Clustering vector:
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1]   981.8135 26528.6636
##  (between_SS / total_SS =  23.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
#conclusion:kmeans clustering with 2 centers on the raw
#data does not do very good. Data are clustered into group of #20,40.
#Comparing to true lable of group 20,20,20.


#(e)
kmeans(classes,centers=4)
```

```
## K-means clustering with 4 clusters of sizes 8, 7, 5, 40
##
## Cluster means:
##          V1       V2       V3       V4       V5       V6       V7       V8
## 1 19.56212 19.94781 20.10848 18.96380 20.00450 19.27734 19.84533 19.39840
## 2 19.08233 20.28617 20.25404 20.80219 19.76759 19.88471 20.98539 20.87715
## 3 20.10245 20.40975 19.99458 19.53492 20.39926 18.96969 18.74449 19.84296
## 4 19.99529 20.04241 20.04325 20.07800 20.10313 19.83560 20.11660 20.14831
##          V9      V10      V11      V12      V13      V14      V15      V16
## 1 20.38276 19.91877 20.34906 20.19055 20.13178 19.16179 21.11662 20.00441
## 2 19.29963 20.19996 19.90726 20.32656 20.65321 19.56774 19.37797 20.07231
## 3 19.23390 20.90203 20.48016 19.30245 20.23996 20.23438 19.67465 20.96950
## 4 20.10866 19.86177 20.05424 20.06527 20.13232 19.96735 20.27288 20.09622
##         V17      V18      V19      V20      V21      V22      V23      V24
```

```
## 1 19.78155 60.33977 59.44847 60.51975 60.20762 59.97149 60.49209 59.94714
## 2 19.70142 60.51355 60.19035 59.21461 59.61131 59.97599 60.04979 59.65091
## 3 20.97739 58.95622 59.35079 60.42640 60.28575 59.31503 59.15274 60.18826
## 4 39.67457 59.85288 60.02483 60.03434 59.92059 59.98564 60.09356 60.23351
##        V25      V26      V27      V28      V29      V30      V31      V32
## 1 60.26494 60.05207 60.26413 59.82642 60.07497 60.74504 60.37066 59.71442
## 2 59.42675 60.48676 60.37637 60.32674 60.02320 60.55996 59.93232 59.20039
## 3 60.27295 59.85290 60.07076 59.49793 60.02667 59.51201 59.74276 60.27987
## 4 60.07280 60.01225 60.09903 59.94069 60.03754 59.87608 60.03652 59.86354
##        V33      V34      V35      V36      V37      V38      V39      V40
## 1 59.66822 59.62723 90.19308 89.79843 91.06421 90.42217 89.60359 89.59578
## 2 59.44802 60.72279 90.33206 89.91867 90.01080 90.21603 90.13034 90.27468
## 3 60.63674 59.65913 89.82385 89.99183 89.03825 89.67892 89.30829 90.20872
## 4 59.63473 89.74849 89.68387 89.88317 89.78453 89.95039 89.82089 89.89911
##        V41      V42      V43      V44      V45      V46      V47      V48
## 1 89.91380 90.28064 89.71629 90.09695 90.42620 89.65995 89.62624 90.27249
## 2 90.25672 89.92131 90.09554 89.73747 90.68182 90.26345 90.45276 89.78519
## 3 91.42946 89.24425 90.54796 90.10767 89.68868 89.80047 91.00490 90.40670
## 4 89.89142 90.06117 90.14038 89.57381 89.95931 89.99105 90.29316 90.02653
##        V49      V50 group
## 1 89.58139 90.24704   1.0
## 2 90.21372 89.61369   1.0
## 3 89.48012 89.26726   1.0
## 4 89.89779 89.85397   2.5
##
## Clustering vector:
##  [1] 2 1 1 1 1 2 3 1 3 2 3 2 2 1 2 1 1 3 3 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [36] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##
## Within cluster sum of squares by cluster:
## [1]   338.1934   260.2919   189.1158 18116.2135
##  (between_SS / total_SS =  47.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
#conclusion:kmeans clustering with 4 centers on the raw data
#does not do very good.
#Data are clustered into group of 11,9,20,20.
#Comparing to true lable of group 20,20,20.
#This clusering slightly improves the result, just that more detailed seperation is captured.


