Week 1 Question Sheet

This question sheet covers material from Week 1 of MM916: Data Analytics in R. There may be some things in this sheet that have not specifically been covered in lectures. Just remember that you are always allowed to make use of the help files in R and please feel free to ask us for help, in person, on the Q&A forum, or by email.

When starting this lab please make sure to open a new R Script so that you don't lose any work.

Question 1

(a) Create an object `x` which contains the value 534 and an object `y` which contains the value 23.897.

```
x <- 534
y <- 23.897
```

(b) Create a third object that contains the product of `x` and `y` (you may name this object anything).

```
xy <- x*y
```

(c) Calculate $3x^2(x + y)$.

```
3 * x^2 * (x+y)
```

```
## [1] 477263031
```

(d) Create a vector called `age` that takes the values 13, 17, 21, 25, 29, 33, 37, 41, 45, 49. Can you do this without typing out all the values?

```
# you could just type out all of the values if you wanted
age <- c(13,17,25,29,33,37,41,45,49)

# or if you noticed that there was 4 between each of the values
# you could save time by using the seq() function
age <- seq(13, 49, by=4)
```

(e) Create a vector called `msg_length` that takes the values 26, 30, 33, 34, 40, 41, 47, 50, 55, 65

```
# There is no common difference here so we have to type the value out
msg_length <- c(26,30,33,34,40,41,47,50,55,65)
```

(f) Create a vector called `selected_ages` that contains the values of `age` for which the corresponding value of `msg_length` is greater than 49, and a variable called `num_selected` that contains the number of elements in `selected_ages`. (Find a way to do that doesn't require going through the values by hand: make R do the analysis.)

```
selected_ages = age[msg_length > 49]
selected_ages
```

```
## [1] 41 45 49
```

```
num_selected = length(selected_ages)
num_selected
```

```
## [1] 3
```

Question 2

(a) Create the matrix shown below:

$$\begin{bmatrix} 1 & 7 & 33 \\ 7 & 1 & 6 \\ 0.03 & 0.65 & 1 \end{bmatrix}$$

```
matrix(c(1,7,33, 7,1,6,0.03, 0.65,1), nrow=3, byrow=TRUE)
```

```
##      [,1] [,2] [,3]
## [1,] 1.00 7.00   33
## [2,] 7.00 1.00    6
## [3,] 0.03 0.65    1
```

```
# You could also change the order of the values in the vector
# and use bycol=TRUE.
```

(b) Create a data frame with columns containing the `age` vector and `msg_length` vector from Question 1. Extract the *rows* of the data frame where `age` is less than or equal to 21.

```
# You could leave the columns unnamed here
# since the variable names were meaningful
msg_data <- data.frame(age, msg_length)
msg_data[age <= 21,]
```

```
##   age msg_length
## 1  13         26
## 2  17         30
## 3  21         33
```

(c) Construct a list that consists of 3 elements:

- `fruit` - contains the words apples, oranges, pears, bananas
- `count` - contains integer values from 12 to 20
- `data` - contains the data frame created in part (b)

```
random <- list(fruit = c("apples","oranges","pears","bananas"),
               # Can use : to specify ever integer between 12 and 20
               count = 12:20,
               data = msg_data)
```

(d) Create a data frame in R called `salmon` that contains the following values:

weight	farm
1.5	A
1.2	A
1.9	A
0.9	A
1.8	A
1.1	A
1.4	B
0.7	B
1.3	B
1.3	B
1.5	B
1.3	B
1.2	C
2.4	C
2.4	C
2.0	C
2.2	C
2.3	C

You might find the `rep()` function useful here.

```
salmon <- data.frame(weight=c(1.5, 1.2, 1.9, 0.9, 1.8, 1.1, 1.4, 0.7, 1.3, 1.3,
                              1.5, 1.3, 1.2, 2.4, 2.4, 2.0, 2.2, 2.3),
                     farm = rep(c("A","B","C"), each=6))
```

(e) Create three new data frames that contain data from farm A, farm B and farm C separately. You should use conditional statements to do this!

```
# when checking for equality you need to use two equals signs

farmA <- salmon[salmon$farm=="A",]

farmB <- salmon[salmon$farm=="B",]

farmC <- salmon[salmon$farm=="C",]
```

Question 3

(a) Install the package `palmerpenguins`, and load the data frame `penguins` that it contains. (For information about what this dataset is, see https://allisonhorst.github.io/palmerpenguins/ .)

```
install.packages("palmerpenguins")
require(palmerpenguins)
data(penguins)
```

(b) List the species that are included, using the `unique()` function.

```
unique(penguins$species)
```

```
## [1] Adelie    Gentoo    Chinstrap
## Levels: Adelie Chinstrap Gentoo
```

