Individual Homework 1

Name: Arthur Viegas Eguia

I worked with:

Click the "Knit" button in RStudio to knit this file to a pdf.

Problem 1: Markdown output

a. What output is produced when all three chunks include the option echo = FALSE. Explain your answer in words, not by adding these options to chunks above.

answer: 2 3 4

Echo means that the code is not printed when the PDF is knitted, but the results of the code are printed. The code won't be on the document, but its results are going to be there. The first chunk creates vector 1 2 3, the second chunk adds 1 to each of the elements of the vector. So the output is 2 3 4.

b. What output is produced when all three chunks include the option echo = FALSE and chunk2 includes the options eval=FALSE. Explain your answer in words, not by adding these options to chunks above

answer: 1 2 3

Eval=false means that the chunk of code will not be evaluated when the document is knitted. So it is like the code is not run. Everything that runs on a chunk that has eval=FALSE is ignored by knitting, and by the other code cells in the document.

The output is the result of creating vecrtor 1 2 3, ignoring chunk 2 and printing the vector

---

Problem 2: Logical vectors

Suppose we have a list of food (carrot, orange, m&m, apple, candy corn) that we characterize by color (orange or not) and candy (candy or not). Here are the data vectors describing each food:

```r
orange <- c(TRUE, TRUE, FALSE, FALSE, TRUE)
candy <- c(FALSE, FALSE, TRUE, FALSE, TRUE)
table(orange, candy)
```

```
##        candy
## orange  FALSE TRUE
##   FALSE     1    1
##   TRUE      2    1
```

a. What type of food does the product of these vectors represent? (e.g. what does x of 0 mean? what does 1 mean?)

answer: 0 0 0 0 1 (all the candy that is orange)

This is all the candy that is orange. If a number is, 0 it can be interpreted as false. If a number is 1, it can be interpreted as true. Here only the last item is 1, which is the only candy that is also orange.

```
x <- orange*candy
x
```

```
## [1] 0 0 0 0 1
```

b. What type of food does the vector y represent? (e.g. what does y of 0 mean? what does 1 mean?)

answer: All the food in the vectors that is not orange and is not candy

Here, (1 - orange) is the opposite of orange, if something was 1(true) in orange, it will be 0(false) in (1 - orange). What is more, what is 0 in orange is going to be 1 in (1 - orange). This happens as 1 - 0 = 1 and 1 - 1 is 0. in other words you do 1 minus every element of the vector.

The same thing happens for (1 - candy)

So, this is every food that is not candy and is not orange

```
y <- (1-orange)*(1-candy)
y
```

```
## [1] 0 0 0 1 0
```

c. What type of food does the vector z represent? (e.g. what does z of 0 mean? 1?)

answer: All the food in the vectors that is orange and is not candy

```
z <- orange*(1-candy)
z
```

```
## [1] 1 1 0 0 0
```

Here (1 - candy) is the opposite of candy, so if something was 1 (true) in candy, it is going to be 0(false) in (1 - candy), and everything that was 0(false) in candy is going to be 1(true) in (1 - candy). R gets these results by doing 1 - item for every single item in candy.

This gives us every food that is orange and is not candy.

Problem 3: Coercion

```
obj1 <- 2:10
obj2 <- c(-1, 1)
obj3 <- c(TRUE, FALSE)
obj4 <- 10
```

a.

answer: 3 4 5

This is the output because obj1 is the vector 2 3 4 5 6 7 8 9 10

By doing obj[2:4] we are choosing the items in positions 2, 3 and 4 of this vector. The items in these positions are 3 4 5

b.

answer: 2 3 5 6 7 8 9 10

This happens because obj1 is vector 2 3 4 5 6 7 8 9 10

When we do -3, we are removing the third element in the vector (which is 4). So the output is the entire vector without 4 which will be removed.

c.

answer: A warning about the object legnths, and 1 4 3 6 5 8 7 10 9

The warning is produced because the lists are of differnt lengths and their lengths are not multiples of each other.

The number list is produced by adding the first element of obj1 to the first element of obj2, then the second element is the second element of obj1 plus the second element of obj2. The third element is the third element of obj1 plus the first element of obj2 (as obj 2 only has 2 elements), and so on.

d.

answer: A warning about object lengths and -2 3 -4 5 -6 7 -8 9 -10

The warning is produced because the lists are of different lengts and their lengths are not multiples of each other.

