The University of British Columbia

Irving K. Barber School of Sciences

DATA 101

Lab 3 Solution

Date: September 16–20, 2019

In each question below, write out the required lines of R code.

1. Given the vector `colors`:

```
[1] "red"     "yellow"  "blue"    "green"
[5] "magenta" "cyan"
```

   (a) use an appropriate subsetting method to create the vector

```
[1] "yellow"  "blue"  "magenta"
```

```
colors <- c("red", "yellow", "blue", "green", "magenta", "cyan")
  # colors is assumed to be in the workspace already
colors[c(2, 3, 5)]
```

   (b) use `rep()` and `seq()` to create the vector

```
[1] "red"     "yellow"  "blue"    "yellow"  "blue"
[6] "green"   "blue"  "green"   "magenta" "green"
[11] "magenta" "cyan"
```

```
colors[rep(1:3,4)+rep(0:3,rep(3,4))]
```

2. You can create a matrix of data on the girth, height and volume of a sample of trees by typing the following:

```
Trees <- as.matrix(trees) # trees is built-in to R
```

   (a) Find the numbers of rows and columns of the matrix `Trees`.

```
nrow(Trees)
ncol(Trees) # or dim(Trees)
```

   (b) What are the elements of the 15th row of `Trees`?

```
Trees[15, ]
```

   (c) What are the elements of the 1st column of `Trees`?

```
Trees[, 1]
```

   (d) Find the sum of the 2nd column of `Trees`.

```
sum(Trees[, 2])
```

(e) Find the sample variance of the 1st 15 elements of the 2nd column of `Trees`.

```
var(Trees[1:15, 2])
```

3. Load the *MASS* package into R. (It is automatically installed when you install R, so you just need to load it using the `library()` function). Check the help file on `bacteria` and identify the country where the data were originally collected.

```
library(MASS)
help(bacteria)   # country of origin of data: Australia
```

4. Check the help file on `faithful` as well as `example(faithful)`.

   (a) Briefly describe what was measured.

   (b) Does waiting time to the next eruption tend to increase or decrease as a function of eruption time?

   (c) Find the mean of the eruption times.

```
help(faithful)
example(faithful)   # waiting time increases with eruption time
mean(faithful[, 1])
```

5. Install the *DAAG* package as follows:

```
install.packages("DAAG")
```

Load this package as follows:

```
library(DAAG)
```

Check the help file on the `cfseal` object. How did the Cape Fur Seals whose measurements are in the data set die?

```
# as a consequence of commercial fishing
```

6. Continue to refer to the `cfseal` object.

   (a) How many rows and columns are in `cfseal`?

```
nrow(cfseal)
ncol(cfseal)  # or dim(cfseal)
```

   (b) How many elements of the 11th column are missing values?

```
sum(is.na(cfseal[, 11]))
```

(c) Calculate the mean of the non-missing values in the 11th column, and assign this value to `cfs.avg`.

```
cfs.avg <- mean(na.omit(cfseal[, 11]))
```

(d) Replace the missing values in the 11th column of `cfseal` by the value in `cfs.avg`. This is one method of *imputation* of missing values.

```
cfseal[is.na(cfseal[, 11]), 11] <- cfs.avg
```

(e) Subtract the values of the 7th column from the newly completed 11th column and assign the result to a vector called `diff711`.

```
diff711 <- cfseal[, 7] - cfseal[, 11]
```

(f) Plot a histogram of the values in `diff711`. How many of the values in this vector are negative?

```
hist(diff711)
```

7. Type the following lines into an R session:

```
set.seed(31186)
x <- runif(1000); y <- runif(1000)
```

(a) How many entries of `x` are greater than 0.9?

```
sum(x > 0.9)

## [1] 93
```

(b) How many entries of `x` are greater than the corresponding entries of `y`?

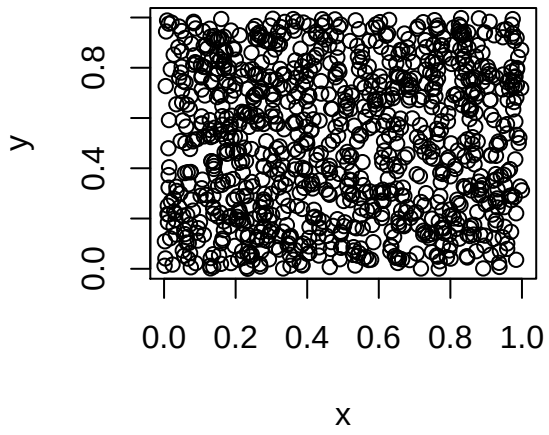
```
sum(x > y)

## [1] 484
```

(c) Calculate the interquartile range for `x` and `x + y` and for `x - y`.

```
IQR(x)

## [1] 0.5176714

IQR(x+y)

## [1] 0.606872

IQR(x-y)

## [1] 0.5574096
```

