Bios 6301: Assignment 2

Megan Taylor

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("C:\\Users\\megan\\OneDrive\\Documents\\Fall 2021\\Statistical Computing\\Bios630
# path is "C:\\Users\\megan\\OneDrive\\Documents\\Fall 2021\\Statistical
# Computing\\Bios6301-main\\Bios6301-main\\datasets\\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 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 the `(number of cases)/(population at risk)`, which in this case means `(number of cases)/(population at risk) * 100,000`. (3)

```
cancer.df <- cbind(cancer.df, Incidence_Rate = (cancer.df$incidence/cancer.df$population*100000))
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 Incidence_Rate
## 1        19     623475       3.047436
## 2         0      28101       0.000000
## 3       110    1640665       6.704598
## 4        18     539198       3.338291
## 5         0      37082       0.000000
## 6       145    1570643       9.231888
```

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

```
ix <- which(cancer.df[,'Incidence_Rate']==0)
length(ix)
```

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

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

```
which(cancer.df[,"Incidence_Rate"]==max(cancer.df$Incidence_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)
        ```

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

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

```
# Max returns the highest value in the vector, which in this case is "7".
# Since these are of the character data type, the "7", and the "5", are higher
# than the "12" because it looks at the "1" first.

sort(x)
```

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

```
# Sort returns the vector sorted into ascending order. As explained above,
# the "12" evaluates as lower than the "5" and "7" because the "1"
# character is lower.
```

```
# Sum(x) will give an error because the elements in the vector are
# characters, and sum expects to add numeric values. Character values cannot
# be coerced into numeric data. In contrast, logical values are also not
# numeric, but can be coerced, so that would not produce 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]
```

```
y <- c("5",7,12)
# This command produces a vector with three character values. The 7 and 12
# are coerced into character values since the "5" is a character.

# Similar to the sum in question 1, the second command in this problem will
# produce an error because it is attempting to perform arithmetic on two
# character values, rather than numbers.
```

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)
# This command produces a data frame with 1 row and 3 columns. The column
# names are z1, z2, and z3, defined in the arguments of the data.frame
# function. Each column is given its value for the only row by the value that
# follows the equal sign. The value in the first column ("5") is a character,
# the value in the second column (7) is numeric, and the value in the third
# column (12) is numeric.

z[1,2] + z[1,3]
```

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

```
# This command adds the value in the first row, second column and the value
# in the first row, third column. These are both numeric, so the command works.
```

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

```
x <- c(seq(1:8),seq(7, 1, -1))
x
```

```
##  [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(1:5, times = 1:5)
```

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

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

```
x <- matrix(rep(1, times = 9), nrow = 3)
diag(x) <- 0
```

```
x
```

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

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

```
a <- c((1:4), (1:4)^2, (1:4)^3, (1:4)^4, (1:4)^5)
am <- matrix(a, nrow = 5, byrow=TRUE)
am
```

```
##      [,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)

```
x = 5
n = 2
h = 1
for (ni in n) {
  h = h + x^ni
}
h
```

```
## [1] 26
```

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)

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

```
x <- (1:999)
sum(which(x %% 3 == 0 | x %% 5 == 0))
```

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

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

```
x <- (1:999999)
sum(which(x %% 4 == 0 | x %% 7 == 0))
```

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

1. 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)

```
# start with the first 3 numbers of the sequence
fibseq = c(1, 2, 3)
```

```r
# initialize i to 4 because the loop will start by adding the fourth
# number to the sequence
i = 4
# initialize a counter for the number of even numbers
even = 0
# initialize the vector that will contain the even numbers
evennum = 0
# set up a while loop that runs until there are 15 even numbers
while (even < 15) {
  # add the next number to the sequence by taking the sum of the two
  # previous numbers
  fibseq = c(fibseq, (fibseq[i-1]+fibseq[i-2]))
  #if the number is even, add 1 to the counter and add the number to
  # the evennum vector
  if(fibseq[i] %% 2 == 0){
    even = even + 1
    evennum = c(evennum, fibseq[i])}
  # increase the index variable by 1
  i = i + 1
}
# sum the first fifteen numbers
sum(evennum)
```

```
## [1] 6293134510
```

Some problems taken or inspired by projecteuler.