Bios 6301: Assignment 2

Valerie Welty

Grade 52/50

(informally) Due Tuesday, 20 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)

    ```
    cancer.df <- as.data.frame(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)

    ```
    (head <- names(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[, 9] <- cancer.df[, 7]/1e+05
    names(cancer.df)[9] = "rate"
    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    rate
## 1        19     623475 0.00019
## 2         0      28101 0.00000
## 3       110    1640665 0.00110
## 4        18     539198 0.00018
## 5         0      37082 0.00000
## 6       145    1570643 0.00145
```

JC Grading - 1 For incidence rate above should be incidence / population * 100000

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

```r
head(table(cancer.df[, 9]))[1]
```

```
##     0
## 23191
```

```r
# 23,191 subgroups have a zero incidence rate.
```

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

```r
(max <- cancer.df[which.max(cancer.df[, 9]), ])
```

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

```r
# The highest incidence rate in the data set is 0.18744, occuring at
# subgroup 21387.
```
~~~

JC Grading - 1 syntax is fine but answer is incorrect b/c of how incidence rate was calculated

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)

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

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

```r
max(x)
```

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

```r
# max(x) will report the last value in the vector for character type.
sort(x)
```

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

```r
# sort(x) will sort the values alphabetically, hence why '12' occurs first
# (the 1 in front)

# sum(x)

# sum(x) will report an error because the mode of the variable x is
# character, and sum(x) can only operate on numeric/integer types.
```

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)
mode(y)
```

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

```r
# y[2] + y[3]

# this operation should produce an error because, while 7 and 12 were
# inputted as numeric values, there was a character value in the vector as
# well. Character is the least flexible value, so because there was a
# character value in the vector the remaining values will be forced into
# character values as well. Thus, y[2] and y[3] are not numeric values and
# thus cannot be operated on mathematically.
```

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
```

```r
# The command will take the value in the 1st row and 2nd column of z (=7)
# and add it to the value in the first row and third column of z (=12) which
# results in 19.
```

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), rev(1:7))
x
```

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

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

```
x <- rep(1:5, times = 1:5)
x
```

```
##  [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(data = 1, nrow = 3, ncol = 3)
diag(x) = 0
x
```

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

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

```
x <- matrix(data = NA, nrow = 5, ncol = 4)
for (i in 1:5) {
    for (j in 1:4) {
        x[i, j] = j^i
    }
}
x
```

```
##      [,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) {
    sum <- 0
    for (i in 0:n) {
        sum <- sum + (x^i)
    }
```

```r
    return(sum)
}

h(2, 3)
```

```
## [1] 15
```

```r
h(4, 5)
```

```
## [1] 1365
```

```r
h(6, 3)
```

```
## [1] 259
```

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)

```r
euler <- function(x.1, x.2, n) {
    x <- vector(mode = "integer", length = 0)
    sum <- 0
    for (i in 1:n - 1) {
        if ((i%%x.1 == 0) | (i%%x.2 == 0)) {
            x <- c(x, i)
            sum <- sum + i
        }
    }
    x
    return(sum)
}
```

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

```r
euler(3, 5, 1000)
```

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

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

```r
euler(4, 7, 1e+06)
```

```
## [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)

```r
sumeven <- function(n) {
    x <- c(1, 2)
    even <- vector(mode = "integer", length = 0L)
    for (i in 3:100) {
        x[i] = x[i - 1] + x[i - 2]
        if (x[i]%%2 == 0) {
            even = c(even, x[i])
        }
        if (length(even) == n) {
            break
        }
    }
    print(x)
    print(even)
    print(paste0("sum of even fibonacci values up to ", n, " = "))
    return(sum(even))
}
sumeven(15)
```

```
##  [1]          1          2          3          5          8         13
##  [7]         21         34         55         89        144        233
## [13]        377        610        987       1597       2584       4181
## [19]       6765      10946      17711      28657      46368      75025
## [25]     121393     196418     317811     514229     832040    1346269
## [31]    2178309    3524578    5702887    9227465   14930352   24157817
## [37]   39088169   63245986  102334155  165580141  267914296  433494437
## [43]  701408733 1134903170 1836311903 2971215073 4807526976
##  [1]          8         34        144        610       2584      10946
##  [7]      46368     196418     832040    3524578   14930352   63245986
## [13]  267914296 1134903170 4807526976
## [1] "sum of even fibonacci values up to 15 = "
```

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

JC Bonus +4 Skipped 2 as the first even

Some problems taken or inspired by projecteuler.