(c) Which species has the highest average body mass?

```
mean(penguins$body_mass_g[penguins$species=="Adelie"],na.rm=TRUE)
```

```
## [1] 3700.662
```

```
mean(penguins$body_mass_g[penguins$species=="Chinstrap"],na.rm=TRUE)
```

```
## [1] 3733.088
```

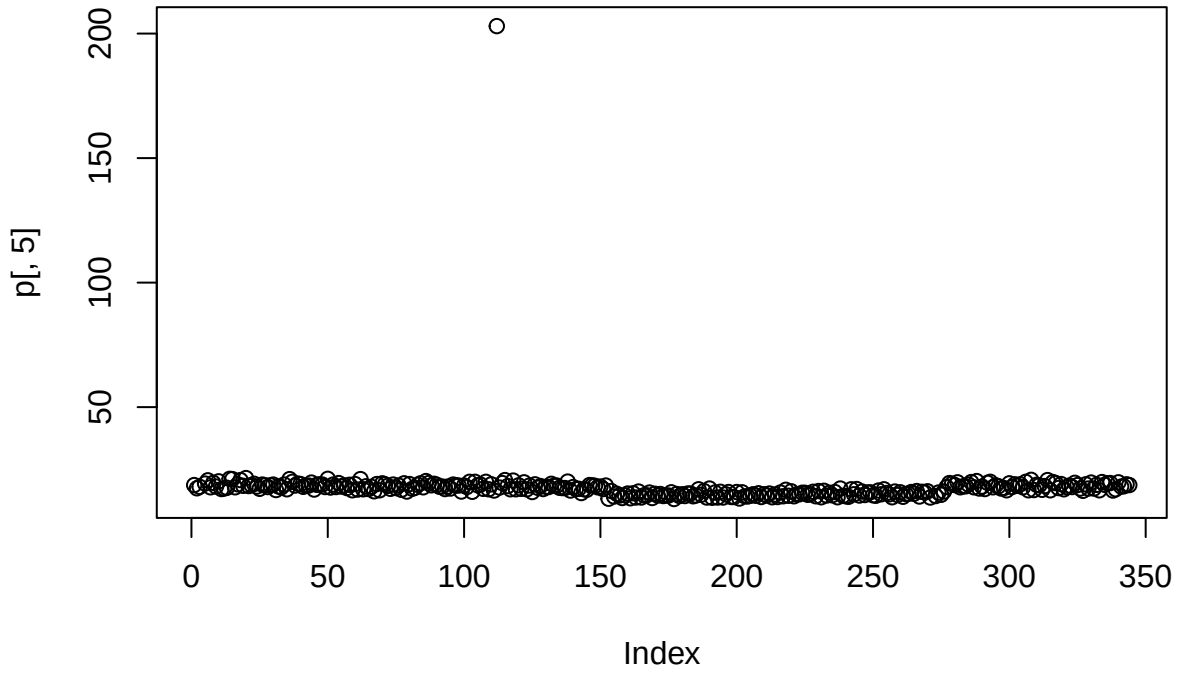
```
mean(penguins$body_mass_g[penguins$species=="Gentoo"],na.rm=TRUE)
```

```
## [1] 5076.016
```

```
# Gentoo penguins are the biggest.
```

(d) The .csv file `penguins_messy.csv` contains a version of this data with a few typing mistakes. Load it using the `read.csv` function, and find the errors. (Hint: use `hist()` or `plot()` to make simple plots of the numeric data and look for outliers.)

```
p <- read.csv('penguins_messy.csv',header=TRUE)
# for each numeric column, I did the following (column 5 used as an example):
# make a simple plot of one column of data
plot(p[,5])
```



