Bios 6301: Assignment 2

Yan Yan

(informally) Due Tuesday, 18 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)

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

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

```
dim(cancer.df)
```

```
## [1] 42120     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] 350.69
```

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[,"rate"] <- cancer.df[,"incidence"]/cancer.df[,"population"]*10^5
```

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

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

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

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

```
max.rate <- max(cancer.df$rate)
max.rate
```

```
## [1] 261.1599
```

```
which(cancer.df$rate == max.rate)
```

```
## [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)

        ```
        max(x)
        sort(x)
        sum(x)
        ```

```
x21 <- c("5","12","7")
max(x21)
```

```
## [1] "7"
```

```
sort(x21)
```

```
## [1] "12" "5"  "7"
```

```
#sum(x21)
```

With the use of quotation marks, all these values are recognized as string values. Thus, max and sort will work fine but sum will produce an error.Max gives the greatest value when comparing these values as strings, which is "7". Sort will sort the first character of each element increasingly and then the second and so on, and gives "12" "5" "7".

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]
        ```

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

```
## [1] "character"
```

Only one type of values is allowed in one vector. Here with presence of "5", the other two values 7 and 12 all turned into character values, thus sum can't work on them.

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]
        ```

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

```
## [1] 19
```

Unlike the single vector, data frame can have different types of values. Here 7 and 12 are numerical values, so the operator "+" works fine and gives the sum of these two values.

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 <- c(1:8,7:1)
a
```

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

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

```
b <- rep(1:5,1:5)
b
```

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

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

```
mx <- matrix(as.numeric(!diag(3)),3,3)
mx
```

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

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

```
f <- c(1,2,3,4)
g <- c(f,f^2,f^3,f^4,f^5)
h <- matrix(g,byrow = T, ncol = 4)
h
```

```
##      [,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)

```
hx <- function(x,n){
  hix <- vector()
  for (i in 0:n) {
   hix <- c(hix, x^i)
  }
  print (hix)
  return(sum(hix))
}
```

```
hx(2,3)
```

```
## [1] 1 2 4 8
```

```
## [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

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

```
ThreeFive <- 0
for (i in 1:999) {
  if(i%%3 == 0|i%%5 == 0){
    ThreeFive <- ThreeFive + i
  }
}
print(ThreeFive)
```

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

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

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

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

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

```
  a <- c(1,2)
  b <- 2
 while (1){
      a <- c(a,a[length(a)] + a[length(a)-1])
   if(a[length(a)]%%2 == 0){
     b <- c(b,a[length(a)])
   }
      if(length(b)==15){break}
    }
sum(b)
```

```
## [1] 1485607536
```

```
print(b)
```

```
##  [1]          2          8         34        144        610       2584
##  [7]      10946      46368     196418     832040    3524578   14930352
## [13]   63245986  267914296 1134903170
```

Some problems taken or inspired by projecteuler.