# Bios 6301: Assignment 2

*Andrea Perreault*

*(informally) Due Tuesday, 20 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.

**QUESTION 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)

```
cancer.df <- read.csv("https://github.com/fonnesbeck/Bios6301/raw/master/datasets/cancer.csv", header=TR
head(cancer.df)
```

```
##   year                           site   state    sex     race mortality
## 1 1999 Brain and Other Nervous System alabama Female    Black      0.00
## 2 1999 Brain and Other Nervous System alabama Female Hispanic      0.00
## 3 1999 Brain and Other Nervous System alabama Female    White     83.67
## 4 1999 Brain and Other Nervous System alabama   Male    Black      0.00
## 5 1999 Brain and Other Nervous System alabama   Male Hispanic      0.00
## 6 1999 Brain and Other Nervous System alabama   Male    White    103.66
##   incidence population
## 1        19     623475
## 2         0      28101
## 3       110    1640665
## 4        18     539198
## 5         0      37082
## 6       145    1570643
```

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

```
nrow(cancer.df)
```

```
## [1] 42120
```

```
ncol(cancer.df)
```

```
## [1] 8
```

```
The cancer.df data frame has 42,120 rows and 8 columns.
```

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

```
names(cancer.df)
```

```
## [1] "year"       "site"       "state"      "sex"        "race"
## [6] "mortality"  "incidence"  "population"
```

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

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

```
## [1] 350.69
```

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

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

```
##     year                           site  state  sex  race mortality
## 172 1999 Brain and Other Nervous System nevada Male Black         0
##     incidence population
## 172         0      73172
```

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

```
cancer.df[,'incidenceRate'] <- cancer.df[,'incidence']/100000
head(cancer.df)
```

```
##   year                           site   state    sex     race mortality
## 1 1999 Brain and Other Nervous System alabama Female    Black      0.00
## 2 1999 Brain and Other Nervous System alabama Female Hispanic      0.00
## 3 1999 Brain and Other Nervous System alabama Female    White     83.67
## 4 1999 Brain and Other Nervous System alabama   Male    Black      0.00
## 5 1999 Brain and Other Nervous System alabama   Male Hispanic      0.00
## 6 1999 Brain and Other Nervous System alabama   Male    White    103.66
##   incidence population incidenceRate
## 1        19     623475       0.00019
## 2         0      28101       0.00000
## 3       110    1640665       0.00110
## 4        18     539198       0.00018
## 5         0      37082       0.00000
## 6       145    1570643       0.00145
```

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

```
zeroIR <- cancer.df[cancer.df$incidenceRate==0, ]
nrow(zeroIR)
```

```
## [1] 23191
```

```
A total of 23,191 rows have a zero incidence rate.
```

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

```
maxIR <- cancer.df[which.max(cancer.df$incidenceRate), ]
maxIR
```

```
##       year   site      state    sex  race mortality incidence population
## 21387 2002 Breast california Female White   3463.74     18774   13690681
##       incidenceRate
## 21387       0.18774
```

**QUESTION 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) sort(x) sum(x)

```
x <- c("5","12","7")
```

```
The command max(x) returns "7", which is not the max numeric number in the vector.
But it's place in the vector is the max location.
The command sort(x) returns "12" "5" "7". This order is not increasing numerically because
of the 1 character in the 12 element.
The command sum(x) returns an error because the entries in the vector
are characters, not numbers.
```

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]

```
The first command 'y <- c("5",7,12)' results in a vector with the entries of 5, 7, and 12.
There are no displayed errors, but it's important to see that the elements are characters.
When attempting to add elements with the command 'y[2] + y[3]', the output is a non-numeric
argument error. Since the first entry in the vector is a character, the others are coerced
into this data type. Therefore, you cannot add numbers in charater data type.
```

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]

```
The first command 'z <- data.frame(z1="5",z2=7,z3=12)' results in a data frame with 3 entries.
Even though the first element is added as a character, it is stored as a numeric data type.
The second command 'z[1,2] + z[1,3]' takes the element in the first row, second column and adds
it to the element in the first row, third column. This results in 19, which is the sum of 7 and 12.
```

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

```
a <- rep(seq(1,8))
b <- rep(seq(7,1))
c <- c(a,b)
c
```

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

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

```
d <- rep(1:5, times=1:5)
d
```

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

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

```
m <- matrix(0,nrow=3, ncol=3)
m[upper.tri(m, diag=FALSE)] <- 1
m[lower.tri(m, diag=FALSE)] <- 1
m
```

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

4. $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256 \\ 1 & 32 & 243 & 1024 \end{pmatrix}$

```
me <- matrix(0, nrow=5, ncol=4)
r1 <- t(rep(seq(1:4)))
me[1,] <- r1

for (i in seq_along(2:6)){
  me[i,] = (me[1,])^i
}
me
```

```
##      [,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
```

**QUESTION 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)

```
n = 10 # change this value as desired
x = 3 # change this value as desired
x.vec <- rep(0,n)
x.vec[1] = 1

for (i in 2:n) {
  x.vec[i] = x^(i-1)
}
x.vec
```

```
##  [1]     1     3     9    27    81   243   729  2187  6561 19683
```

```
sum(x.vec)
```

```
## [1] 29524
```

2. If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Write an R program to perform the following calculations. (5 points)

```
xa <- seq(1:10)
multa <- c()

for (i in xa-1) {
  if(i%%3==0) {
    multa <- c(multa, xa[i])
  } else {
    if(i%%5==0) {
      multa <- c(multa, xa[i])
    }
  }
}
print(sum(multa))
```

```
## [1] 23
```

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

```
xb <- seq(1:1000)
multb <- c()

for (i in xb-1) {
  if(i%%3==0) {
    multb <- c(multb, xb[i])
  } else {
    if(i%%5==0) {
      multb <- c(multb, xb[i])
    }
  }
}
print(sum(multb))
```

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

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

```
xc <- seq(1:1000000)
multc <- c()

for (i in xc-1) {
  if(i%%4==0) {
    multc <- c(multc, xc[i])
  } else {
    if(i%%7==0) {
      multc <- c(multc, xc[i])
    }
  }
}
sum(as.numeric(multc))
```

```
## [1] 178571071431
```

Some problems taken or inspired by projecteuler. [euler1]: https://projecteuler.net/problem=1 [euler2]: https://projecteuler.net/problem=2