Assignment 4

Cameron Atkins

2023-12-08

Problem 01

1a. First start by calculating the covariance matrix (denoted as Σ). Set up the equation, but use R to calculate the matrix.

```
df <- data.frame(x1 = c(2, 3, 5, 6, 10),
                 x2 = c(3, 4, 6, 7, 11))
cov_matrix <- matrix((cov(df)), nrow = 2, byrow=TRUE)
print(cov_matrix)
```

```
##      [,1] [,2]
## [1,]  9.7  9.7
## [2,]  9.7  9.7
```

1b. Find the eigenvalues by calculating det(Σ – λI) and solving for λ

```
eigen_result <- eigen(cov_matrix)
eigenvalues <- eigen_result$values
eigenvectors <- eigen_result$vectors

# Print eigenvalues
print(eigenvalues)
```

```
## [1] 19.4  0.0
```

1c. Compute the eigenvectors corresponding to each eigenvalue using the properties of eigenvalues and eigenvectors: Σei = λei Where ei is the ith eigenvector. Then convert ei to a unit eigenvector using the formula: ei/||ei||

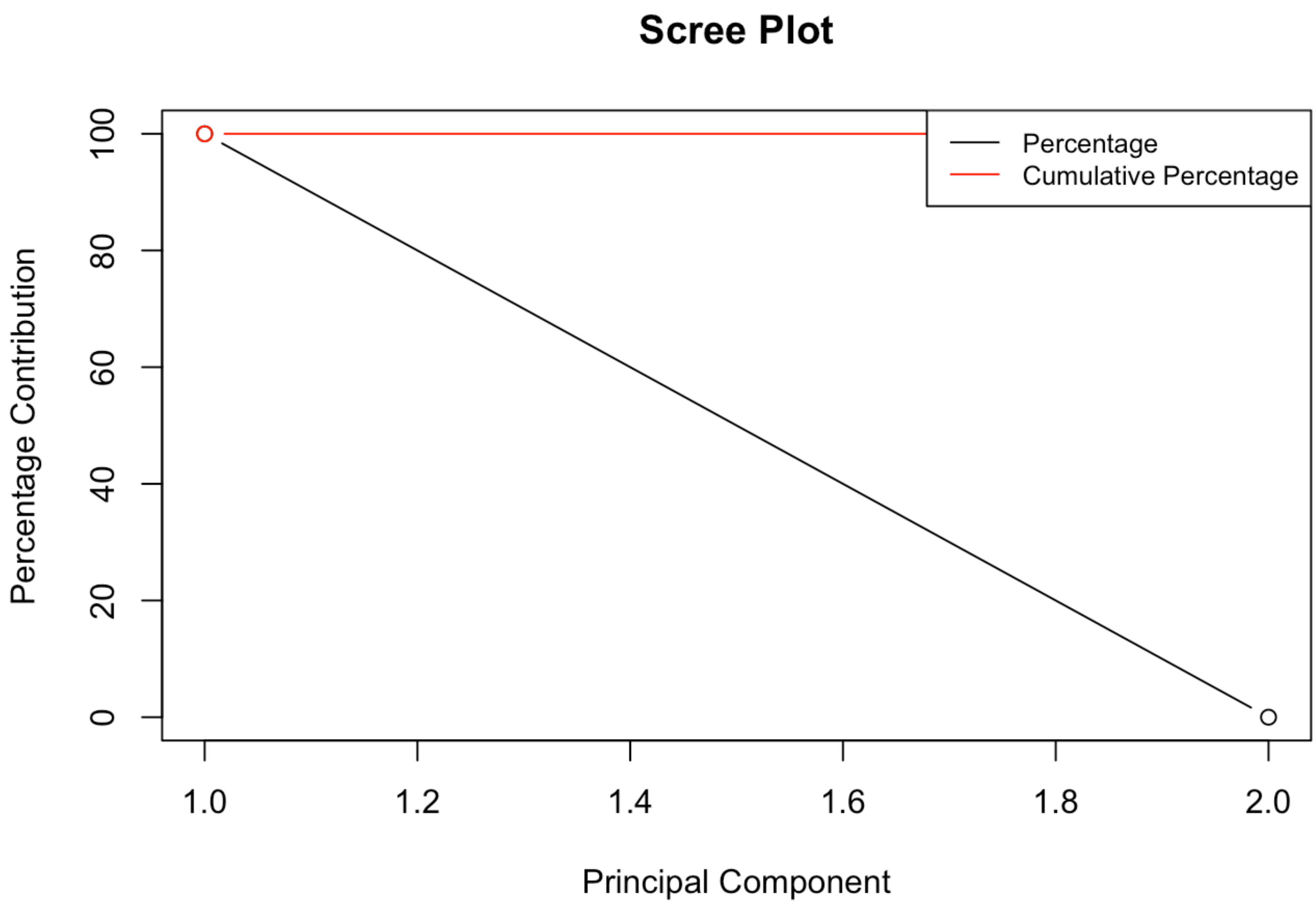
```
#Eigenvectors
print(eigenvectors)
```

```
##           [,1]       [,2]
## [1,] 0.7071068 -0.7071068
## [2,] 0.7071068  0.7071068
```

```r
# Normalize and print each eigenvector
normalized_eigenvectors <- eigenvectors / sqrt(rowSums(eigenvectors^2))
print(normalized_eigenvectors)
```

```
##           [,1]       [,2]
## [1,] 0.7071068 -0.7071068
## [2,] 0.7071068  0.7071068
```

```r
# Finally compute the percentage of contribution from each eigenvector by summing
# the eigenvalues together and dividing each individual eigenvalue by the total. Draw
a
# Scree plot to show the differences in contribution from the principal components

percentage_contribution <- eigenvalues / sum(eigenvalues) * 100

# Create a Scree plot
plot(percentage_contribution, type = "b", main = "Scree Plot",
     xlab = "Principal Component", ylab = "Percentage Contribution")

# Add cumulative percentage contribution
cum_percentage <- cumsum(percentage_contribution)
lines(cum_percentage, col = "red", type = "b")

# Add labels
legend("topright", legend = c("Percentage", "Cumulative Percentage"),
       col = c("black", "red"), lty = 1:1, cex = 0.8)
```



