Bios 6301: Assignment 2

Josh DeClercq

September 14, 2016

(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 <- read.csv("cancer.csv")
df.cancer <- data.frame(cancer)
```

2. Determine the number of rows and columns in the data frame. (2)

```
nrow(df.cancer)
```

```
## [1] 42120
```

```
ncol(df.cancer)
```

```
## [1] 8
```

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

```
names(df.cancer)
```

```
## [1] "year"       "site"       "state"      "sex"        "race"
## [6] "mortality"  "incidence"  "population"
```

4. Report the value of the 3000th row in column 6. (2)

```
names(df.cancer)[6]
```

```
## [1] "mortality"
```

```
df.cancer$'mortality'[3000]
```

```
## [1] 350.69
```

5. Report the contents of the 172nd row. (2)

```
df.cancer[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)

```
df.cancer$'rate' <- df.cancer$'incidence'/100000
```

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

```
length(which(df.cancer$'rate' == 0))
```

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

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

```
match(max(df.cancer$'rate'),df.cancer$'rate')
```

```
## [1] 21387
```

```
df.cancer[21387,]
```

```
##       year   site      state    sex  race mortality incidence population
## 21387 2002 Breast california Female White   3463.74     18774   13690681
##          rate
## 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)
     sort(x)
     sum(x)
```

The vector x is made up not of numbers, but of characters because of the quotations. Therefore, the max function is returning the larges ascii value, which is 7. The sort function sorts the 12 first because 12 is alphabetically before 5 and 7. the sum function returns an error because it cannot add character strings.

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]
```

The vector y is coerced to the least flexible type, which in this case is a character. Therefore, adding the character "7" to the character "12" returns an error for the same reason as stated in the answer to number 1.

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]
```

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

```
# z[1,1] + z[1,3] This would generate an error
```

This does not produce an error because the character value is not being called from the data frame. If I were to try to add z[1,1] to either of the other inputs, then it would return an error.

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

```
8 - abs(-7:7)
```

```
##  [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
```

```
c(seq(1,8),seq(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(1:5, 1:5)
```

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

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

```
1-diag(3)
```

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

```
matrix(1,nrow=3,ncol=3) - diag(3)
```

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

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

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

```
##      [,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)

```
n <- 10
x <- 2
vn <- numeric(n)
for (i in 1:n){
  vn[i] <- x^(i-1)
}
sum(vn)
```

```
## [1] 1023
```

```
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])
```

```r
x1 <- NULL
for (i in 1:999){
  if (i %% 3 == 0){
    x1 <- c(x1,i)
  } else if (i %% 5 == 0){
    x1 <- c(x1,i)
  }
}
sum(x1)
```

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

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

```r
n <- 999999
x1 <- numeric(n)
for (i in 1:n){
  if (i %% 4 == 0){
    x1[i] <- i
  } else if (i %% 7 == 0){
    x1[i] <- i
  }
}
sum(x1)
```

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

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

```r
F <- c(1,2,0)
EF <- NULL
for(i in 3:1000){
  F[i] <- F[i-1] + F[i-2]
  if (F[i] %% 2 == 0){
    EF <- c(EF, F[i])
  }
  if(length(EF) == 15) break
}
sum(EF)
```

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

Some problems taken or inspired by projecteuler.