Homework12

Jun Rao

11/2/2020

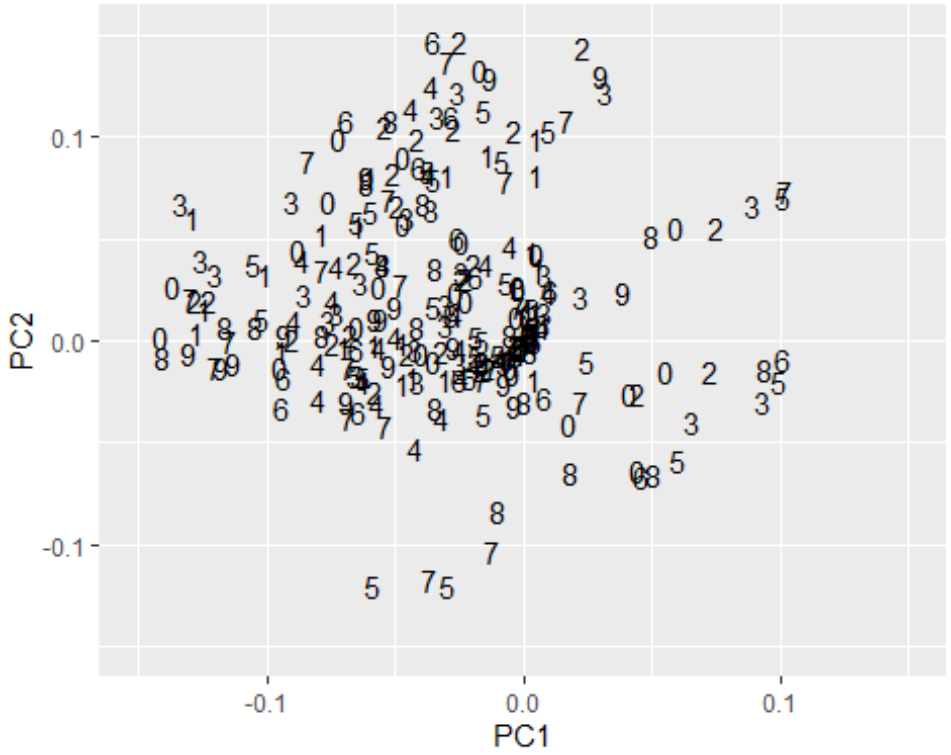
Principal components analysis. The goal of this homework is to use PCA for dimensionality reduction in a data set of handwritten digit images.

```
library(ggplot2)
library(data.table)
dt<- fread("zip.train.gz")
```

1.use prcomp(scale=TRUE) on 100 rows of the zip.train data from the ESL book (10 from each class). Make a ggplot with a geom_text that shows each row with its corresponding class label (x=PC1, y=PC2, label=class). Do the classes cluster together in the plot?

```
sub.dt <- data.table()

index_seq <- sort(unique(dt$V1))

for (index in index_seq) {

  sub <- subset(dt, dt$V1 == index)
  row.dt <- sub[sample(nrow(sub), 10),]

  # rotation.dt <- rbind(rotation.dt,temp.dt)
  sub.dt <- rbind(sub.dt,row.dt)
}



pc.fit <- prcomp(sub.dt)

class <- rep(0:9, each=10)

PC1 = pc.fit[["rotation"]][,1]
PC2 = pc.fit[["rotation"]][,2]

digit.dt <- data.table(
    PC1 = PC1,
    PC2 = PC2,
    class
)

ggplot()+
  geom_text(aes(x=PC1, y=PC2, label=class),
```

```
    data=digit.dt) +
  xlim(-0.15,0.15) + ylim(-0.15,0.15)
```