Problem 02

With R, create a PCA plot with PC1 on the X-axis and PC2 on the Y-axis. Construct a Scree plot.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
iris_test = iris[order(iris$Species),]
rownames(iris_test) = paste(substr(iris_test[,5],1,2),
sample(1:2000, 150, replace = FALSE),sep = "")
set.seed(641)
iris_test = sample_n(iris_test, 20)

iris_test
```

	Sepal.Length <dbl>	Sepal.Width <dbl>	Petal.Length <dbl>	Petal.Width <dbl>	Species <fct>
se408	5.1	3.8	1.9	0.4	setosa
ve1717	5.6	3.0	4.1	1.3	versicolor
ve850	5.5	2.6	4.4	1.2	versicolor
se1631	4.7	3.2	1.6	0.2	setosa
ve1865	5.8	2.7	4.1	1.0	versicolor
se130	5.1	3.5	1.4	0.3	setosa
ve1311	6.0	3.4	4.5	1.6	versicolor
ve1672	5.7	3.0	4.2	1.2	versicolor
se544	5.0	3.5	1.3	0.3	setosa
se543	5.7	4.4	1.5	0.4	setosa

1-10 of 20 rows Previous **1** 2 Next

```
iris_matrix <- as.matrix(iris_test[1:20, 1:5])
iris_matrix
```

```
##        Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## se408  "5.1"        "3.8"       "1.9"        "0.4"       "setosa"
## ve1717 "5.6"        "3.0"       "4.1"        "1.3"       "versicolor"
## ve850  "5.5"        "2.6"       "4.4"        "1.2"       "versicolor"
## se1631 "4.7"        "3.2"       "1.6"        "0.2"       "setosa"
## ve1865 "5.8"        "2.7"       "4.1"        "1.0"       "versicolor"
