STAT 542 Coding Assignment1

Chenzi Zhang, NetID: chenziz2, UIN 654728837

1/23/2019

Contents

Method 1: generate one matrix for all means

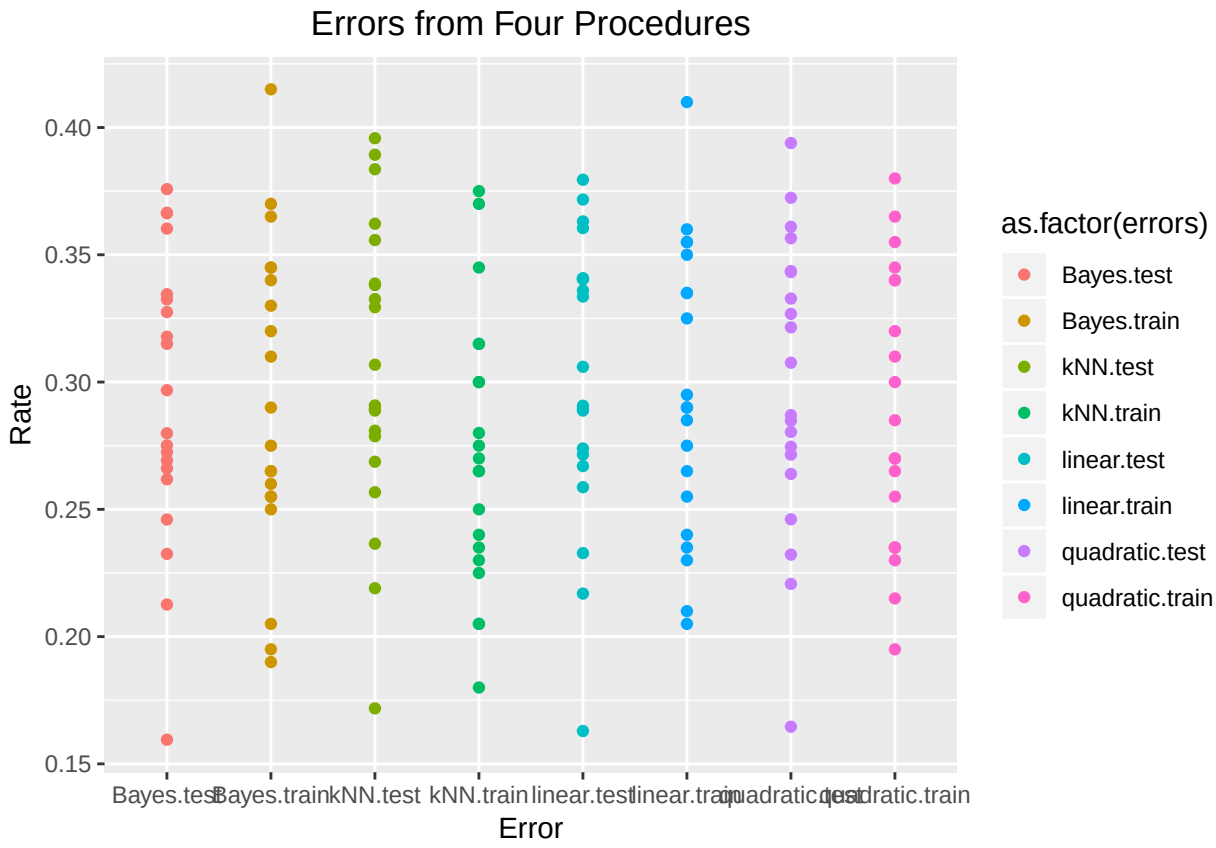
```
set.seed(200)
#6. Simulation for 20 times

library("class")
library("ggplot2")

#6.1 Set some constant value
lsize <- 10
p <- 2
n <- 100
N <- 5000
s <- 1
times <- 20
kNN_k <-c(rep(0,20))


#6.2 Set result storeage space
data_train.20 <- NULL
data_test.20 <- NULL
Ytrain.20 <-NULL
Ytest.20 <- NULL

train.error.ls.20 <- NULL
test.error.ls.20 <-NULL

train.error.Q.20 <- NULL
test.error.Q.20 <- NULL

train.error.knn.20 <-NULL
test.error.knn.20 <- NULL

train.error.Bayes.20 <- NULL
test.error.Bayes.20 <- NULL


#Simulation loop for 20 times
for(time in 1:times){
```

```r
#Set mean matrix
m <- matrix(rnorm(2*lsize*p),2*lsize,p) +
  rbind(cbind(rep(1,lsize),rep(0,lsize)),cbind(rep(0,lsize),rep(1,lsize)))

#Generate the training data
i1 <- sample(1:lsize, n, replace = TRUE)
i0 <- lsize + sample(1:lsize, n, replace = TRUE)

data_train <- matrix(rnorm(2*n*p),2*n,p)*s + rbind(m[i1, ], m[i0, ])
Ytrain <- factor(c(rep(1,n),rep(0,n)))

#Generate the test data
i1 <- sample(1:lsize, N, replace = TRUE)
i0 <- lsize + sample(1:lsize, N, replace = TRUE)

data_test <- matrix(rnorm(2*N*p),2*N,p)*s + rbind(m[i1, ], m[i0, ])
Ytest <- factor(c(rep(1,N),rep(0,N)))

#1. Linear regression with cut-off value 0.5
LinModel <- lm(as.numeric(Ytrain) - 1 ~ data_train)
Ytrain_pred_LS <- as.numeric(LinModel$fitted > 0.5)
Ytest_pred_LS <- LinModel$coef[1] + LinModel$coef[2]*data_test[,1] +
                 LinModel$coef[3]*data_test[,2]
Ytest_pred_LS <- as.numeric(Ytest_pred_LS > 0.5)

train.error.ls <- sum(Ytrain != Ytrain_pred_LS) / (2*n)
test.error.ls <- sum(Ytest != Ytest_pred_LS) / (2*N)


#2. Quadratic regression with cut-off value 0.5

QModel <- lm(as.numeric(Ytrain) - 1 ~ polym(data_train,degree=2, raw=TRUE))