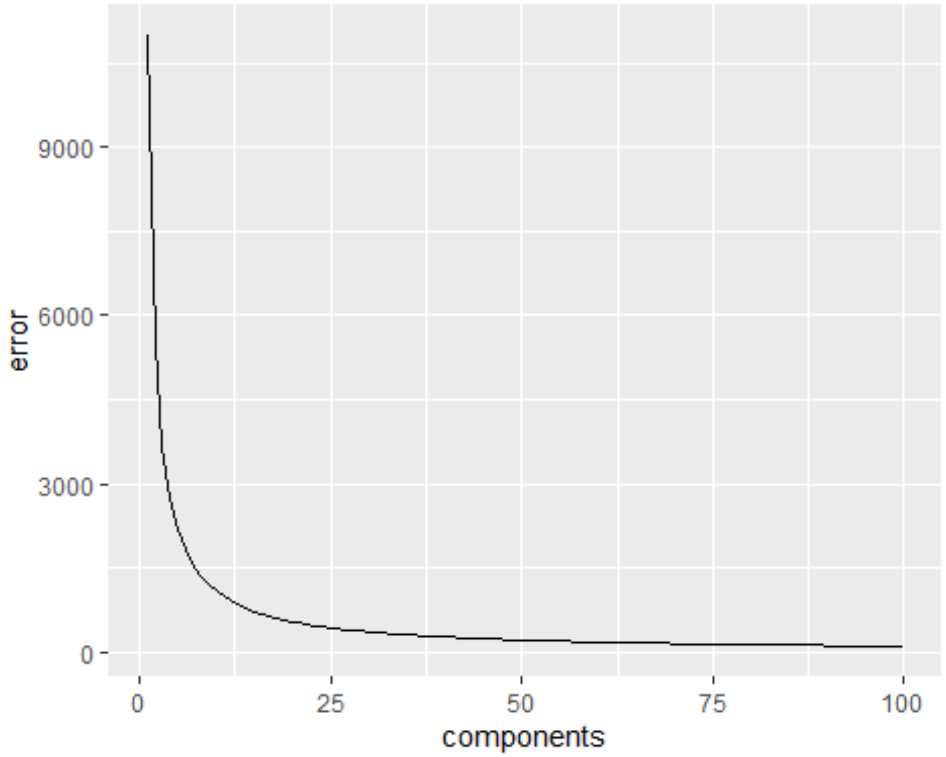


Yes. It is cluster

2.For each number of principal components from 1 to 100, compute the reconstruction error (sum of squares between the data and model). Then plot y=error as a function of x=number of components. Is the reconstruction error zero at 100 components as expected?

```
max.components <- 100

error.dt <- data.table()

for (component in 1:max.components) {
  lambda.vec <- pc.fit$x[,1] ## this is lambda from the book!
  PC <- pc.fit[["rotation"]][,1]
  PC.mat <- matrix(PC, nrow=nrow(sub.dt), ncol=ncol(sub.dt), byrow=TRUE)
  mean.vec <- colMeans(sub.dt)
  mean.mat <- matrix(mean.vec, nrow=nrow(sub.dt), ncol=ncol(sub.dt), byrow=TR
UE)
  pred.mat <- mean.mat + PC.mat * pc.fit[["x"]][,1]
  temp.error.dt <- data.table()
  temp.error.dt[, error := sum( (pred.mat - sub.dt)^2)/component]
  temp.error.dt[, components := component]
  temp.error.dt[, name := "prcomp/2"]

  error.dt <- rbind(error.dt,temp.error.dt)
```

```
}
ggplot()+
  geom_line(aes(x=components, y=error),
  error.dt)
```

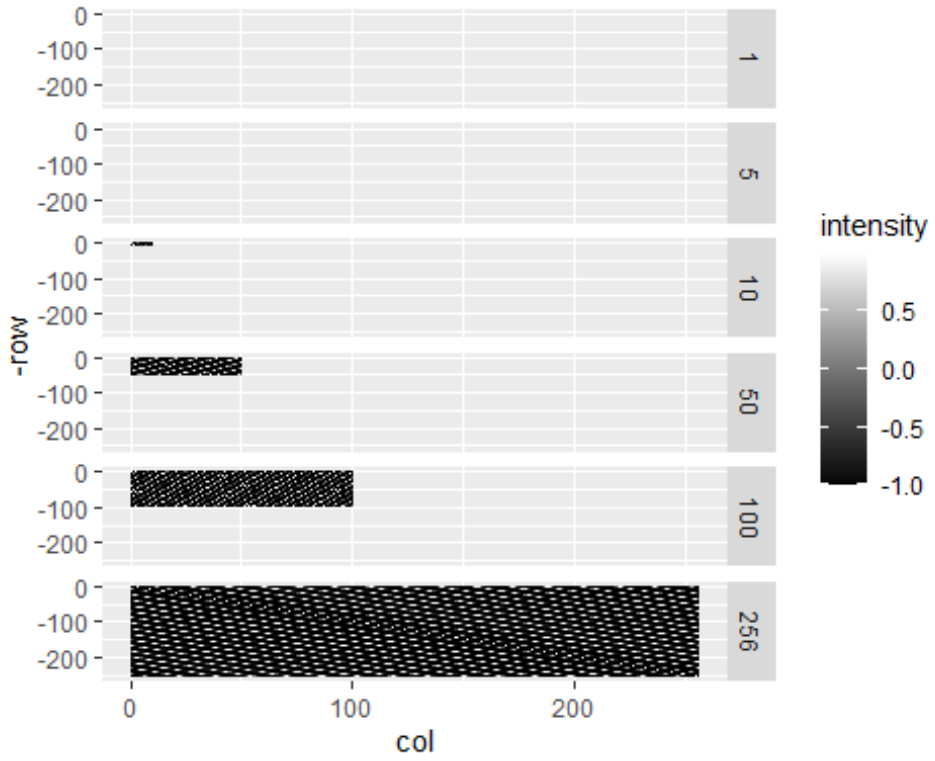


Yes. The reconstruction error zero at 100 components as we expected.