## se130  "5.1"        "3.5"       "1.4"        "0.3"       "setosa"
## ve1311 "6.0"        "3.4"       "4.5"        "1.6"       "versicolor"
## ve1672 "5.7"        "3.0"       "4.2"        "1.2"       "versicolor"
## se544  "5.0"        "3.5"       "1.3"        "0.3"       "setosa"
## se543  "5.7"        "4.4"       "1.5"        "0.4"       "setosa"
## ve1490 "6.1"        "3.0"       "4.6"        "1.4"       "versicolor"
## vi37   "7.7"        "2.6"       "6.9"        "2.3"       "virginica"
## vi1694 "6.7"        "3.1"       "5.6"        "2.4"       "virginica"
## se8    "5.4"        "3.9"       "1.3"        "0.4"       "setosa"
## ve1210 "5.8"        "2.6"       "4.0"        "1.2"       "versicolor"
## vi672  "6.0"        "3.0"       "4.8"        "1.8"       "virginica"
## vi1442 "6.3"        "3.3"       "6.0"        "2.5"       "virginica"
## ve567  "6.6"        "3.0"       "4.4"        "1.4"       "versicolor"
## vi1011 "7.4"        "2.8"       "6.1"        "1.9"       "virginica"
## se1945 "4.9"        "3.1"       "1.5"        "0.1"       "setosa"
```

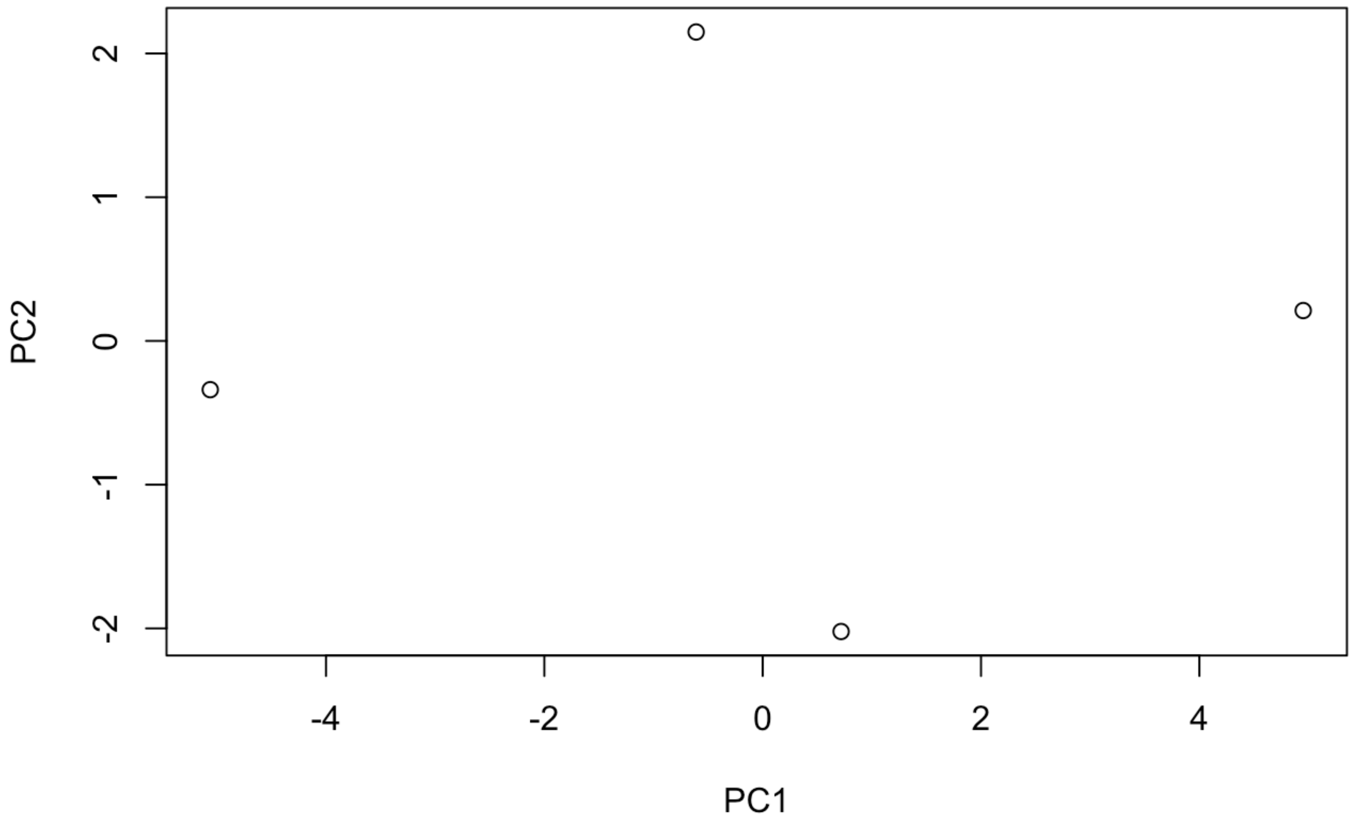
```r
extract_and_convert_matrix <- function(mat) {
  middle_four <- mat[, 1:4]
  middle_four <- apply(middle_four, 2, function(x) as.numeric(as.character(x)))
  return(middle_four)
}