Ytrain_pred_Q <- as.numeric(QModel$fitted > 0.5)

Ytest_pred_Q <- QModel$coef[1] + QModel$coef[2]*data_test[,1] +
  QModel$coef[3]*I(data_test[,1]^2) + QModel$coef[4]*data_test[,2] +
  QModel$coef[5]*I(data_test[,1]*data_test[,2]) +
  QModel$coef[6]*I(data_test[,2]^2)
Ytest_pred_Q <- as.numeric(Ytest_pred_Q > 0.5)

train.error.Q <- sum(Ytrain != Ytrain_pred_Q) / (2*n)

test.error.Q <- sum(Ytest != Ytest_pred_Q) / (2*N)

#3. kNN classification with k chosen by 10-fold cross-validation

fold <- 10
foldsize <- 2*n/fold
i.cv <- sample(1:nrow(data_train),n*2,replace = FALSE)

test.error.knn.cv <- matrix(0,(2*n-foldsize),fold)

for(j in 1:(2*n-foldsize)){
```

```
  for(i in 0:9){
    index <- i.cv[(1+i*foldsize):(foldsize+i*foldsize)]
    train.cv <- data_train[-index,]
    test.cv <- data_train[index,]
    Ytrain.cv <- Ytrain[-index]
    Ytest.cv <- Ytrain[index]

    Ytest_pred_cv <- knn(train.cv,test.cv,Ytrain.cv, k = j)
    test.error.knn.cv[j,(i+1)] <- sum(Ytest.cv != Ytest_pred_cv)/foldsize
  }
}

test.error.knn1 <- matrix(apply(test.error.knn.cv, 1 ,sum),(2*n-foldsize),1)/foldsize

kNN_k[time] <- max(which(test.error.knn1 == min(test.error.knn1)))
Ytrain_pred  <- knn(data_train, data_train, Ytrain, k = kNN_k[time])
train.error.knn <- sum(Ytrain != Ytrain_pred)/(2*n)
Ytest_pred  <- knn(data_train, data_test, Ytrain, k = kNN_k[time])
test.error.knn <- sum(Ytest != Ytest_pred)/(2*N)




#4. Bayes rule

mixnorm <- function(x){
  sum(exp(-apply((t(m[1:lsize,])-x)^2, 2,
                 sum)))/sum(exp(-apply((t(m[((lsize+1)):(2*lsize),])-x)^2, 2, sum)))
}

Ytrain_pred_Bayes <- apply(data_train, 1, mixnorm)
Ytrain_pred_Bayes <- as.numeric(Ytrain_pred_Bayes > 1)
train.error.Bayes <- sum(Ytrain != Ytrain_pred_Bayes) / (2*n)

Ytest_pred_Bayes <- apply(data_test, 1, mixnorm)
Ytest_pred_Bayes <- as.numeric(Ytest_pred_Bayes > 1)
table(Ytest, Ytest_pred_Bayes)
test.error.Bayes <- sum(Ytest != Ytest_pred_Bayes) / (2*N)

