Bios 6301: Assignment 2

Nick Strayer

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)

```
cancer.df <- read.csv("/Users/Nick/Dropbox/vandy/computing/Bios6301/datasets/cancer.csv")
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)

```
dims <- dim(cancer.df)
```

There are **42120** rows and **8** columns.

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

```
(cn <- 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 * 100000
```

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

```
zeros <- sum(cancer.df$incidence == 0)
```

23191 rows have an incidence rate of `0`.

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
##           rate
## 21387 137.1298
```

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")
max(x)
```

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

```r
sort(x)
```

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

```r
#sum(x)
```

`max()` works because it knows it is looking for numbers and it automatically converts the strings to them.

`sort()` doesn't throw an error but it doesn't sort from highest to lowest. This is because it is sorting alphanumerically. In doing so it just looks at the first digit of the number to sort.

`sum()` throws an error because it isn't neccesarily a specifically numeric function. It can take booleans as well. Because of this it doesn't automatically parse the strings into 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]`

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

This will produce an error because the fact that a single value of the vector was a string makes R convert all of the vector's values to strings. We can see this by running

```r
typeof(y[2])
```

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

. Because of this we are attempting to do addition on two strings. Hence the 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]`

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

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

This works because the individual values are stored in seperate columns of a dataframe. Because of this they don't get automatically converted to strings. Again we can verify this by…

```r
typeof(z[1,2])
```

```
## [1] "double"
```

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

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

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

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

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

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

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

```
(m <- matrix(1, 3,3) - diag(3))
```

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

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

```
m2 <- matrix(rep(1:4, 5),5,4,byrow = T)
(m2 <- m2 ^ row(m2))
```

```
##      [,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)

```
h <- function(x,n){
  res = 0
  for(i in seq(n)){
    res = res + x^i
  }
  return(res)
}

h(5,3)
```

```
## [1] 155
```

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 sum of these multiples is 23. Write an R program to perform the following calculations. (5 points)
2. Find the sum of all the multiples of 3 or 5 below 1,000. (3, euler1)
3. Find the sum of all the multiples of 4 or 7 below 1,000,000. (2)

```r
#1
sum = 0
for(num in seq(1000)){
  if(num %% 3 == 0 || num %% 5 == 0)
    sum = sum + num
}
sum
```

```
## [1] 234168
```

```r
#2
sum2 = 0
for(num in seq(1000000)){
  if(num %% 4 == 0 || num %% 7 == 0)
    sum2 = sum2 + num
}
sum2
```

```
## [1] 178572071431
```

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
#generate the first n terms of the Fibonacci sequence.
fibGen <- function(n){
  seq = c(1,2)
  evens = c(2)
  while(length(evens) < n){
    i      <- length(seq) #where are we in the sequence
    newVal <- seq[i] + seq[i-1] #make the new value
    seq <- c(seq, (seq[i] + seq[i-1])) #add it to the sequence
    if(newVal %% 2 == 0) evens <- c(evens, newVal) #If the new val is an even, add it to the sequence
  }
  return(evens)
}

sum(fibGen(15))
```

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

Some problems taken or inspired by projecteuler.