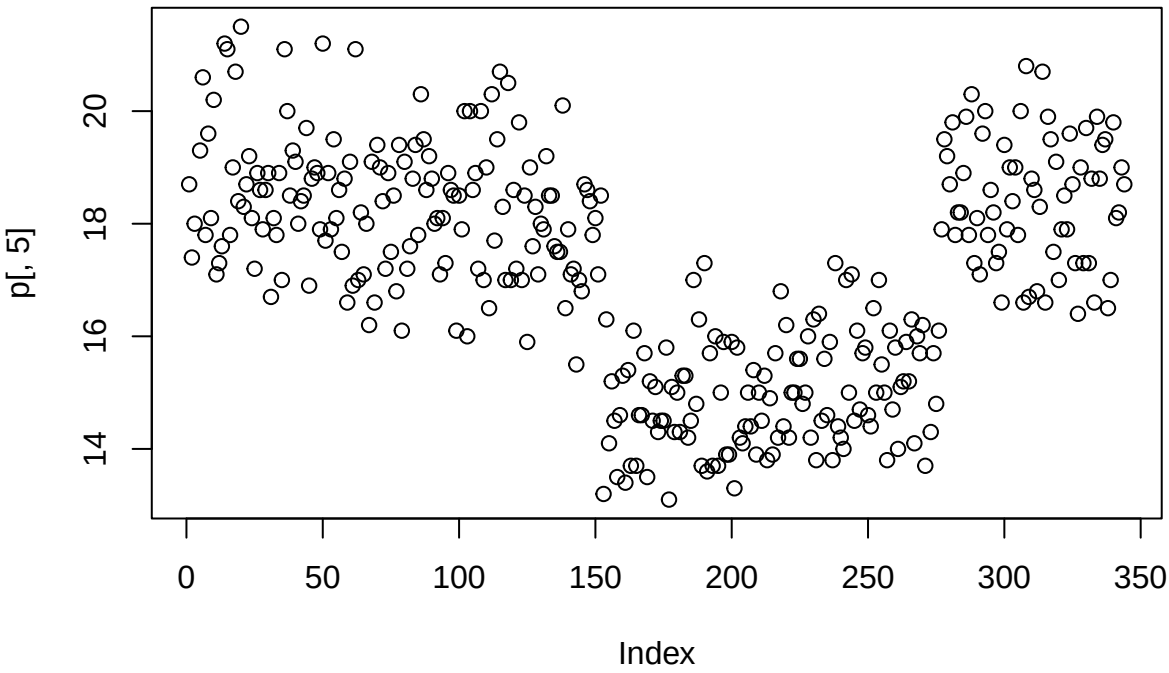

```
# there's one value near 200 that's clearly not like the others. We can locate it
# like this:
which(p[,5] > 150)
```

```
## [1] 112
```

```
# so it's on row 112. Let's see what the actual value is:
p[112,5]
```

```
## [1] 203
```

```
# probably there's a missing decimal point. (If we weren't confident about how to
# fix the bad value we could replace it with NA)
p[112,5] <- 20.3
# now plot it again to see if the outliers are gone:
plot(p[,5])
```




```
# You certainly can't locate all typing errors this way, but you can locate obvious ones!
```

(e) This is a good moment to investigate the options for reading text files. Try loading `penguins_messy.csv` with

- `read.csv`
- `read.table`
- `readr::read_csv2`

and with alternate choices for a couple of input options:

- `header=TRUE`
- `header=FALSE`

and

- `sep = ','`
- `sep = ';'`

You don't have to try all the combinations–just enough to form a theory about what each of these choices does. Which work best and worst for this file, and why?

```
penguintest <- read.csv('penguins_messy.csv',header=TRUE)
# works pretty well

penguintest <- read.csv('penguins_messy.csv',header=FALSE)
# turns the column names into a first row of data: not good

penguintest <- read.table('penguins_messy.csv',header=TRUE,sep=',')
# this also works pretty well, but only when we specify that the separator
# between columns is a comma. Otherwise it smashes each line of the file into
# one long character value.

require(readr)
```

```
penguintest <- readr::read_csv2('penguins_messy.csv',header=TRUE)
# very similar to read.csv, but the punctuation in the column names is different.
```

Question 4

Sandeels are really important food for seabirds in the North Sea and beyond: search "puffin sandeel" for cute photos of parent puffins carrying mouthfuls home to their chicks. But sandeels have been getting smaller because of climate change, and this has been causing long-term declines in many of the UK's seabirds.

(a) Create a vector called `numFish` that contains the values 9, 18, 20, 23, 16, 28, 19, 7, and a vector called `fishLength` that contains 25, 35, 45, ... 95. Each element of `numFish` represents a number of sandeels found within a certain length range by a fisheries sampling programme: 20–29 mm for the first element, 30–39 mm for the second, and so on. `fishLength` is a vector that represents the midpoint of these bins.

```
numFish <- c(9, 18, 20, 23, 16, 28, 19, 7)
fishLength <- seq(25,95,10)
```

(b) Extract the 4th, 7th and 8th elements from `numFish`

```
numFish[c(4,7,8)]
```

```
## [1] 23 19  7
```

(c) Calculate the proportion of the total sample of sandeels in each length category. You may find it useful to investigate the `sum()` function.

```
# Use the sum function find find the total sample size
# divide the individual values in frequency by the sample size
p <- numFish/sum(numFish)
```

(d) Use these proportions to calculate the overall mean length ($\bar{x}$) and variance (s^2) using the formulae below:

$$\bar{x} = \sum_{i=1}^{n} p_i x_i$$

$$s^2 = \sum_{i=1}^{n} p_i (x_i - \bar{x})^2$$

Here x_i is the length and p_i the proportion of fish of that length.

```
# x in the formulas is equivalent to the vector fishLength in our code. Let's rename it,
# so that the R code looks as much like the formulas as possible.
x = fishLength
# p in the formulas is equivalent to the vector created in part c), which I already called p

# mean length
xbar <- sum(p*x)

# length variance
s_squared <- sum(p*(x-xbar)^2)
```