#5. Store errors
data_train.20 <- cbind(data_train.20, data_train)
data_test.20 <- cbind(data_test.20, data_test)
Ytrain.20 <- cbind(Ytrain.20, Ytrain)
Ytest.20 <- cbind(Ytest.20, Ytest)

train.error.ls.20 <- cbind(train.error.ls.20, train.error.ls)
test.error.ls.20 <- cbind(test.error.ls.20, test.error.ls)

train.error.Q.20 <- cbind(train.error.Q.20, train.error.Q)
test.error.Q.20 <- cbind(test.error.Q.20, test.error.Q)

train.error.knn.20 <-cbind(train.error.knn.20, train.error.knn)
test.error.knn.20 <- cbind(test.error.knn.20, test.error.knn)
```

```
  train.error.Bayes.20 <- cbind(train.error.Bayes.20, train.error.Bayes)
  test.error.Bayes.20 <- cbind(test.error.Bayes.20, test.error.Bayes)

}



#7. Plot
library(ggplot2)

#7.1 Plot errors from four procedures

mydata <- data.frame(x1 = rbind(t(train.error.ls.20),t(test.error.ls.20),
                                t(train.error.Q.20), t(test.error.Q.20),
                                t(train.error.knn.20),t(test.error.knn.20),
                                t(train.error.Bayes.20),t(test.error.Bayes.20)),
                     errors = c(rep("linear.train",20),rep("linear.test",20),
                                rep("quadratic.train",20), rep("quadratic.test",20),
                                rep("kNN.train",20), rep("kNN.test",20),
                                rep("Bayes.train",20),rep("Bayes.test",20)))

plot1 <- ggplot(mydata,aes(x = as.factor(errors),y = x1)) +
                geom_boxplot(fill = "cornflowerblue", colour = "darkblue") +
                xlab("Error") + ylab("Rate")+
                ggtitle("Errors from Four Procedures") +
                theme(plot.title = element_text(hjust=0.5))
plot1
```


```
plot2 <- ggplot(mydata,aes(x = as.factor(errors),y = x1, color = as.factor(errors))) +
          geom_point() +
          ggtitle("Errors from Four Procedures") + xlab("Error") + ylab("Rate") +
