STAT432_HW4

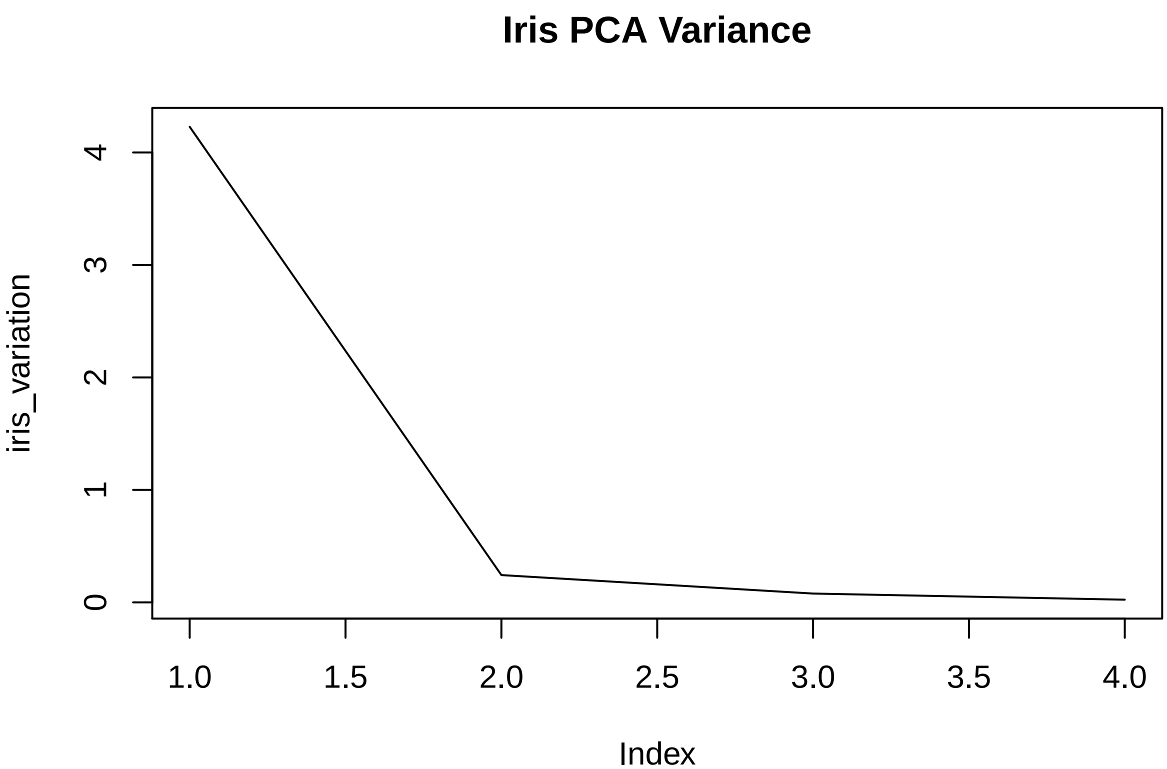
Taiga Hasegawa(taigah2)

2019/2/17

Question1

```
data("iris")
n=dim(iris)[1]
x=iris[,1:4]
cx <- sweep(x, 2, colMeans(x), "-")
svd=svd(cx)
```

