Bios 6301: Assignment 2

Lydia Yao

Due Tuesday, 21 September, 1:00 PM

50 points total.

Add your name as `author` to the file's metadata section.

Submit a single knitr file (named `homework2.rmd`) by email to michael.l.williams@vanderbilt.edu. Place your R code in between the appropriate chunks for each question. Check your output by using the `Knit HTML` 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("cancer.csv")
```

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 incidence
## 172 1999 Brain and Other Nervous System nevada Male Black         0         0
##     population
## 172      73172
```

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

```
cancer.df["rate"] = (cancer.df$incidence/cancer.df$population)*100000
```

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

```
nrow(cancer.df[cancer.df$rate == 0,])
```

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

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

```
cancer.df[which.max(cancer.df$rate),]
```

```
##      year     site                state  sex  race mortality incidence
## 5797 1999 Prostate district of columbia Male Black     88.93       420
##      population     rate
## 5797     160821 261.1599
```

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)
        ```

        `sum(x)` will produce an error message. Because the list created is a list of characters, we are unable to sum them thus we get the error `invalid 'type' (character) of argument`. `max(x)` will give the max of the character. Since 7 is after both 5 and 1, it is our max. `sort(x)` will sort the list based on the first characters thus give us `"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]
        ```

        Line two of the code above should return errors because of type incompatibility. We can not sum two characters. Since 7 and 12 will be saved as characters because of "5", the result will return `non-numeric argument to binary operator`.

    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 result of the above code is 19. Unlike the previous question, this does not produce an error because z[1,2] and z[1,3] are not saved as characters given the structure of 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(seq(1:8), rev(seq(1:7)))
```

```
##  [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(1:5), seq(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(0,3,3)
m[lower.tri(m)] <- 1
m[upper.tri(m)] <- 1
m
```

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

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

```
mat1 <-rep(seq(1:4),5)
mat2 <- matrix(mat1,nrow=5,ncol=4,byrow=TRUE)
mat3 <- seq(1:5)
mat4 <- mat2**mat3
mat4
```

```
##      [,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. As an example, use `x = 5` and `n = 2`. (5 points)

```
sum_h <- function(x, n) {
  sum = 0
  for(i in 0:n) {
    sum = sum + x**i
  }
  print(sum)
}
sum_h(5,2)
```

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

```
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])
```

```
l = (0)
for (i in 1:999){
  if (i%%3 == 0 || i%%5 == 0){
    l = append(l, i)
  }
}
```

```
sum(l)
```

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

```
three <- floor(999/3)
five <- floor(999/5)
both <- floor(999/15)
3*three*(three+1)/2 + 5*five*(five+1)/2 - 15*both*(both+1)/2
```

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

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

```
four <- floor(1000000/4)
seven <- floor(1000000/7)
both <- floor(1000000/28)
4*four*(four+1)/2 + 7*seven*(seven+1)/2 - 28*both*(both+1)/2
```

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

```
l = (0)
for (i in 1:1000000){
  if (i%%4 == 0 | i%%7 == 0){
    l = append(l, i)
  }
}
sum(l)
```

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

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

```
sum_fib <- function(n) {
  fib <- rep(NA, n*3)
  if (n <= 0){print(0)}
  else{
    fib[1] <- 0
    fib[2] <- 1
    count = 0
    sum = 0
    i = 3
    while(count < n){

      fib[i] = fib[i - 1] + fib[i-2]
      if(fib[i]%%2==0){
        sum = sum + fib[i]
        count = count + 1
      }
      i = i + 1
    }
  }
  print(sum)
}
sum_fib(15)
```

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

Some problems taken or inspired by projecteuler.