          theme(plot.title = element_text(hjust=0.5))
plot2
```




```
#7.2 Show the mean and standard error of selected k values

kNN_k
```

```
##  [1]  12  88  60  59  37  37  21  39  60  68 154 119  14  10  33  13  56
## [18]   7  49 130
```

```
mean(kNN_k)
```

```
## [1] 53.3
```

```
sd(kNN_k)
```

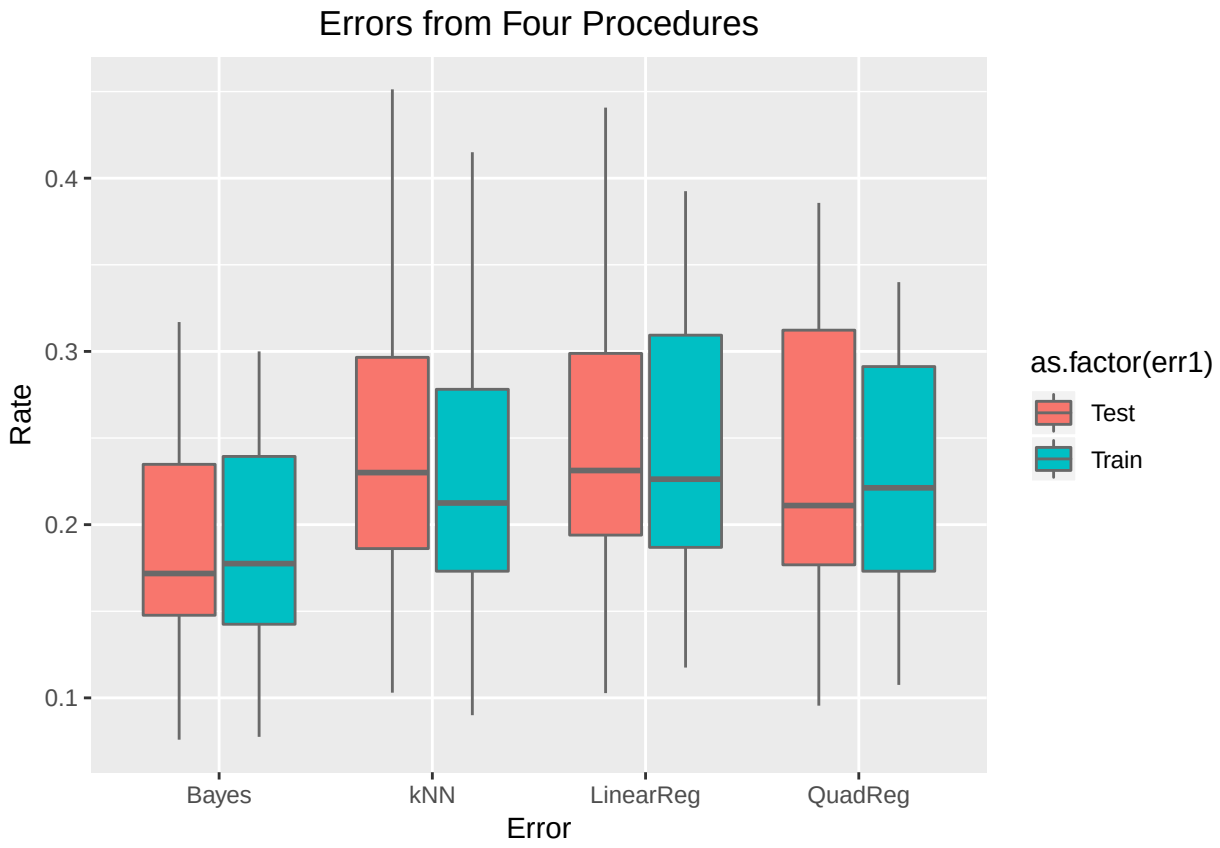
```
## [1] 41.67051
```

Method 2: generating method from Rcode example II

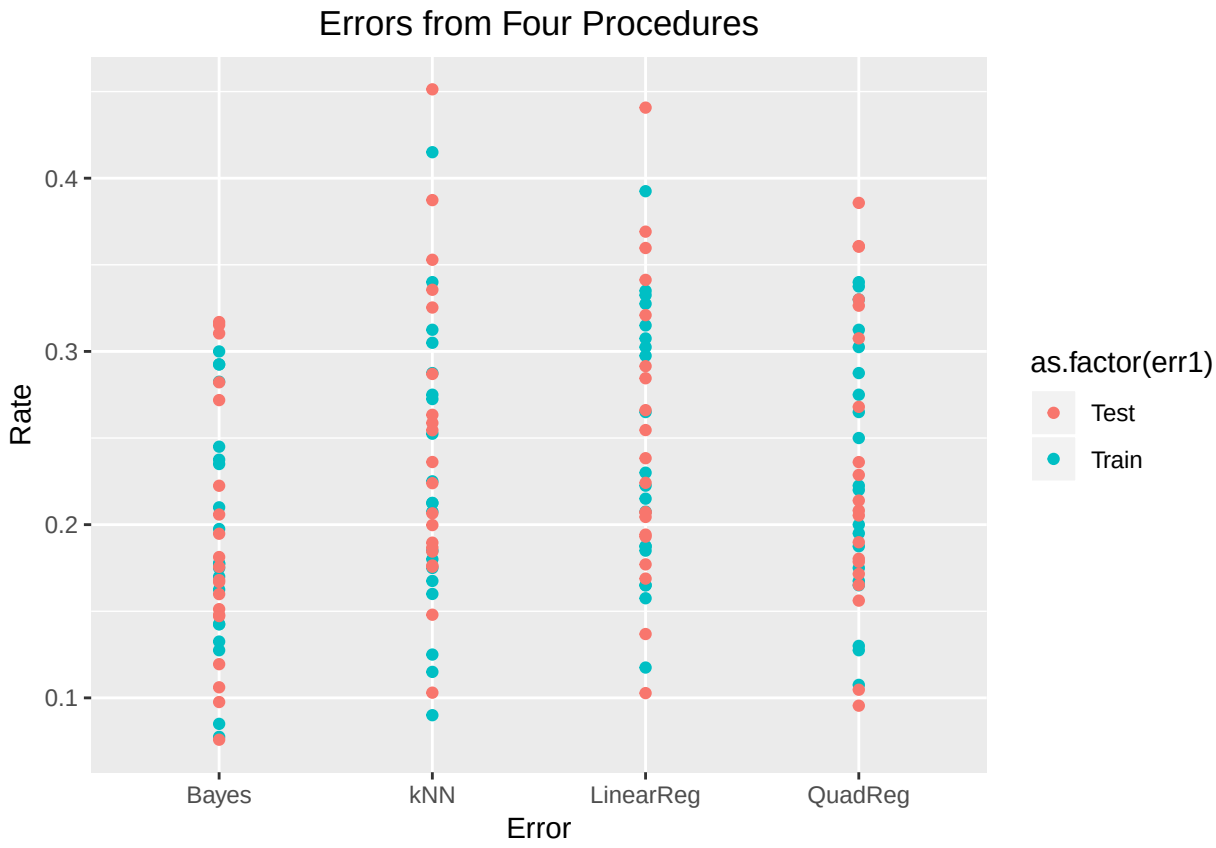
```
# 1.Set some constant values
T = 20
no.method = 4
Test.err = matrix(0, T, no.method);
colnames(Test.err) = c("LinearReg", "QuadReg", "kNN", "Bayes")
Train.err=Test.err
k.vals = rep(0, T)

# 2.Simulation for 20 times
for(t in 1:T){
```

```
# 3. generate training and test data
csize = 10;       # number of centers = l
p = 2;
s = 1;      # sd for generating the centers within each class
m1 = matrix(rnorm(csize*p), csize, p)*s + cbind( rep(1,csize), rep(0,csize));
m0 = matrix(rnorm(csize*p), csize, p)*s + cbind( rep(0,csize), rep(1,csize));

# 3.1 training data
n=200;
id1 = sample(1:csize, n, replace = TRUE);
id0 = sample(1:csize, n, replace = TRUE);
s= sqrt(1/5);
traindata = matrix(rnorm(2*n*p), 2*n, p)*s + rbind(m1[id1,], m0[id0,])
Ytrain = factor(c(rep(1,n), rep(0,n)))

# 3.2 test data
N = 10000;
id1 = sample(1:csize, N, replace=TRUE);
id0 = sample(1:csize, N, replace=TRUE);
testdata = matrix(rnorm(2*N*p), 2*N, p)*s +
  rbind(m1[id1,], m0[id0,])
Ytest = factor(c(rep(1,N), rep(0,N)))

# 4.1 call linear regression; record training and test errors
LinModel <- lm(as.numeric(Ytrain) - 1 ~ traindata)
Ytrain_pred_LS <- as.numeric(LinModel$fitted > 0.5)
Ytest_pred_LS <- LinModel$coef[1] + LinModel$coef[2]*testdata[,1] +
  LinModel$coef[3]*testdata[,2]
Ytest_pred_LS <- as.numeric(Ytest_pred_LS > 0.5)

Train.err[t,1] <- sum(Ytrain != Ytrain_pred_LS) / (2*n)
Test.err[t,1] <- sum(Ytest != Ytest_pred_LS) / (2*N)


