Computer Assignment 3 - Matrix Algebra and PCA

Machine Learning, Spring 2020

Rui Li

In this assignment, we will explore how to run a few simple (but incredibly useful) linear algebra commands in **R**. Then, we will examine how one calculates principle components using built-in **R** functions.

Transpose and Checking Equality

Given the following 5 vectors, use the `cbind` and `rbind` functions to create the matrices X and Y, where the *rows* of X are x.1-x.5 and the *columns* of Y are x.1-x.5.

```
x.1 = c(2, .5, 4, 2)
x.2 = c(1, 1, 1, 1)
x.3 = c(-1, 0, 2, 1)
x.4 = c(1, -.5, .25, 3)
x.5 = c(3, 6, 9, 12)

# Concatenate the vectors as rows
X = rbind(x.1,x.2,x.3,x.4,x.5)
# Concatenate the vectors as columns
Y = cbind(x.1,x.2,x.3,x.4,x.5)
```

We can calculate the transpose of a matrix by using the *t( )* command. For example, try typing *t(X)* and look at the output.

```
t(X)

##      x.1 x.2 x.3   x.4 x.5
## [1,] 2.0   1  -1  1.00   3
## [2,] 0.5   1   0 -0.50   6
## [3,] 4.0   1   2  0.25   9
## [4,] 2.0   1   1  3.00  12
```

Based on our construction, t(X) = Y. We can check that this is true by using a logical equivalence command ==. When checking the equality of two matrices, the == command will check the entry by entry equality of the two matrices. To do this, the two matrices must be of the same dimensions. Verify that t(X) = Y by using the command *t(X) == Y*. Verify that **R** will return an error if you type *t(x.1) == Y* because *x.1* and Y are not the same dimension.

```
t(X)==Y

##       x.1  x.2  x.3  x.4  x.5
## [1,] TRUE TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE TRUE
## [4,] TRUE TRUE TRUE TRUE TRUE
```

Matrix Multiplication

Matrix multiplication is performed in **R** using the %*% command. For example, to obtain the product AB of two matrices A and B you can type $A\%^*\%B$. Calculate the empircial covariance matrix of X using the following code:

```
#calculate the column means of X
col.means = colMeans(Y)

#create a matrix of these column means repeated across the rows
Y.bar = matrix(rep(col.means,4),ncol = 5, byrow = TRUE)

#column center X
Y.col.centered = Y - Y.bar

#calculate the empirical covariance matrix of X
sigma = 1/5 * t(Y.col.centered) %*% Y.col.centered
```

Verify that *sigma* is symmetric using the *t( )* and == commands.

```
t(sigma)==sigma

##       x.1  x.2  x.3  x.4  x.5
## x.1 TRUE TRUE TRUE TRUE TRUE
## x.2 TRUE TRUE TRUE TRUE TRUE
## x.3 TRUE TRUE TRUE TRUE TRUE
## x.4 TRUE TRUE TRUE TRUE TRUE
## x.5 TRUE TRUE TRUE TRUE TRUE
```

It is important to distinguish the difference between the %*% command and the $*$ command in **R**. The $*$ command is used for scalar multiplication and for entry-wise multiplication of matrices. Knowning this, answer the following questions:

Questions

1. Write out *sigma*, the empirical covariance matrix of X
2. Based on *sigma*, what is $Cov(x.1, x.3)$?
3. What is $X * X$? Note that this is different than what one obtains when typing $t(X)\% * \%X$.

YOUR ANSWERS HERE 1.

```
print("Write out *sigma*:")

## [1] "Write out *sigma*:"

sigma
```

```
##          x.1 x.2   x.3      x.4   x.5
## x.1 1.23750   0 0.750 0.156250 1.050
## x.2 0.00000   0 0.000 0.000000 0.000
## x.3 0.75000   0 1.000 0.125000 2.400
## x.4 0.15625   0 0.125 1.359375 2.025
## x.5 1.05000   0 2.400 2.025000 9.000
```

```
The Cov(x.1, x.3) = 0.75.
```

```
X*X
```

```
##     [,1]  [,2]    [,3] [,4]
## x.1    4  0.25 16.0000    4
## x.2    1  1.00  1.0000    1
## x.3    1  0.00  4.0000    1
## x.4    1  0.25  0.0625    9
## x.5    9 36.00 81.0000  144
```

```
t(X)%*%X
```

```
##       [,1]   [,2]     [,3]   [,4]
## [1,] 16.00 19.500  34.2500  43.00
## [2,] 19.50 37.500  56.8750  72.50
## [3,] 34.25 56.875 102.0625 119.75
## [4,] 43.00 72.500 119.7500 159.00
```

```
`X*X` is to sqaure each value of the matrix X, while `%*%` is for matrix mult
iplication.
```