# Pass the matrix to the function
result_matrix <- extract_and_convert_matrix(iris_matrix)
print(result_matrix)
```

```
##       Sepal.Length Sepal.Width Petal.Length Petal.Width
##  [1,]          5.1         3.8          1.9         0.4
##  [2,]          5.6         3.0          4.1         1.3
##  [3,]          5.5         2.6          4.4         1.2
##  [4,]          4.7         3.2          1.6         0.2
##  [5,]          5.8         2.7          4.1         1.0
##  [6,]          5.1         3.5          1.4         0.3
##  [7,]          6.0         3.4          4.5         1.6
##  [8,]          5.7         3.0          4.2         1.2
##  [9,]          5.0         3.5          1.3         0.3
## [10,]          5.7         4.4          1.5         0.4
## [11,]          6.1         3.0          4.6         1.4
## [12,]          7.7         2.6          6.9         2.3
## [13,]          6.7         3.1          5.6         2.4
## [14,]          5.4         3.9          1.3         0.4
## [15,]          5.8         2.6          4.0         1.2
## [16,]          6.0         3.0          4.8         1.8
## [17,]          6.3         3.3          6.0         2.5
## [18,]          6.6         3.0          4.4         1.4
## [19,]          7.4         2.8          6.1         1.9
## [20,]          4.9         3.1          1.5         0.1
```

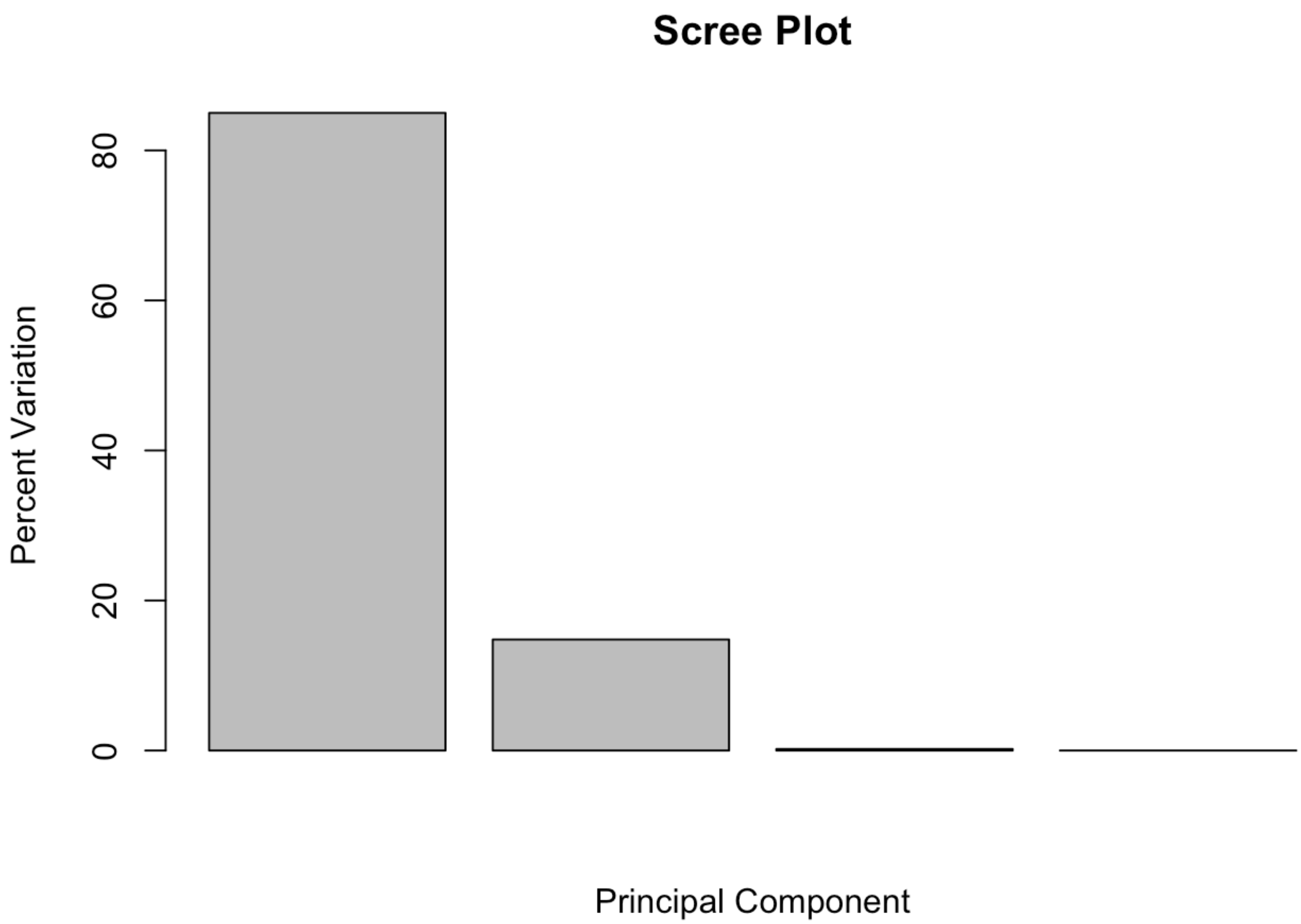
```
pca <- prcomp(t(result_matrix), scale=TRUE)
pca
```