# 4.2 call quadratic regression; record training and test errors
QModel <- lm(as.numeric(Ytrain) - 1 ~ traindata[,1] + I(traindata[,1]^2) +
             traindata[,2] + I(traindata[,1]*traindata[,2]) +
             I(traindata[,2]^2))
Ytrain_pred_Q <- as.numeric(QModel$fitted > 0.5)

Ytest_pred_Q <- QModel$coef[1] + QModel$coef[2]*testdata[,1] +
  QModel$coef[3]*I(testdata[,1]^2) + QModel$coef[4]*testdata[,2] +
  QModel$coef[5]*I(testdata[,1]*testdata[,2]) +
  QModel$coef[6]*I(testdata[,2]^2)
Ytest_pred_Q <- as.numeric(Ytest_pred_Q > 0.5)

Train.err[t,2] <- sum(Ytrain != Ytrain_pred_Q) / (2*n)
Test.err[t,2]<- sum(Ytest != Ytest_pred_Q) / (2*N)


# 4.3 call kNN; record training and test errors, and k value
fold.id = rep(1:10, each = 20)
fold.id = fold.id[sample(1:n, n, replace = FALSE)]
```

```
  cv.error <- rep(0,180)
  for(j in 1:180){
    Y.pred = rep(0, n)
    for(i in 1:10){
      test.id = (1:n)[fold.id == i]
      Y.pred[test.id] = knn(traindata[-test.id, ],
                            traindata[test.id, ],
                            Ytrain[-test.id], k=j)
    }
    cv.error[j] <- sum(Ytrain != Y.pred)/n
  }
  k.vals[t] <- max(which(cv.error == min(cv.error)))

  Ytrain_pred  <- knn(traindata, traindata, Ytrain, k = k.vals[t])
  Train.err[t,3] <- sum(Ytrain != Ytrain_pred)/(2*n)
  Ytest_pred  <- knn(traindata, testdata, Ytrain, k = k.vals[t])
  Test.err[t,3] <- sum(Ytest != Ytest_pred)/(2*N)

  # 4.4 call Bayes; record training and test errors
  mixnorm <- function(x){
    sum(exp(-apply((t(m1)-x)^2, 2, sum)*5/2))/sum(exp(-apply((t(m0)-x)^2, 2, sum)*5/2))
  }

  Ytrain_pred_Bayes <- apply(traindata, 1, mixnorm)
  Ytrain_pred_Bayes <- as.numeric(Ytrain_pred_Bayes > 1)
  Train.err[t,4] <- sum(Ytrain != Ytrain_pred_Bayes) / (2*n)

  Ytest_pred_Bayes <- apply(testdata, 1, mixnorm)
  Ytest_pred_Bayes <- as.numeric(Ytest_pred_Bayes > 1)
  Test.err[t,4] <- sum(Ytest != Ytest_pred_Bayes) / (2*N)
}