Eigenvalues and Eigenvectors

The eigenvalues and eigenvectors of a matrix *A* can be calculated simply by typing the *eigen(A)* command. The output of this command will contain two components: the eigenvectors and the eigenvalues of the matrix *A*. Using the *eigen( )* command, answer the following questions:

Questions

1. What are the first and second eigenvectors of *sigma*? What are the first and second eigenvalues of *sigma*?

```
eigen(sigma)
```

```
## eigen() decomposition
## $values
## [1]  1.027860e+01  1.453891e+00  8.643809e-01  3.234013e-16 -4.996004e-16
## 
## $vectors
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,] -1.332276e-01  7.112746e-01 -6.079888e-01  3.266320e-01  0.000000e+00
## [2,] -1.110223e-16  1.387779e-16  3.885781e-16  4.996004e-16 -1.000000e+00
## [3,] -2.549459e-01  4.636730e-01  1.495866e-01 -8.352447e-01 -2.220446e-16
## [4,] -2.176586e-01 -5.172122e-01 -7.479001e-01 -3.546290e-01 -5.134781e-16
## [5,] -9.326724e-01 -1.076447e-01  2.204968e-01  2.644164e-01  1.110223e-16

The first eigenvectors of *sigma* is the first column of $vectors, and the co
rresponding eigenvalue is the first value of $values, 1.027860e+01.

The second eigenvectors of *sigma* is the second column of $vectors, and the
corresponding eigenvalue is the second value of $values, 1.453891.
```

2. Let *eig.1* be the first eigenvector of *sigma*. Based on the equations given in class, what is the variance of the projections of $x.1, x.2, \dots, x.4$ onto *eig.1*? What is the relationship between this variance and the first eigenvalue? Without calculations, what do you expect the variance of the projections of $x.1, x.2, \dots, x.4$ onto the second eigenvector of *sigma* to be?

```
eig.1 = eigen(sigma)$vectors[,1]

#Calculate the Variance
var_eig.1_x = t(eig.1)%*%sigma%*%eig.1

The varaince is equal to the eigenvalue eig.1. The variance of projections on
to the second eigenvector of *sigma* should be equal to the second eigenvalue.
```

Norms

The Euclidean norm $||\cdot||$ of a vector can be calculated in **R** using the *norm( )* command. For example, to calculate the norm of the a vector u, we can type *norm(u)*. Note that $||{\bf x}||^2$ is the sum of squared distance of entries of ${\bf x}$. For example,

```
norm(as.matrix(x.1))

## [1] 8.5
```

The default in **R** is to use the Euclidean norm; however, one can specify other types of norms by using the *type* argument in *norm(u, type =)*. Use *help(norm)* for more information. Answer the following questions:

Questions

1. Project *z.1* onto the first eigenvector of *sigma*. What is the sum of squared distance between *z.1* and its projection (calculated below)?

PCA Basics

Running a principal components analysis in **R** is fast and simple, and only makes use of one function which is available to all versions of **R**. The below description is partially taken from the publicly available **R** documentation. This documentation can be seen in full if you type *?prcomp* or *help(prcomp)* in your **R** console.

```
require('stats')
?prcomp

## starting httpd help server ... done
```

prcomp performs a principal components analysis on the given numeric data matrix **x** and returns the following outputs:

Outputs:

- *sdev*: the standard deviations of the principal components.
- *rotation*: the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors).
- *x*: the rotated data; i.e., a matrix whose columns are the principal components.
- *center*: the means that were subtracted.
- *scale*: the scalings applied to each variable.

Let's now see PCA in action on a real dataset. We first consider the *iris* dataset available in **R**. Load the data and separate the species names and quantitative data using the following commands:

```
data(iris)

#Separate the species names from the quantitative data
data = iris[,1:4]
species = iris$Species
```

Questions

1. How many variables does this data set contain? What are their names?
2. Why might PCA be helpful for this data set?
3. Run a principal components analysis on the iris dataset. What proportion of the total variation in the data is explained by each of the principal components?

```
#Run principal components analysis
pcs = prcomp(data)

#Summarize the pcs
summary(pcs)

## Importance of components:
##                           PC1     PC2    PC3     PC4
## Standard deviation     2.0563 0.49262 0.2797 0.15439
## Proportion of Variance 0.9246 0.05307 0.0171 0.00521
## Cumulative Proportion  0.9246 0.97769 0.9948 1.00000
```

YOUR ANSWERS HERE 1.

