Homework_2

Molly Olson

September 6, 2015

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 csv format. This is a large cancer incidence dataset that summarizes the incidence of different cancers for various subgroups. i. Load the data set into R and make it a data frame called `cancer.df`.

```
setwd("/Users/mollyolson/documents/Vanderbilt 2015 Fall/Statistical Computing/Bios6301-master/datasets"
cancer.df <- read.csv("cancer.csv",header=TRUE)
```

ii. Determine the number of rows and columns in the

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

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

```
#8 columns
nrow(cancer.df)
```

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

```
#42,120 rows
```

iii. Extract the names of the columns in `cancer.df`

```
colnames(cancer.df)
```

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

iv. Report the value of the 3000th row in column 6

```
cancer.df[3000,6]
```

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

v. Report the contents of the 172nd row

```
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
```

vi. Create a new column that is the incidence rate (per 100,000) for each row.

```
cancer.df$IncidenceRate <- (cancer.df$incidence / cancer.df$population) * 10^5
```

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

```
length(which(cancer.df$IncidenceRate == 0.00))
```

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

```
#23,191 rows have 0 incidence rates
```

viii. Find the subgroup with the highest incidence rate

```
which.max(cancer.df$IncidenceRate)
```

```
## [1] 5797
```

```
#row 5797
```

2. Data Types

i. 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.

```
x <- c("5","12","7")
#y <- c("apple","banana")
#z <- c(T,F)
```

sum(x) will give an error.

max(x) gives the maximum value, regardless of the data type.
sort(x) will sort the data, regardless of type. It sorts in increasing pattern. It sorts 12 first, because it is looking at the "10's" position, which is equal to 1. Then it will sort 5, since 1<5<7, and then 7.
sum(x) produces an error because the vector is not numerical, where sum works on numberical vectors.

```
max(x)
```

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

```
sort(x)
```

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

```
#sum(x)
```

ii. For the next two commands, either explain their results, or why they should produce errors.

The first command: you can have different "types" of elements into a vector, but a vector can only be one type, so it will pick the least flexible type. So defining y will work using this command, but the vector type will be character, not a combination of characters and integers. So, if we call class(y), we will get character.

```
y <- c("3",7,12)
#class(y)
```

The second command: You can not have a vector be more than one data type, so it will choose the least flexible data type in the vector to be the vector type. In this case, character is the least flexible, so 7 and 12 also become characters. You can not add character values together, so it will produce an error that says "non-numerica argument to binary operator"

```
y[2]
```

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

iii. For the next two commands, either explain their results, or why they should produce errors.

For the first command, we can define the dataframe this way because each column of the data frame is considered to be it's own part. So if we call class on z, we will get back "data.frame"

```
z <- data.frame(z1="5",z2=7, z3=12)
```

For the second command, since each column of the data frame is it's own part, if we call class(z*z*2), *wewillget*"*numeric*", *andlike*
So, since z2 and z3 are both numberic, we can perfrom additon on them.

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

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

3. Data Structures.

Give R expressions that return the following matrices and vectors. (ie do not construct them manually)

i. (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
```

ii. (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
```

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

```
matrix(rep(1,9),nrow=3) - diag(3)
```

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

iv.

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

1 2 3 4 1 4 9 16 1 8 27 64 1 14 81 256

```
x <- c(1:4)
t(matrix(c(x,x^2,x^3,x^4),ncol=4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    1    4    9   16
## [3,]    1    8   27   64
## [4,]    1   16   81  256
```

Basic Programming

i. 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.

```
#for(i in )
```