```
## Standard deviations (1, .., p=4):
## [1] 4.123522e+00 1.719234e+00 2.019965e-01 1.053843e-15
##
## Rotation (n x k) = (20 x 4):
##               PC1         PC2         PC3         PC4
##  [1,] -0.2107231 -0.28560380 -0.30817168  0.30313513
##  [2,] -0.2405404  0.06988648 -0.20712872 -0.37113092
##  [3,] -0.2336353  0.15330929 -0.24194794  0.17869170
##  [4,] -0.2158382 -0.26498198 -0.09066822  0.17615669
##  [5,] -0.2396897  0.08832806 -0.04230031  0.01390490
##  [6,] -0.2072234 -0.30157289  0.15893279 -0.09175166
##  [7,] -0.2404977  0.06706676 -0.28186140 -0.02043216
##  [8,] -0.2400592  0.07604282 -0.27239928  0.34056723
##  [9,] -0.2039711 -0.31407180  0.15856709 -0.20743066
## [10,] -0.1956041 -0.34373234 -0.07054998 -0.16625044
## [11,] -0.2377513  0.11403609 -0.10287254 -0.07225620
## [12,] -0.2078194  0.29773955  0.29765901 -0.23870726
## [13,] -0.2236814  0.22206726  0.29292281  0.03524217
## [14,] -0.1982759 -0.33418254  0.18816463 -0.14661853
## [15,] -0.2390500  0.09338199  0.25075613  0.14781019
## [16,] -0.2335009  0.15706580 -0.01524706 -0.23748686
## [17,] -0.2135240  0.27282795 -0.34150167 -0.31549350
## [18,] -0.2399485  0.07416740  0.34166732  0.01790097
## [19,] -0.2224719  0.22989614  0.23313064  0.48856918
## [20,] -0.2178272 -0.25530910  0.11548634  0.09990850
```