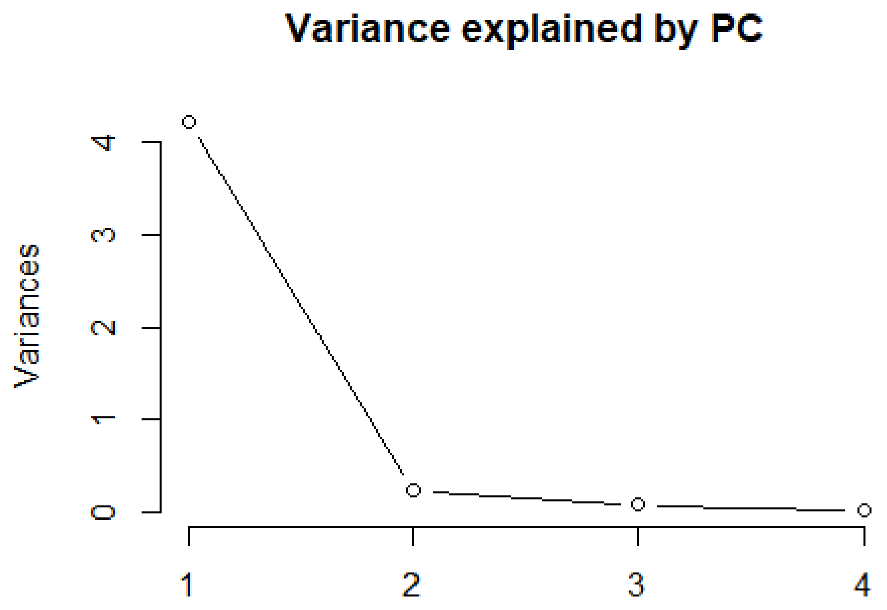
```
There are four variables in this data set. They are Sepal Length, Sepal Width,
 Petal Length and Petal Width.
```

```
The four variables may be correlated to each other, and the main idea of prin
cipal component analysis (PCA) is to reduce the dimensionality of a data set
consisting of many variables correlated with each othe.r
```

```
As shown from the output above, PC1 explains 92.46% of the total variance, PC
2 explains 5.30% of the total variance, PC3 explains 1.72% of the total varia
nce, PC4 explains 0.52% of the total variance.
```

4. Plot a scree plot of the variance explained:

```r
# Make scree plot
screeplot(pcs, type = "lines", main = "Variance explained by PC")
```



Based on the variation explained for each of these components, would you be comfortable projecting the *iris* data set down to only one dimension?

```
Yes, it is satisfied to project the *iris* data set down to only one dimensio
n. Because PC1 can explain 92.46% of the variance, showing that most informat
ion (4 variables) of the data can be encapsulated by just that one Principal
Component
```

