Bios 6301: Assignment 2

Max Rohde

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)

```
# Read in the data
cancer.df <- read_csv('cancer.csv')
```

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

```
# Get the dimensions, ROWS x COLUMNS
dim(cancer.df)
```

```
## [1] 42120     8
```

There are 42120 rows and 8 columns in this data frame.

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

```
# Get the column names
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)
```

```
# Get the value of the 6th column of row 3000
# [[row, column]]
cancer.df[[3000,6]]
```

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

The value is 350.69

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

```
# [ROW 172, all columns]
cancer.df[172,] %>%
  kbl(caption = "The 172nd row") %>%
  kable_classic(full_width = F, html_font = "Source Sans Pro")
```

```
# another approach
cancer.df %>% slice(172) %>%
```

Table 1: The 172nd row

year	site	state	sex	race	mortality	incidence	population
1999	Brain and Other Nervous System	nevada	Male	Black	0	0	73172

Table 2: The 172nd row (tidyverse method)

year	site	state	sex	race	mortality	incidence	population
1999	Brain and Other Nervous System	nevada	Male	Black	0	0	73172

Table 3: Added incidence rate

year	site	state	sex	race	mortality	incidence	population	inc_rate
1999	Brain and Other Nervous System	alabama	Female	Black	0.00	19	623475	3.047436
1999	Brain and Other Nervous System	alabama	Female	Hispanic	0.00	0	28101	0.000000
1999	Brain and Other Nervous System	alabama	Female	White	83.67	110	1640665	6.704598
1999	Brain and Other Nervous System	alabama	Male	Black	0.00	18	539198	3.338291
1999	Brain and Other Nervous System	alabama	Male	Hispanic	0.00	0	37082	0.000000
1999	Brain and Other Nervous System	alabama	Male	White	103.66	145	1570643	9.231888

```
  kbl(caption = "The 172nd row (tidyverse method)") %>%
  kable_classic(full_width = F, html_font = "Source Sans Pro")
```

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

```
# Calculate incidence rate
cancer.df <- mutate(cancer.df, inc_rate = (incidence / population) * 1e5)

# Check the results
head(cancer.df) %>%
  kbl(caption = "Added incidence rate") %>%
  kable_classic(full_width = F, html_font = "Source Sans Pro")
```

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

```
cancer.df %>%
  filter(inc_rate==0) %>%  # Filter to rows with zero incidence rate
  nrow() # Calculate number of rows
```

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

23191 subgroups have a zero incidence rate.

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

```
cancer.df %>%
  arrange(desc(inc_rate)) %>% # Sort the data frame from highest incidence rate to lowest
  head(1) %>% # Take the first row
  kbl(caption = "Row with highest incidence rate") %>%
  kable_classic(full_width = F, html_font = "Source Sans Pro")
```

The subgroup shown above has the highest incidence rate.

2. **Data types** (10 points)

Table 4: Row with highest incidence rate

year	site	state	sex	race	mortality	incidence	population	inc_rate
1999	Prostate	district of columbia	Male	Black	88.93	420	160821	261.1599

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")

# The max() command compares character strings in order from first to last character.
# Because 7 > 5 > 1, 7 is returned.
max(x)
```

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

```r
# To obtain the standard result (treating each element as a number)
# we can cast to integer first
x %>% as.integer() %>% max()
```

```
## [1] 12
```

```r
# For the same reasons as above, the vector will be sorted as
# 12, 5, 7 because only the first character is compared
# and 1 < 5 < 7
sort(x)
```

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

```r
# The usual result after conversion
x %>% as.integer() %>% sort()
```

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

```r
# sum() will return an error because it has no rules defined
# for adding together two character vectors

#sum(x)
```

```r
# we must first convert to an integer
x %>% as.integer() %>% sum()
```

```
## [1] 24
```

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
# This code produces an error because when creating the vector `y`,
# all of the elements are coerced to characters so when the 2nd
# line is run, we are adding "7" + "12". This is undefined
# because R does not have addition defined for characters
# (unlike a language like Python where they would be concatenated)

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

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
# This code works as expected, adding 7 + 12 = 19, unlike the last problem,
# because while vectors can only have one type, data frames can have each column
# be a different type. Thus, 7 and 12 are allowed to be integers, and exist
# in the data frame with "5" because they are in their own columns, and each
# column is a vector of a single type.
```

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

```r
count_to <- function(x) c(1:(x-1),x,(x-1):1)

count_to(8)
```

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

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

```r
rep(1:5, times=1:5)
```

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

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

```r
# diag(3) creates a 3x3 identity matrix
# then subtract 1 and take the absolute value to obtain the desired output

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

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

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

```r
# First we create a list of numbers 1:5
m <- rep(1:4, 5)
```

```r
# Then put into a matrix
m <- matrix(m, nrow=5, byrow = TRUE)

# Then raise the first row to the first power, second row to second power...etc
m^(1:5)
```

```
##      [,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)

```r
h <- function(x,n){
  sum <- 0
  for (i in 0:n){
    # Add each term of the polynomial to the sum
    sum <- sum + x^i
  }
  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])
```

```r
div3 <- ((1:999) %% 3) == 0 # Boolean array of elements divisible by 3
div5 <- ((1:999) %% 5) == 0 # Boolean array of elements divisible by 5
div3_5 <- div3 | div5 # Divisible by 3 OR 5

(1:999)[div3_5] %>% sum() # Get the relevant elements by boolean indexing, then sum.
```

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

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

```r
# Same approach as last problem

div4 <- ((1:10^6-1) %% 4) == 0
div7 <- ((1:10^6-1) %% 7) == 0
div4_7 <- div4 | div7

(1:10^6-1)[div4_7] %>% sum()
```

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

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

```r
# Stores the Fibonacci sequence
x <- numeric()

# Stores only the even terms
evens <- numeric()

# Define initial parameters
x[1] <- 1
x[2] <- 1

# We start the loop at 3, because the first two numbers are already defined
i <- 3

# We loop until we found 15 even fibonacci numbers
while(length(evens) < 15)
{
  # The new value is the sum of the previous two in the sequence
  new <- x[i - 2] + x[i - 1]
  x[i] <- new

  # If the new value is even, append it to the list of even numbers
  if((new %% 2) == 0) evens <- append(evens, new)

  # Increment the counter to move on to the next fibonacci number
  i <- i+1
}

# Sum the 15 even fibonacci numbers
s <- sum(evens)
```

```
The sum is 1485607536
```

Some problems taken or inspired by projecteuler.