(d) Construct a scatterplot of `y` against `x`. Do you see any obvious patterns? In particular, would you be able to predict a value of `y` from a corresponding value of `x`?

```
plot(y ~ x)
```



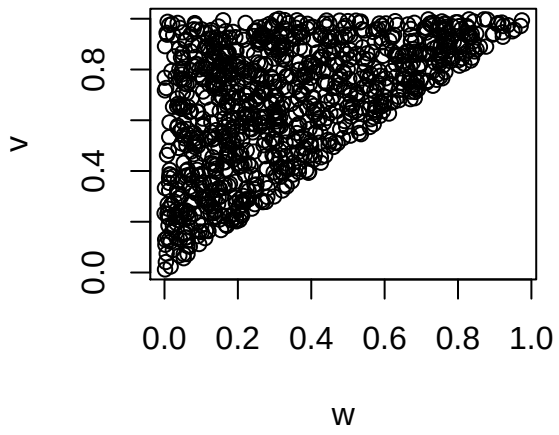

```
  # no obvious patterns; you cannot predict values of y from
  #  corresponding x values.
```

(e) Find the pairwise minima of `x` and `y` and assign the result to `w`. Assign pairwise maxima of `x` and `y` to `v`.

```
w <- pmin(x, y)
v <- pmax(x,y)
```

(f) Construct a scatterplot of `v` against `w`. Do you see any obvious patterns? In particular, would you be able to predict a value of `v` from a corresponding value of `w`?

```
plot(v ~ w)
```




```
  # now there is a pattern; you can sometimes predict values of v
  # from the corresponding w values, with better precision as w
  # increases.
```

8. (a) Use the `rep()` and `seq()` functions in R to obtain the following patterned data vectors:

```
 [1] 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 6 6
[33] 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8
```

```
rep(1:8, 3:10)

##  [1] 1 1 1 2 2 2 2 3 3 3 3 3 4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6
## [36] 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8
```

(b) Withthe result of previous part, with the `factor()`, `levels()`, and `as.character()` functions to create the vector:

```
 [1] "Toronto"   "Toronto"   "Toronto"   "Toronto"   "Montreal"  "Montreal
 [7] "Montreal"  "Montreal"  "Montreal"  "Vancouver" "Vancouver" "Vancouve
[13] "Vancouver" "Vancouver" "Vancouver" "Calgary"   "Calgary"   "Calgary"
[19] "Calgary"   "Calgary"   "Calgary"   "Calgary"   "Edmonton"  "Edmonton
[25] "Edmonton"  "Edmonton"  "Edmonton"  "Edmonton"  "Edmonton"  "Edmonton
[31] "Ottawa"    "Ottawa"    "Ottawa"    "Ottawa"    "Ottawa"    "Ottawa"
[37] "Ottawa"    "Ottawa"    "Ottawa"    "Quebec"    "Quebec"    "Quebec"
[43] "Quebec"    "Quebec"    "Quebec"    "Quebec"    "Quebec"    "Quebec"
[49] "Quebec"    "Winnipeg"  "Winnipeg"  "Winnipeg"  "Winnipeg"  "Winnipeg
[55] "Winnipeg"  "Winnipeg"  "Winnipeg"  "Winnipeg"  "Winnipeg"  "Winnipeg
```

You will need the vector

```
Cities <- c("Toronto", "Montreal", "Vancouver", "Calgary",
            "Edmonton", "Ottawa", "Quebec", "Winnipeg")
```

```r
Citiesfactor <- factor(rep(1:8,4:11))
Cities <- c("Toronto", "Montreal", "Vancouver", "Calgary",
            "Edmonton", "Ottawa", "Quebec", "Winnipeg")
levels(Citiesfactor) <- Cities
as.character(Citiesfactor)
```

```
##  [1] "Toronto"   "Toronto"   "Toronto"   "Toronto"   "Montreal"
##  [6] "Montreal"  "Montreal"  "Montreal"  "Vancouver" "Vancouver"
## [11] "Vancouver" "Vancouver" "Vancouver" "Vancouver" "Calgary"
## [16] "Calgary"   "Calgary"   "Calgary"   "Calgary"   "Calgary"
## [21] "Calgary"   "Calgary"   "Edmonton"  "Edmonton"  "Edmonton"
## [26] "Edmonton"  "Edmonton"  "Edmonton"  "Edmonton"  "Edmonton"
## [31] "Ottawa"    "Ottawa"    "Ottawa"    "Ottawa"    "Ottawa"
## [36] "Ottawa"    "Ottawa"    "Ottawa"    "Ottawa"    "Quebec"
## [41] "Quebec"    "Quebec"    "Quebec"    "Quebec"    "Quebec"
## [46] "Quebec"    "Quebec"    "Quebec"    "Quebec"    "Winnipeg"
## [51] "Winnipeg"  "Winnipeg"  "Winnipeg"  "Winnipeg"  "Winnipeg"
## [56] "Winnipeg"  "Winnipeg"  "Winnipeg"  "Winnipeg"  "Winnipeg"
```