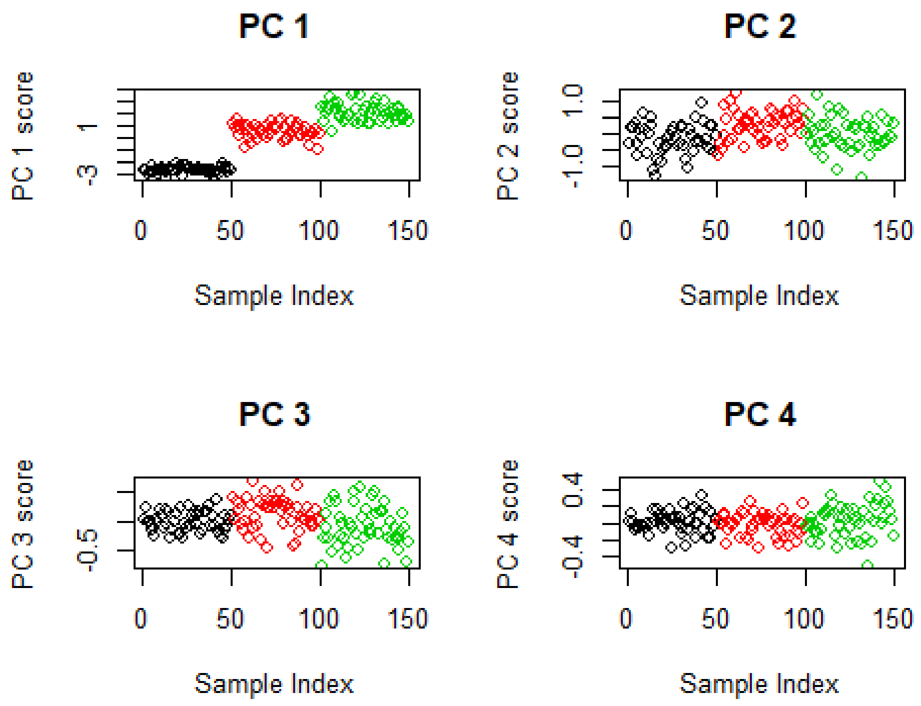
The PCs

Let's first look at each of these PCs individually. One way to do this is to look at the distribution of the PC scores for each PC. This is a good time to introduce the *for()* loop command. The *for()* loop allows one to sequentially call a command over an index specified in (). For example, try

```
for(i in 1:4){
  print(i)
}

## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

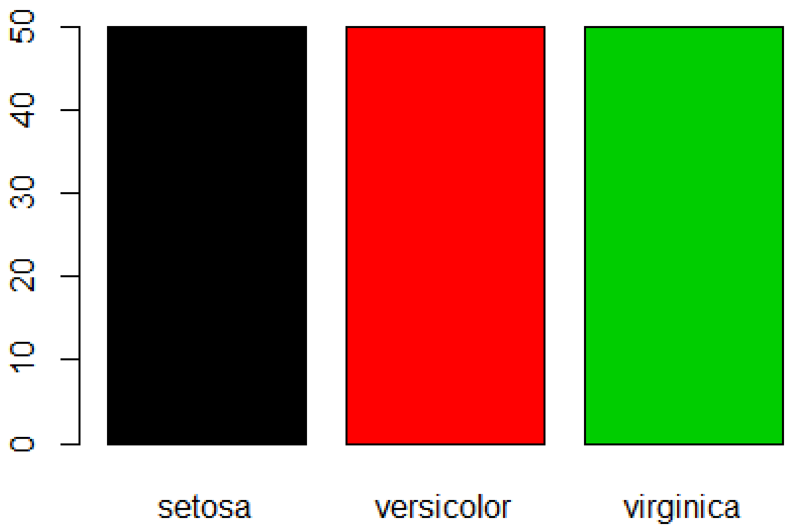
We will plot the four pcs using a *for* loop:

```
#Partition the image space into a 2 x 2 grid
par(mfrow = c(2,2))

#Now run the for loop to plot each image separately
#For each plot, we will label the points according to species

for(i in 1:4){
  plot(pcs$x[,i],
       main = paste("PC", eval(i)),
       xlab = "Sample Index",
       ylab = paste("PC", eval(i), "score"),
       col = species)
}
```



Note that the *col = species* command automatically colors the points by the flower species. Here's a quick trick for figuring out which colors correspond to which species.

```
barplot(table(species), col = unique(species))
```



Questions

1. Comment on the four pc plots. Do any of the plots reveal any interesting structure?
2. Do you think that you could distinguish the species of plant if you projected the original data onto PC4? How about PC1?

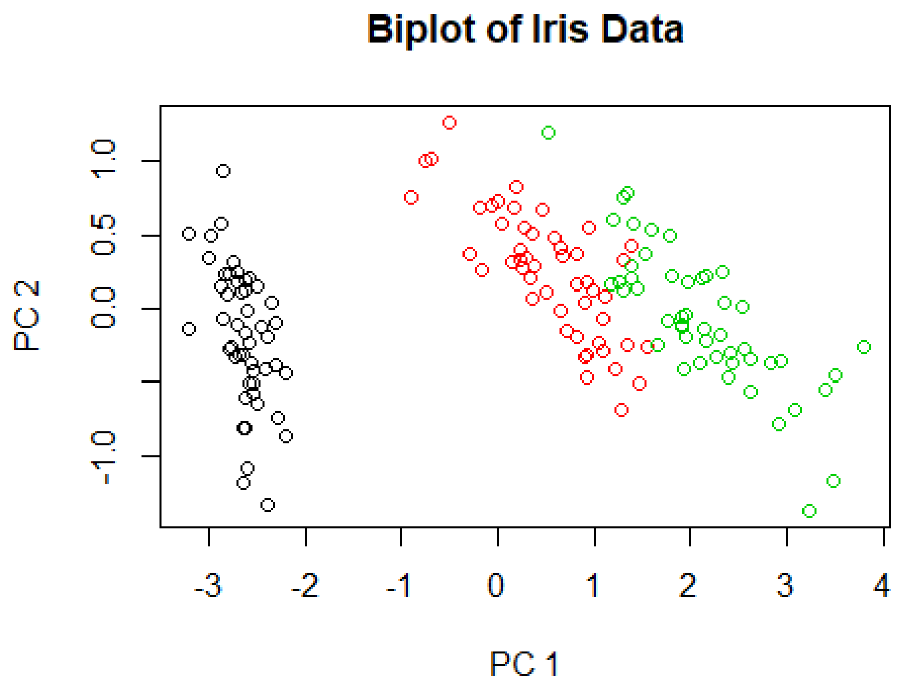
YOUR ANSWERS HERE 1.

