Bios 6301: Assignment 2

Brooklyn Stanley

(informally) Due Thursday, 17 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)

```
# Assumes file is in working directory
cancer.df <- 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] 350.69
```

    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
```

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

```
count = 0
for (row in cancer.df$incidence) {
  if (row == 0) count = count + 1
}
count
```

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

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

```
cancer.df[cancer.df$incidence == max(cancer.df$incidence ), ]
```

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

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)

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

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

```
sort(x)
```

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

```
sum(x)
```

```
## Error in sum(x): invalid 'type' (character) of argument
```

The functions max and sort operate by looking at the numerical representations of the first character in the string. The character "5" is observed as 5, "12" is observed as 1 and "7" is observed as 7. Therefore, the max function returns 7, and the sort function returns the characters in ascending order by their numerical representations: 12, 5, 7. The last command results in an error because sum() only accepts numeric values.

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]
```

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

In order to create y, R converts all of the elements to the same data type. Since the character type is the least flexible, 7 and 12 are converted to character strings. The addition opperator produces an error because it requires numeric values.

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

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

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

The first command creates a data frame with 3 columns each named z1, z2, and z3 respectively. When you pass a character into the data.drame() function, it converts all strings to factor columns unless you set stringAsFactors to false. The second command simply calls on R to add 7 and 12 together, which are both numeric so this works fine.

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(1:8,7:1)
```

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

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

```
rep(seq(5),seq(5))
```

```
##  [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(1,3,3)
diag(mat) <- 0
mat
```

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

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

```r
pwr <- matrix(0,5,4)
for (i in 1:5) {
  pwr[i, ] <- (1:4)^i
}
pwr
```

```
##      [,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)

```r
h_xn <- function(x,n) {
  out <- 0
  for (i in 0:n) {
    out = out + x^i
  }
  return(out)
}
# If working correctly, h_xn(2,4) = 1 + 2 + 4 + 8 + 16 = 31
h_xn(2,4)
```

```
## [1] 31
```

    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, euler1)

```r
  eulerSum <- function(x1, x2, N) {
    # This function assumes that the LCD of x1 and x2 is 1
    euler <- sum(seq(0,N-1,x1)) + sum(seq(0,N-1,x2)) - sum(seq(0,N-1,x1*x2))
    return(euler)
  }
  eulerSum(3,5,1000)
```

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

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

```r
  eulerSum(4,7,1000000)
```

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

    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, euler2)

```r
evenFib <- c(2)
Fib <- c(1,2)
while (length(evenFib) < 15) {
  val <- sum(tail(Fib, 2))
  if (val%%2 ==0) evenFib <- c(evenFib,val)
  Fib <- c(Fib, val)
}
sum(evenFib)
```

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

Some problems taken or inspired by projecteuler.