```
plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab = "PC2")
```




```
# make a scree plot
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)

barplot(pca.var.per, main="Scree Plot", xlab="Principal Component", ylab="Percent Var
iation")

library(ggplot2)
```

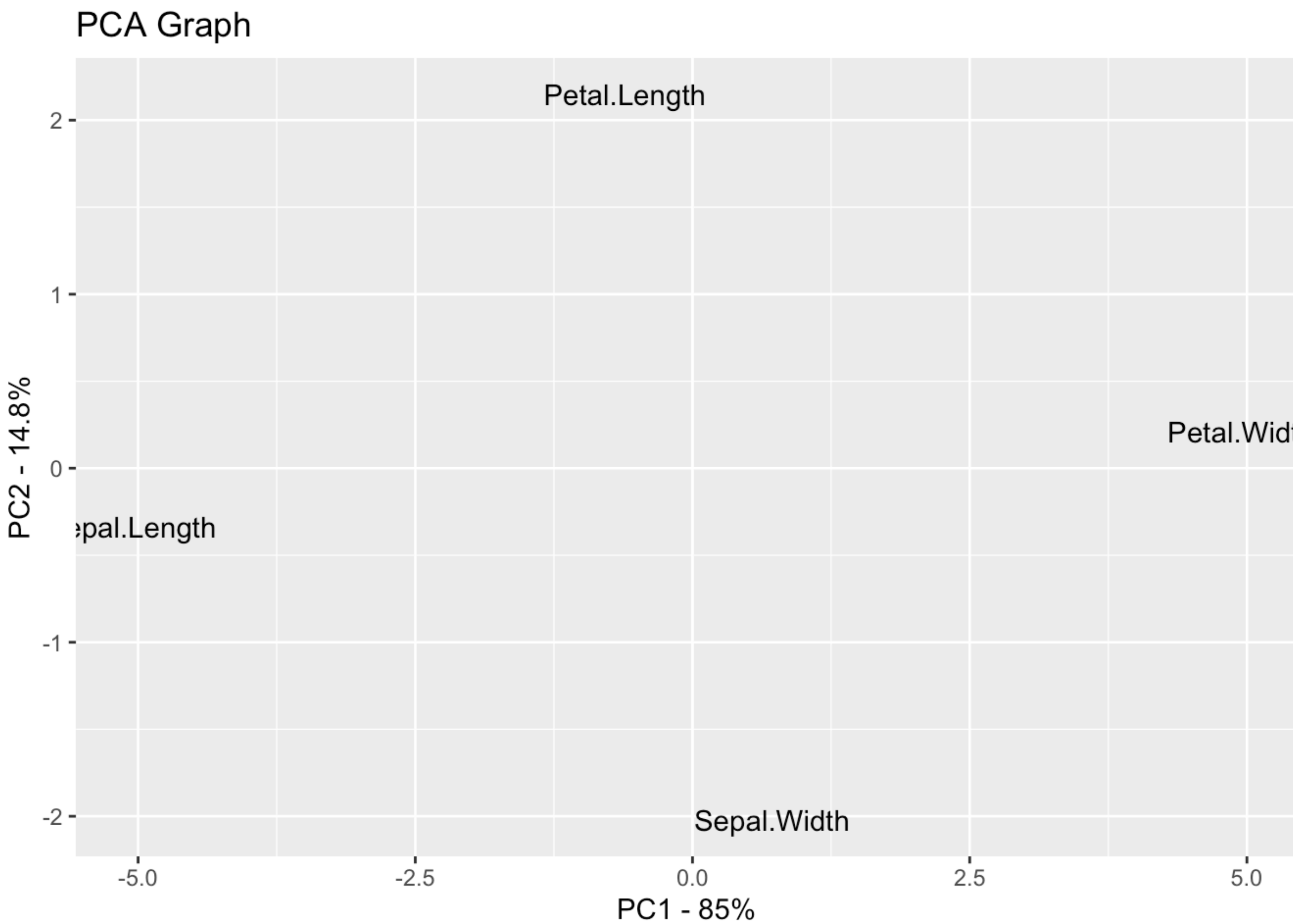



```r
library(ggthemes)

pca.data <- data.frame(Sample=rownames(pca$x),
                       X=pca$x[,1],
                       Y=pca$x[,2])
pca.data
```

	Sample	X	Y
	<chr>	<dbl>	<dbl>
Sepal.Length	Sepal.Length	-5.0602217	-0.3391065
Sepal.Width	Sepal.Width	0.7183045	-2.0217211
Petal.Length	Petal.Length	-0.6095419	2.1494843
Petal.Width	Petal.Width	4.9514592	0.2113433

4 rows