#5.1 Produce boxplot(s) based Train.err and Test.err
mydata2 <- data.frame(x1 = c(cbind(Train.err,Test.err)),
                      err1 = c(rep("Train",20*4),rep("Test",20*4)),
                      err2 = c(rep(colnames(Test.err), each = 20),
                               rep(colnames(Test.err), each = 20)))

plot3 <- ggplot(mydata2,aes(x = as.factor(err2),y = x1,fill = as.factor(err1))) +
         geom_boxplot(colour = "dimgray") +
         xlab("Error") + ylab("Rate") +
         ggtitle("Errors from Four Procedures") +
         theme(plot.title = element_text(hjust=0.5))
plot3
```




```
#5.2 Produce point(s) based Train.err and Test.err

plot4 <- ggplot(mydata2,aes(x = as.factor(err2),y = x1, color = as.factor(err1))) +
  geom_point() +
  ggtitle("Errors from Four Procedures") + xlab("Error") + ylab("Rate") +
  theme(plot.title = element_text(hjust=0.5))
plot4
```




```
#These two graphs are seperated from the graph above
mydata3 <- data.frame(x1 = c(Train.err), x2 = c(Test.err),
                      err1 = rep(colnames(Test.err), each = 20),
                      err2 = rep(colnames(Test.err), each = 20))
plot5 <- ggplot(mydata3, aes(x = as.factor(err1), y = x1)) +
  geom_point(colour = "cornflowerblue") +
  ggtitle("Errors from Four Procedures") + xlab("Train.Error") + ylab("Rate") +
  theme(plot.title = element_text(hjust=0.5))
plot6 <- ggplot(mydata3,aes(x = as.factor(err2),y = x2)) +
  geom_point(colour = "lightcoral") +
  ggtitle("Errors from Four Procedures") + xlab("Test.Error") + ylab("Rate") +
  theme(plot.title = element_text(hjust=0.5))
plot5
```


Errors from Four Procedures
Rate
0.4
0.3
0.2
0.1
Bayes
kNN
LinearReg
QuadReg
Train.Error

```
plot6
```


```
#6. Record mean and sd for k.vals
k.vals
```

```
##  [1] 175 166 145   2 160 129 180 180 168  82 180  75 179 138 154 180 135
## [18]   2   2  74
```

```
mean(k.vals)
```

```
## [1] 125.3
```

```
sd(k.vals)
```

```
## [1] 63.30054
```