```
The PC1 showing that 'virginica' has highest PC score, then 'versicolor', and
 finally 'setosa'.

I don't think I can distinguish the species of plant if I projected the origi
nal data onto PC4, since each group have similar PC score. But I can tell the
 difference by using PC1, since different group has different PC score.
```

Biplots

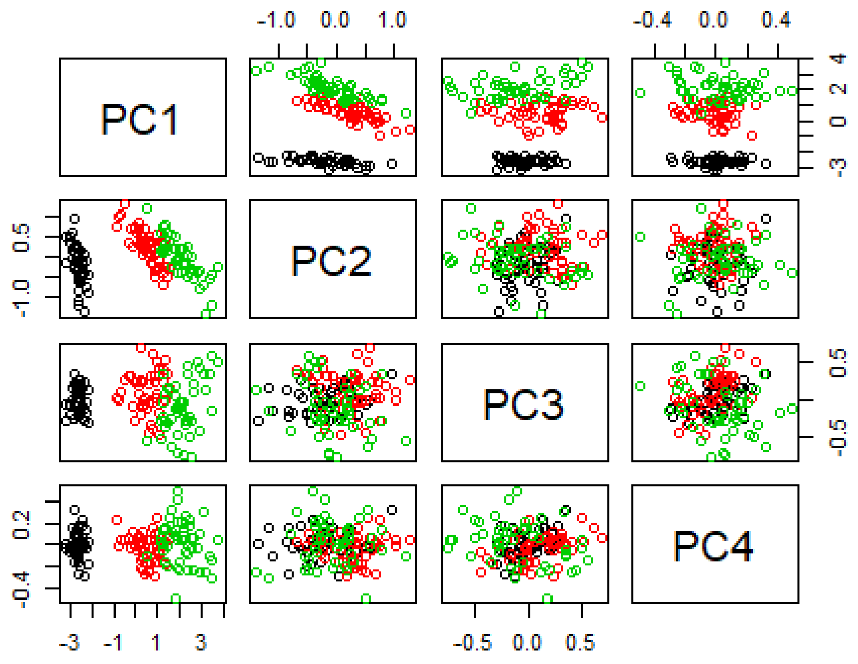
Let's see if we have any further evidence of clustering in the data by looking at a biplot of the PCs. That is, we will plot the scores of the first PC against the scores of the second. In this way, we are illustrating a 2-dimensional view of this 4-dimensional data. Create a biplot with labeled points using the following code:

```
plot(pcs$x[,1:2],
     main = "Biplot of Iris Data",
     xlab = "PC 1", ylab = "PC 2",
     col = species)
```



Next, let’s plot a pairwise biplots for each pair of pcs. Do this by using the *pairs()* command as follows:

```
pairs(pcs$x, col = species)
```



Questions

1. What features do you notice from the resulting biplot? Do the data appear to `cluster`?
2. Compare your pairwise plots to the plot of PC 1 alone. Does there appear to be any added benefit in projecting the data into more than one direction? In conclusion, to how many dimensions can we project this data and still capture the features of the species?

YOUR ANSWERS HERE 1.

```
From the biplot, it is very clear that, there are three groups showing up, an
d the data appear to `cluster`.

There is no added benefit with more than one direction, we can not tell any d
ifference between the groups with more than one dimension. We should only use
 one dimension.
```