3.Choose one image/row from the 100 data you used in problem 1. For each number of components in {1, 5, 10, 50, 100} plot the reconstruction of that image in a different panel, using geom_tile. For comparison also include a panel for the original image. Is the original image equal to the model with 100 components, as expected?

```
obs.i <- 1
#Choose one image/row from the 100 data you used in problem 1
sub.one.image <- sub.dt[V1== obs.i,]

pixels <- c(1,5,10,50,100,256)

pr.one.image <- prcomp(sub.one.image, rank=2)

all.digit.list <- list()

for (n.pixels in pixels) {
```

```
  all.digit.list[[paste(n.pixels)]] <-
    data.table(
        n.pixels,
        intensity=pr.one.image$center + pr.one.image$rotation[,1] *pr.one.ima
ge$x[obs.i,1],
        row=rep(1:n.pixels, n.pixels),
        col=rep(1:n.pixels, each=n.pixels)
    )

}

all.digit.dt <- do.call(rbind,all.digit.list )

ggplot()+
  facet_grid(n.pixels ~ .)+
    geom_tile(aes(col, -row, fill=intensity),data=all.digit.dt)  +
      scale_fill_gradient(low="black", high="white")
```



We used 256 to represent the original image.

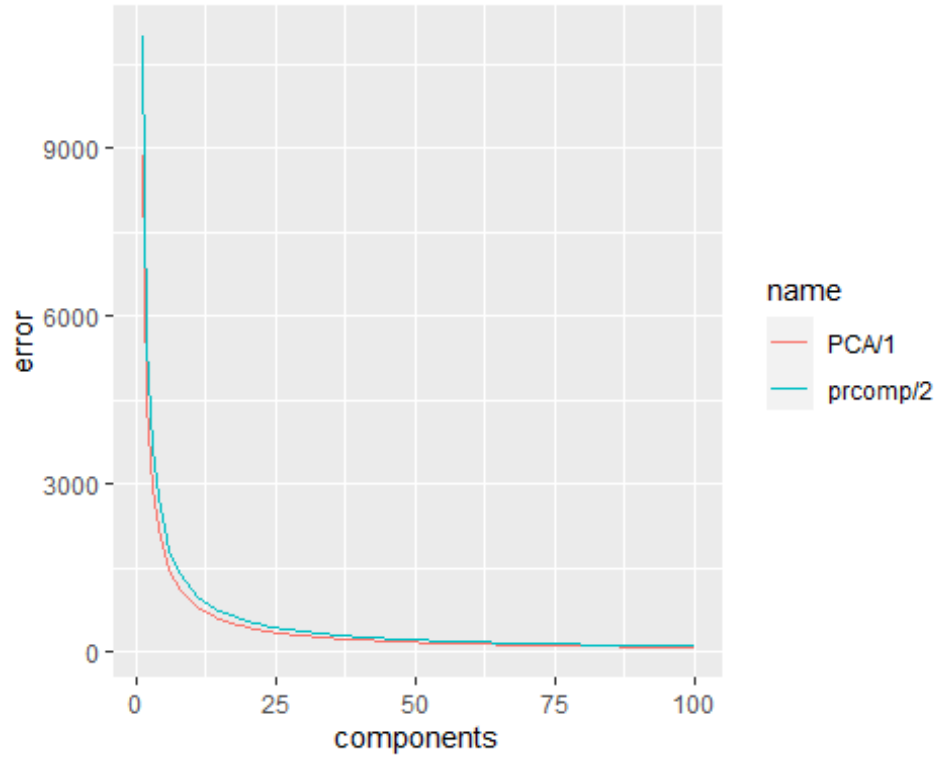
Yes. the original image equal to the model with 100 components, as expected

Your task this week is to implement a function PCA which computes principal components analysis "from scratch" using the base::svd function in R. Use the text and equations in the book, and make sure to center each column before using svd, via scale(X, center=TRUE, scale=FALSE). Use can use u %*% diag(d) to convert the svd output to the principal components matrix (rotation element of prcomp output). Use your function on the same

data set as problem 1 above. Plot the reconstruction error as in problem 2 using different colors/sizes for different functions (e.g., black=prcomp/2, red=PCA/1). Does your function result in the same values for the reconstruction error?

```
pca <- function(dt){
  #center each column
  sca.dt <- scale(dt, center=TRUE, scale=FALSE)
  #calculate the Covariance matrix
  cov.sca.dt <- cov(sca.dt)

  #Singular value decomposition
  svd.dt <- svd(cov.sca.dt)

  svd.value <- svd.dt$d
  svd.vector <-svd.dt$u

  order_value <- order(svd.value,decreasing = T)
  values <- svd.value[order_value]
  valueSum <- sum(values)
  cumVar <- cumsum(values)/valueSum * 100

  order_vector <- svd.vector[,order_value]

  principal <- sca.dt %*% order_vector


  return(list(PCA=principal, cumVar=cumVar ))
}



my.pc.fit <-pca(sub.dt)


my.error.dt <- data.table()

for (component in 1:max.components) {
    temp.error.dt <- data.table()
    temp.error.dt[, error := sum( my.pc.fit$cumVar[1:max.components])/compone
nt]
     temp.error.dt[, components := component]
    temp.error.dt[, name := "PCA/1"]
    my.error.dt <- rbind(my.error.dt,temp.error.dt)
}

final.error.dt <- data.table()

final.error.dt <- rbind(error.dt,my.error.dt)
```

```
ggplot()+
  geom_line(aes(x=components, y=error, color = name),
   final.error.dt)
```



Yes. My function isthe same values for the reconstruction error.