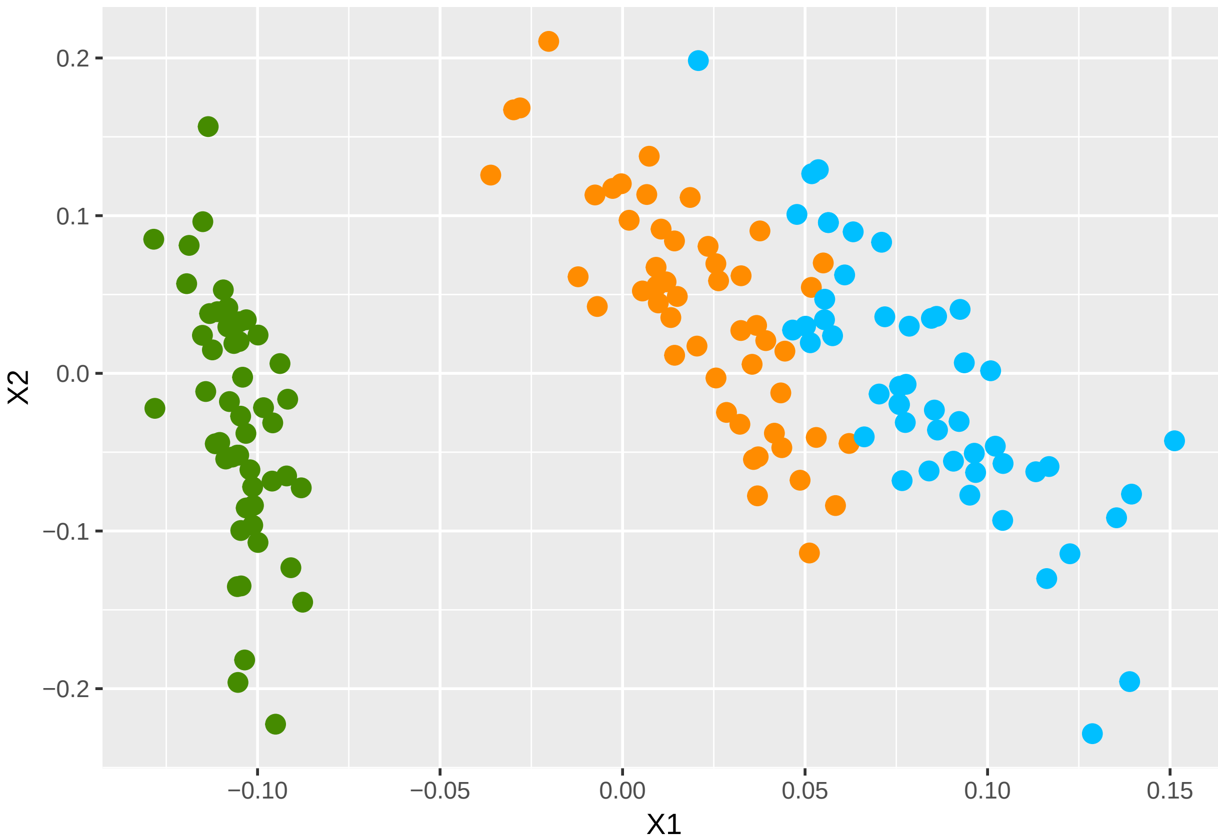
```
iris_variation=svd$d^2/(n - 1)
plot(iris_variation, type = "l", pch = 19, main = "Iris PCA Variance")
```




```
svd$v
```

```
##             [,1]        [,2]        [,3]       [,4]
## [1,]  0.36138659 -0.65658877  0.58202985  0.3154872
## [2,] -0.08452251 -0.73016143 -0.59791083 -0.3197231
## [3,]  0.85667061  0.17337266 -0.07623608 -0.4798390
## [4,]  0.35828920  0.07548102 -0.54583143  0.7536574
```

```
library(ggplot2)
ggplot(data = data.frame(svd$u), aes(x=X1 ,y=X2)) +
    geom_point(color=c("chartreuse4", "darkorange", "deepskyblue")[iris$Species], size = 3)
```



