# Homework2

*Ben Lane*

*9/26/2017*

*(informally) Due Tuesday, 19 September, 1:00 PM*

50 points total.

This assignment won't be submitted until we've covered Rmarkdown. Create R chunks for each question and insert your R code appropriately. Check your output by using the `Knit PDF` button in RStudio.

1. **Working with data** In the `datasets` folder on the course GitHub repo, you will find a file called `cancer.csv`, which is a dataset in comma-separated values (csv) format. This is a large cancer incidence dataset that summarizes the incidence of different cancers for various subgroups. (18 points)

    1. Load the data set into R and make it a data frame called `cancer.df`. (2 points)

```
  #getwd()
#setwd("Desktop")
#cancer.df <- read.csv("../Desktop/cancer.csv", header = T, sep = ",")
#cancer.df <- read.csv("cancer.csv")
```

    2. Determine the number of rows and columns in the data frame. (2)

```
#str(cancer.df)
#There are 42120 rows and 8 variables
```

    3. Extract the names of the columns in `cancer.df`. (2)

```
  # names <- colnames(cancer.df)
  # names
```

    4. Report the value of the 3000th row in column 6. (2)

```
#  cancer.df[3000,6]
```

    5. Report the contents of the 172nd row. (2)

```
#  cancer.df[172, ]
```

    6. Create a new column that is the incidence *rate* (per 100,000) for each row.(3)

```
  # rate <- cancer.df[,7]/100000
```

    7. How many subgroups (rows) have a zero incidence rate? (2)

```
#sum(rate == 0)
```

    8. Find the subgroup with the highest incidence rate.(3)

```
  # which(rate == max(rate))
```

2. **Data types** (10 points)

    1. Create the following vector: `x <- c("5","12","7")`. Which of the following commands will produce an error message? For each command, Either explain why they should be errors, or explain the non-erroneous result. (4 points)

max(x): this returns a value of 7. I think what's going on is that R is coercing each string to a specific numeric value, which it then takes the max of. It happens that the string '7' has the maximum coerced

value, but it is obviously not the max value of the vector if each string were converted to an integer, e.g. '7' becomes 7, '12' becomes 12, etc.

sort(x): this returns the values incorrectly sorted. I think it's doing the same thing as before. It's sorting these not based on their integer values, but instead based on the value that results from the string being coerced into a numeric.

sum(x): this simply returns an error because the function sum() cannot use a character data type.

2. For the next two commands, either explain their results, or why they should produce errors. (3 points

y <- c("5",7,12) y[2] + y[3]

This produces an error because when we save the variable y, the 7 and 12 are coerced into strings so that is a character vector. So, when we type y[2] + y[3], we are not writing 7 + 12, but rather '7' + '12', which is nonsensical because we cannot add strings.

3. For the next two commands, either explain their results, or why they should produce errors. (3 points

```
z <- data.frame(z1="5",z2=7,z3=12)
z[1,2] + z[1,3]
```

This is fine because when it was saved as a data.frame, R coerced the value of "5" into a factor, which means that it uses 5 as an integer value in the data.frame for calculations but displays it as the character value "5".

3. **Data structures** Give R expressions that return the following matrices and vectors (*i.e.* do not construct them manually). (3 points each, 12 total)

    1. $(1, 2, 3, 4, 5, 6, 7, 8, 7, 6, 5, 4, 3, 2, 1)$

```
c(seq(1,8,1),seq(7,1,-1))
```

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

2. $(1,2,2,3,3,3,4,4,4,4,5,5,5,5,5)$

There has to be a better way to do this. If I wanted to do this for 1000 numbers, what's the best way? I literally spent 45 minutes sitting here trying to figure out how to do this more efficiently.

```
x <- c(1, rep(2,2), rep(3,3), rep(4,4), rep(5,5))
x
```

```
##  [1] 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5
```

3. $\begin{pmatrix}
  0 & 1 & 1  \\
  1 & 0 & 1  \\
  1 & 1 & 0  \\
\end{pmatrix}$

```
x <- rep(1,9)
mymatrix <- matrix(x, nrow = 3, ncol = 3)
for (i in 1:3){mymatrix[i,i] <- 0}
mymatrix
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    1
## [2,]    1    0    1
## [3,]    1    1    0
```

4. $\begin{pmatrix}
  1 & 2 & 3 & 4 \\
  1 & 4 & 9 & 16 \\

```
  1 & 8 & 27 & 64  \\
  1 & 16 & 81 & 256 \\
  1 & 32 & 243 & 1024  \\
\end{pmatrix}$
```

```
x <- numeric(20)
mymatrix <- matrix(x, nrow = 5, ncol = 4)
for (i in 1:4){
  mymatrix[,i]<- i}
for (i in 1:4){
  mymatrix[,i] <- mymatrix[,i]^seq(1:5)
}
mymatrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    1    4    9   16
## [3,]    1    8   27   64
## [4,]    1   16   81  256
## [5,]    1   32  243 1024
```

4. **Basic programming** (10 points)

    1. Let $h(x,n) = 1 + x + x^2 + \ldots + x^n = \sum_{i=0}^{n} x^i$. Write an R program to calculate $h(x,n)$ using a `for` loop. (5 points)

```
myfunction <- function(x,n){
  y <- numeric(n)
  for (i in 1:n){
    y[i] <- x^i
  }
  return(sum(y)+1)
}
myfunction(2,3)
```

```
## [1] 15
```

```
1. If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The s
```

```
#Euler 1:

x <- 1:(1000-1)
y <- c(which(x%%3 == 0), which(x%%5 == 0))
z <- unique(y)
sum(z)
```

```
## [1] 233168
```

```
#Euler 2:

x <- 1:(1e6-1)
y <- c(which(x%%4 == 0), which(x%%7 == 0))
z<- unique(y)
#This returns an overflow error sum(z)

#Euler 3
y <- numeric(15)
y[1] <- 1
```

```
y[2] <- 1
for (i in 3:15){
  y[i]<- y[i-1] + y[abs(i-2)]
}
y
```

```
##  [1]   1   1   2   3   5   8  13  21  34  55  89 144 233 377 610
```

```
length(y)
```

```
## [1] 15
```

```
sum(y[which(y%%2 == 0)])
```

```
## [1] 798
```

1. Find the sum of all the multiples of 3 or 5 below 1,000. (3, [euler1])

1. Find the sum of all the multiples of 4 or 7 below 1,000,000. (2)

1. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting w

Some problems taken or inspired by projecteuler.