#(f)
kmeans(pca2$scores[,1:2],centers=3)
```

```
## K-means clustering with 3 clusters of sizes 9, 21, 30
##
## Cluster means:
##       Comp.1     Comp.2
## 1  2.1681472 -2.3458518
## 2 -1.8971563 -0.7062883
```

```
## 3  0.6775653  1.1981573
##
## Clustering vector:
##  [1] 3 3 3 3 3 3 1 3 3 3 1 3 3 1 3 3 3 3 1 3 1 3 3 3 3 3 1 2 1 2 1 3 3 3 2
## [36] 3 2 2 3 3 2 3 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 25.03110 70.38924 71.08876
##  (between_SS / total_SS =  58.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
#conclusion:kmeans clustering with 3 centers on the raw data
#does not do very good.
#Data are clustered into group of 20,20,20.
#Comparing to true lable of group 20,20,20.
#This clusering improves the result obviously.
#Thus conducting PCA on raw data is a good data preparation step before doing k-means clustering.

#(g)
classes=scale(classes,scale=FALSE)
kmeans(classes,centers=3)
```

```
## K-means clustering with 3 clusters of sizes 20, 20, 20
##
## Cluster means:
##           V1          V2          V3         V4          V5         V6
## 1 -0.3106779  0.09287121  0.05846669 -0.2186527 -0.05523840 -0.2817279
## 2  0.1324483 -0.36199322 -0.09825346  0.4347386  0.09063568  0.5815171
## 3  0.1782296  0.26912201  0.03978677 -0.2160860 -0.03539728 -0.2997892
##            V7          V8          V9         V10          V11         V12
## 1 -0.09830424 -0.08080226 -0.26147587  0.26748553  0.115309188 -0.03275754
## 2 -0.14070531 -0.06143541  0.04545665 -0.06739588 -0.105933241  0.32564879
## 3  0.23900955  0.14223767  0.21601922 -0.20008965 -0.009375947 -0.29289125
##           V13         V14         V15        V16       V17        V18
## 1  0.13933443 -0.26355004 -0.08351886  0.1154844 -13.08140  0.1345516
## 2 -0.21318405  0.17648645 -0.14042898  0.1314904 -13.52377 -0.3256786
## 3  0.07384962  0.08706359  0.22394784 -0.2469749  26.60518  0.1911270
##           V19          V20         V21        V22         V23         V24
## 1 -0.22741437  0.003518543  0.06523788 -0.1177908 -0.06074092 -0.21984543
## 2  0.32205465  0.052308333 -0.15789477  0.2282767  0.09430418  0.02355246
## 3 -0.09464028 -0.055826876  0.09265689 -0.1104859 -0.03356326  0.19629298
##           V25         V26         V27         V28          V29         V30
## 1 -0.06614528  0.09478157  0.10402963 -0.01418606  0.004827075  0.33061853
## 2  0.14660511 -0.21233149  0.09060813  0.09896742 -0.045909706  0.01801024
## 3 -0.08045983  0.11754992 -0.19463776 -0.08478136  0.041082631 -0.34862877
##           V31         V32         V33        V34        V35          V36
## 1  0.01583252 -0.12511602  0.13236642 -19.819893  0.3103659  0.003798904
## 2  0.13389053  0.09970160  0.05896957  10.046420 -0.3953485  0.037813755
## 3 -0.14972305  0.02541442 -0.19133599   9.773473  0.0849826 -0.041612659
##          V37         V38         V39         V40         V41         V42
```

```
## 1  0.2696606  0.142544002 -0.07117465  0.05834985  0.34754229 -0.110261642
## 2 -0.1639172 -0.001760122  0.16338347  0.25964424 -0.01606217  0.113077603
## 3 -0.1057434 -0.140783880 -0.09220882 -0.31799409 -0.33148012 -0.002815961
##           V43         V44        V45         V46         V47         V48
## 1 -0.05562541  0.26666790  0.2479859 -0.05649671 -0.02198147  0.07263781
## 2 -0.26712270  0.05976912 -0.1335521 -0.16329093 -0.21322356  0.02068774
## 3  0.32274812 -0.32643702 -0.1144338  0.21978764  0.23520502 -0.09332555
##           V49         V50 group
## 1 -0.08026974 -0.04903391    -1
## 2  0.17790282 -0.03355384     0
## 3 -0.09763308  0.08258775     1
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##
## Within cluster sum of squares by cluster:
## [1] 976.9757 981.8135 964.4516
##  (between_SS / total_SS =  91.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
#conclusion: stabalize variable variance helps a lot
#on k-means clustering results( with raw data).
#It could be an alternative way of doing PCA as
#a data pre-processing step before doing k-means clustering.
```

5.T/F

(a)T. We calculate transpose(X)*X, in such way that #eigenvalues are always non-negative.

(b)T. The purpose of PCA is to find orthogonal basis,

and to lower dimensions until there is a

balance between dimension and variability.

(c)F. M has to be symmetric and has non-negative eigenvalues.

(d)T. One purpose of PCA is to deduct dimensions.

(e)F. Eigenvalurs are very possible to be negative.

(f).F.y-axis is ordered eigenvalues of PC, possible to exceed 1.

(g)T.PCA will never increase dimensions.