Question2

step1)

```
library(ElemStatLearn)
train.x = zip.train[, -1]
train.y = as.factor(zip.train[, 1])
test.x.one = zip.test[4, -1]
test.y.one=zip.test[4,1]
```

```
#change the test.x.one into matrix
test.x.one.matrix=t(matrix(rep(test.x.one,7291),256,7291))
#calculate the Eucleadian distance and find 15 nearest neighbors
(index=order(rowSums((test.x.one.matrix-train.x)^2))[1:15])
```

```
##  [1] 5198 5143 1825 2240 6450 4188 4187 1619 3988 5106  521 6774 6976  389
## [15] 3471
```

step2)

```
#the most frequent digit among these 15 observations
names(which.max(table(train.y[index])))
```

```
## [1] "0"
```

```
#true digit
test.y.one
```

```
## [1] 6
```

```r
#change the k to 3
index=order(rowSums((test.x.one.matrix-train.x)^2))[1:3]
names(which.max(table(train.y[index])))
```

```
## [1] "6"
```

We could get the true label by changeing the value of k.

step3)

```r
# knn function
knn=function(x,k){
  index=order(rowSums((x-train.x)^2))[1:k]
  label=as.numeric(names(which.max(table(train.y[index]))))
  return(label)
}
```

```r
#define the test dataset
test.x=zip.test[1:100,-1]
test.y=zip.test[1:100,1]

#find the best k by calculating the accuracy of each k
for(k in 1:20){
  correct=0
  for (i in 1:100){
    x=t(matrix(rep(test.x[i,],7291),256,7291))
    label=knn(x,k)
    correct=correct+sum(label==test.y[i])
  }
  accuracy=correct/length(test.y)
  print(accuracy)
}
```

```
## [1] 0.94
## [1] 0.94
## [1] 0.96
## [1] 0.93
## [1] 0.94
## [1] 0.95
## [1] 0.95
## [1] 0.95
## [1] 0.93
## [1] 0.93
## [1] 0.93
## [1] 0.93
## [1] 0.92
## [1] 0.92
## [1] 0.92
## [1] 0.92
## [1] 0.92
## [1] 0.92
## [1] 0.92
## [1] 0.92
```

When k=3, the test accuracy is the highest.

Question3

```r
library(caret)
```

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

```r
#cross validation using aret package
TrainData=data.frame(train.x)
knnFit1 <- train(TrainData, train.y,
                 method = "knn",
                 preProcess = c("center", "scale"),
                 tuneLength = 10,
                 trControl = trainControl(method = "cv",number = 3))
knnFit1
```

```
## k-Nearest Neighbors
##
## 7291 samples
##  256 predictor
##   10 classes: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'
##
## Pre-processing: centered (256), scaled (256)
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 4859, 4864, 4859
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9480213  0.9417286
##    7  0.9450053  0.9383392
##    9  0.9430848  0.9361812
##   11  0.9404781  0.9332505
##   13  0.9355408  0.9277056
##   15  0.9316994  0.9233909
##   17  0.9296432  0.9210793
##   19  0.9285453  0.9198450
##   21  0.9262147  0.9172267
##   23  0.9255282  0.9164594
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

k=5 was selected by using cross validation.

```r
#define the test dataset
test.x=zip.test[,-1]
test.y=zip.test[,1]
#number of test data
n=dim(test.x)[1]
labels=rep(NA,n)
#knn
for (i in 1:n){
  x=t(matrix(rep(test.x[i,],7291),256,7291))
  label=knn(x,5)
  labels[i]=label
}
#confusing matrix
```

```r
table(labels,test.y)
```

```
##       test.y
## labels   0   1   2   3   4   5   6   7   8   9
##      0 354   0   7   2   0   5   3   0   5   1
##      1   0 259   0   0   4   0   0   3   0   0
##      2   3   0 182   2   1   1   2   1   0   0
##      3   0   0   1 154   0   7   0   0   4   0
##      4   0   3   1   0 183   0   2   4   0   3
##      5   0   0   0   5   0 144   0   1   2   1
##      6   1   2   1   0   2   0 163   0   1   0
##      7   0   0   2   1   2   0   0 138   1   4
##      8   0   0   4   0   0   0   0   0 151   0
##      9   1   0   0   2   8   3   0   0   2 168
```