The number vector is produced by multiplying the first element of obj1 with the first element of obj2, then the second element is the second element of obj1 times the second element of obj2. The third element is the third element of obj1 multiplied by the first element of obj2 (as obj2 only has two elements), and so on.

e.

answer: 1

Under the hood FALSE is 0 and TRUE is 1. obj3 is TRUE, FALSE. That can be translated to 1, 0. So their sum is 1

Problem 4: Data frames

```r
library(dslabs)
data(murders)
```

a.

answer: Object a is of class character

```r
a <- murders$abb
a
```

```
##  [1] "AL" "AK" "AZ" "AR" "CA" "CO" "CT" "DE" "DC" "FL" "GA" "HI" "ID" "IL" "IN"
## [16] "IA" "KS" "KY" "LA" "ME" "MD" "MA" "MI" "MN" "MS" "MO" "MT" "NE" "NV" "NH"
## [31] "NJ" "NM" "NY" "NC" "ND" "OH" "OK" "OR" "PA" "RI" "SC" "SD" "TN" "TX" "UT"
## [46] "VT" "VA" "WA" "WV" "WI" "WY"
```

b.

answer: They are not identical, using [] to access an object preserves the list type. So b is of class data.frame while a is of class character

```
b<- murders[2]
b
```

```
##    abb
## 1   AL
## 2   AK
## 3   AZ
## 4   AR
## 5   CA
## 6   CO
## 7   CT
## 8   DE
## 9   DC
## 10  FL
## 11  GA
## 12  HI
## 13  ID
## 14  IL
## 15  IN
## 16  IA
## 17  KS
## 18  KY
## 19  LA
## 20  ME
## 21  MD
## 22  MA
## 23  MI
## 24  MN
## 25  MS
## 26  MO
## 27  MT
## 28  NE
## 29  NV
## 30  NH
## 31  NJ
## 32  NM
## 33  NY
## 34  NC
## 35  ND
## 36  OH
## 37  OK
## 38  OR
## 39  PA
## 40  RI
## 41  SC
## 42  SD
## 43  TN
## 44  TX
## 45  UT
## 46  VT
## 47  VA
## 48  WA
## 49  WV
## 50  WI
## 51  WY
```

```
class(b)
```

```
## [1] "data.frame"
```

```
identical(a, b)
```

```
## [1] FALSE
```

c.

answer: It is of class data.frame, as the square brackets preserve the data type

```
c <- murders[1:2,1:4]
c
```

```
##     state abb region population
## 1 Alabama  AL  South    4779736
## 2  Alaska  AK   West     710231
```

d.

answer: Because of Implicit Coercion in R. As character is the most complex data type in the data we are getting from the dataframe to populate the matrix, all entries are casted to character.

```
d <- as.matrix(murders[1:2,1:4])
d
```

```
##   state     abb  region  population
## 1 "Alabama" "AL" "South" "4779736"
## 2 "Alaska"  "AK" "West"  " 710231"
```

Problem 5: Lists

```
mylist <- list(x1="molly", x2=42, x3=FALSE, x4=1:5)
```

a.

answer: use mylist$x1, which accesses that element

```
mylist$x1
```

```
## [1] "molly"
```

```
class(mylist$x1)
```

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

b.

answer: use mylist$x2 which accesses that element and returns an element of the desired class

```r
mylist$x2
```

```
## [1] 42
```

```r
class(mylist$x2)
```

```
## [1] "numeric"
```

c.

answer: use mylist$x4[3:4] which returns the 3rd and 4th elements of x4

```r
mylist$x4[3:4]
```

```
## [1] 3 4
```

d.

answer: Command length(mylist$x4) returns the length of x4

```r
length(mylist$x4)
```

```
## [1] 5
```

Problem 6: More lists

a.

answer: This is going to be a list, as using [] to access elements preserves the list type

```r
mylist[1]
```

```
## $x1
## [1] "molly"
```

```r
class(mylist[1])
```

```
## [1] "list"
```

b.

answer: This is going to be character, as it is using the [[]] operator, which gives the object stored at that location.

```r
mylist[[1]]
```

```
## [1] "molly"
```

```r
class(mylist[[1]])
```

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

c.

‘ *answer:* It will be an atomic vector of class character. This happens because character is the most complex data type in the original list. So R uses implicit coercion to convert all data in the original list to type character.

```r
class(unlist(mylist))
```

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

```
unlist(mylist)
```

```
##      x1      x2      x3     x41     x42     x43     x44     x45 
## "molly"    "42" "FALSE"     "1"     "2"     "3"     "4"     "5"
```