```
ggplot(data=pca.data, aes(x=X, y=Y, label=Sample)) +
  geom_text() +
  xlab(paste("PC1 - ", pca.var.per[1], "%", sep="")) +
  ylab(paste("PC2 - ", pca.var.per[2], "%", sep="")) +
  ggtitle("PCA Graph")
```



Problem 3

Using the dataset attached to the assignment, honey.csv, first split the data as 60% training data and 40% validation data, using the TVHSplit function created in the notes (keep iseed as the default value so that the sampling is consistent). Train 3 models using average_price as your y variable: 1. A decision tree using the rpart package 2. A model based recursive partitioning tree model from the partykit package 3. A boosted tree from the gbm package Compare the validation R2 for each model

```
library(rpart)
library(partykit)
```

```
## Loading required package: grid
```

```
## Loading required package: libcoin
```

```
## Loading required package: mvtnorm
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

```
data_honey <- read.csv('Honey.csv')
head(data_honey)
```

	colonies_number <int>	yield_per_colony <int>	stocks <int>	average_price <dbl>
1	16000	58	28000	62
2	52000	79	986000	68
3	50000	60	900000	64
4	420000	93	4687000	60
5	45000	60	1404000	68
6	230000	86	1780000	63

6 rows

```
str(data_honey)
```

```
## 'data.frame':    1115 obs. of  4 variables:
##  $ colonies_number : int  16000 52000 50000 420000 45000 230000 70000 8000 125000
11000 ...
##  $ yield_per_colony: int  58 79 60 93 60 86 62 129 48 74 ...
##  $ stocks          : int  28000 986000 900000 4687000 1404000 1780000 260000 10300
0 1020000 212000 ...
##  $ average_price   : num  62 68 64 60 68 63 69 55 65 102 ...
```

```
summary(data_honey)
```

```
##  colonies_number  yield_per_colony     stocks          average_price
##  Min.   :  2000   Min.   : 19.00   Min.   :    9000   Min.   :  1.3
##  1st Qu.:  9000   1st Qu.: 45.00   1st Qu.:  112500   1st Qu.: 70.0
##  Median : 26000   Median : 57.00   Median :  370000   Median :128.0
##  Mean   : 62439   Mean   : 59.74   Mean   : 1172625   Mean   :140.6
##  3rd Qu.: 69000   3rd Qu.: 71.00   3rd Qu.: 1253500   3rd Qu.:193.0
##  Max.   :550000   Max.   :155.00   Max.   :13545000   Max.   :874.0
```

```
TVHsplit = function(df, split = c(0.5, 0.25, 0.25), labels=c('T', 'V', 'H'), iseed=39
7){
  set.seed(iseed)
  flags = sample(labels, size = nrow(df),
                 prob = split, replace = T)
  return(flags)
}
```

```
split_flags <- TVHsplit(data_honey, split = c(0.6, 0.4), labels=c('T', 'V'))

train_data = data_honey[which(split_flags == 'T'),]
valid_data = data_honey[which(split_flags == 'V'),]

# Decision tree using rpart
tree_model_rpart <- rpart(average_price ~ ., data = train_data)

# Model based recursive partitioning tree model from the partykit package
tree_model_partykit <- ctree(average_price ~ ., data = train_data)

# A boosted tree from the gbm package
gbm_model <- gbm(average_price ~ ., data = train_data, distribution = "gaussian", n.t
rees = 100, interaction.depth = 3)

# Comparing models by R-squared
ValidationRsq = function (validObs, validHat){
  resids = validHat - validObs
  yBar = mean(validObs)
  offset = validObs - yBar
  num = sum(resids^2)
  denom = sum(offset^2)
  Rsq = 1 - num/denom
  return(Rsq)
}


# Compare the validation R2 for each model
rpartHoneyHatV = predict(tree_model_rpart, newdata = valid_data)
ValidationRsq(valid_data$average_price, rpartHoneyHatV)
```

```
## [1] 0.07977974
```

```
treepartyHatV = predict(tree_model_rpart, newdata = valid_data)
ValidationRsq(valid_data$average_price, treepartyHatV)
```

```
## [1] 0.07977974
```

```
gmbmodelHatV = predict(gbm_model, newdata = valid_data)
```

```
## Using 100 trees...
```

```
ValidationRsq(valid_data$average_price, gmbmodelHatV)
```

```
## [1] 0.08430047
```