# Bios 301: Assignment 1

*Utsav Kumar*

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 <- data.frame(read.csv("cancer.csv"))
```

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

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

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

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

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

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

```
colnames(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] 350.7
```

    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 <- cbind (cancer.df, "incidence rate (per 100,000)" =
  cancer.df$incidence/cancer.df$population*100000)
```

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

```
sum(cancer.df[,9]==0)
```

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

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

```
which.max(cancer.df[,9])
```

```
## [1] 5797
```

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)

       The vector is a string type. So, each number is considered as string and compared digit by digit and not as a number.

       ```
       max(x)
       Output:
       [1] "7"
       ```

       - If we compare the string vector digit by digit 7 is largest as compared to 5, 1 and 2. Here, since it is a string so 12 is assumed as 1 and 2. Therefore, max(x) would give 7 as the output.

       sort(x)
       Output:
       [1] “12” “5” “7”

       - In sorting using the same logic as presented above. 7 would be largest followed by 5 and then by 12.

       sum(x)
       Output:
       *Error in sum(x) : invalid ‘type’ (character) of argument*

       - Since it is a string vector. Therefore, we cannot perform a numeric operation like sum(x) and would output an error.

    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]
       Output:
       Error in y[2] + y[3] : non-numeric argument to binary operator
       ```

       - In this case the vector has one string and two numbers. The type of vector is dictated by the string because as compared to numbers it has the least degree of freedom. Therefore, the two numbers is also considered as string types here to maintain the homogenity and therefore a numeric operation would output an error.

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]
Output:
[1] 19
```

- Data frame provide the freedom to the entries to have their independent data types like string for first and numbers for the next two not affecting the data typeof other entries as in vector entries as seen above. Therefore, even the first entry's data type is string still the other two entries' data type is still numbers and performing a numeric operation will not result in an error and will get the sum of the 2nd and 3rd entry in the data frame.

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(s <- 1:8, rev(s[-length(s)]))
```

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

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

```
nnos <- seq (1,5,1)
rep (nnos, times=nnos, each=1)
```

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

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

```
mat <- matrix(data=1,nrow=3, ncol=3)
diag(mat) <- 0
mat
```

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

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

```
sqmat <- matrix(1:4, 5, 4, byrow=TRUE)
sqmat[seq(1,5,1),] <- sqmat[seq(1,5,1),]**seq(1,5,1)
sqmat
```

```
##      [,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)

```
# x <- as.numeric(readline("Enter the number: "))
# n <- as.integer(readline("Enter n, should be an integer: "))
x <- 3
n <- 3
h <- 0
for (i in 0:n)
{
  h <- h + x**i
}
h
```

```
## [1] 40
```

    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)

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

```
sum <- 0
for (i in seq(1,999,1))
{
  if (i%%3==0 | i%%5==0)
  {
    sum <- sum + i
  }
}
sum
```

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

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

```
sum <- 0
for (i in seq(1,999999,1))
{
  if (i%%4==0 | i%%7==0)
  {
    sum <- sum + i
  }
}
sum
```

```
## [1] 1.786e+11
```

3. Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be $(1, 2, 3, 5, 8, 13, 21, 34, 55, 89)$. Write an R program to calculate the sum of the first 15 even-valued terms. (5 bonus points)

```
          sumfib <- 2
# to store the sum of the series and since 2 is already initialized and is even so ...
#... it is already added to the sum
          fib <- c()
          fib[1] <- 1
          fib[2] <- 2
          i <- 3 # 1st two terms of the series are already initialized so i starts from 3
          counter <- 1 # 2nd term of the series is 2 which is even so counter starts from 1
          while (counter<15)
          {
            fib[i]=fib[i-1]+fib[i-2]
            if (fib[i]%%2==0)
            {
              sumfib <- sumfib + fib[i]
              counter <- counter + 1
            }
            i <- i+1
          }
          sumfib
```

```
## [1] 1.486e+09
```