SPR Week 1 Exercises

Sami Cemek

Vector subsetting

A local animal rescue group is trying to track the effectiveness of their social media presence; they are currently interested in tracking follower growth. The table below summarizes the number of page likes or new followers each day:

	Sun	Mon	Tues	Wed	Thurs	Fri	Sat
FB	30	43	55	89	71	52	42
Twitter	60	32	86	44	21	30	28

Continue this problem from last week by creating vectors for fb and twitter containing the likes/follows for each day. Assign the days of the week (“Sunday”, “Monday”, etc.) as names for your vectors.

```
fb <- c(30,43,55,89,71,52,42)
twitter <- c(60,32,86,44,21,30,28)

days <- c("Sun", "Mon", "Tues", "Wed", "Thurs", "Fri", "Sat")

names(fb) <- days
names(twitter) <- days
```

VS1. Print just the new fb number of likes/follows from Monday. What about just the likes/follows from the last day in the data set? Can you see which day of the week the last day is (supposing you didn’t know it was Saturday)?

Your answer: See the code below.

```
#Monday is the second index therefore:
fb[2]
```

```
## Mon
##  43
```

```
#or we can use characters such as:
fb["Mon"]
```

```
## Mon
##  43
```

```
#We can check if they are equal
identical(fb[2],fb["Mon"])
```

```
## [1] TRUE
```

```
#if we don't know the last day is Saturday we can use index since there are
# 7 days in a week.
fb[7] #this will print the Saturday's value
```

```
## Sat
##  42
```

```
#or we can use the tail function:
tail(fb, 1)
```

```
## Sat
##  42
```

VS2. Print just the weekends' number of fb likes/follows. Can you show two different ways to do this?

Your answer:

```
#We can add Saturday + Sunday using their vertex
weekend_fb_like <- fb[1] + fb[7]
weekend_fb_like #print the variable to see the value
```

```
## Sun
##  72
```

```
#Using tail and head functions:
weekend_fb_like1 <- tail(fb, 1) + head(fb,1)
weekend_fb_like1
```

```
## Sat
##  72
```

```
#We can add Saturday + Sunday using their names
weekend_fb_like2 <- fb["Sat"] + fb["Sun"]
weekend_fb_like2
```

```
## Sat
##  72
```

VS3. Find which days had more than 50 new likes/follows on FB.

Your answer: Tuesday, Wednesday, Thursday, Friday

```
fb[1:7] > 50
```

```
##   Sun   Mon  Tues   Wed Thurs   Fri   Sat
## FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
```

```
#return TRUE if the day has more than 50 new likes/follows on FB
```

VS4. Let's define a day as "Facebook favorite" if there were more than 50 new likes/follows on FB and fewer than 31 new likes/follows on twitter. Determine whether each day in our data set is a facebook favorite.

Your output should be a vector of TRUE's and FALSE's, corresponding to each day of the week.

Your answer: Thursday, Friday

```
facebook_favorite <- (fb[1:7] > 50) & (twitter[1:7] < 31)
facebook_favorite
```

```
##   Sun   Mon  Tues   Wed Thurs   Fri   Sat
## FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
```

VS5. Now print out the number of new fb likes/follows, only for days which are facebook favorites.

Your answer: 123

```
#Thursday + Friday
fb[5] + fb[6]
```

```
## Thurs
##   123
```

Matrix exercises

A local animal rescue group is trying to track the effectiveness of their social media presence; they are currently interested in tracking follower growth. The table below summarizes the number of page likes or new followers each day:

	Sun	Mon	Tues	Wed	Thurs	Fri	Sat
FB	30	43	55	89	71	52	42
Twitter	60	32	86	44	21	30	28

ME1. Create a matrix with 7 rows and 2 columns containing the number of new follows for facebook (column 1) and twitter (column 2). Name the rows and columns appropriately ("Mon", "Tues", etc. for each row, and "Facebook", "Twitter" for the columns).

Save this matrix as "matrix1", and then print it.

Hint 1: This is easiest to do by starting over again with the raw data; it may actually be more challenging to use the named vectors you created earlier.

Hint 2: Break this problem down into smaller pieces. First create the matrix (there is starter code at the very end of this document if you want it; intermediate students should try it on their own first). Then update your code to save the matrix. Then update your code to add column names, etc.

Your answer:

Note to myself 1: You can also use dimnames = list(c("X","Y","Z"), c("A","B","C"))) to name the columns and rows when creating the matrix.

Ex: > x <- matrix(1:9, nrow = 3, dimnames = list(c("X","Y","Z"), c("A","B","C")))

A B C X 1 4 7 Y 2 5 8 Z 3 6 9

Note to myself 2: The matrix is filled column-wise. This can be reversed to row-wise filling by passing TRUE to the argument byrow.

Ex: > matrix(1:9, nrow=3, byrow=TRUE) # fill matrix row-wise

[,1] [,2] [,3] [1,] 1 2 3 [2,] 4 5 6 [3,] 7 8 9

```
#Set number of rows and columns
matrix1 <- matrix(c(fb,twitter), nrow = 7, ncol = 2)

#Set the row names
rownames(matrix1) <- c("Sun","Mon","Tues","Wed","Thurs","Fri","Sat")

#Set the columns names
colnames(matrix1) <- c("Facebook","Twitter")

#print the matrix
matrix1
```

```
##       Facebook Twitter
## Sun         30      60
## Mon         43      32
## Tues        55      86
## Wed         89      44
## Thurs       71      21
## Fri         52      30
## Sat         42      28
```

ME2. Add a column for instagram, and save the result as `matrix2` .

	Sun	Mon	Tues	Wed	Thurs	Fri	Sat
Insta	45	68	25	76	50	41	44

Hint: You can either re-write your code to create a new matrix, or Google how to add a new column to a matrix. There is a function that will do this.

Your answer: We can use function rbind() to add the row to any existing matrix. cbind() to add a new column.

```
matrix2 <- cbind(matrix1, c(45,68,25,76,50,41,44))
colnames(matrix2) <- c("Facebook","Twitter", "Instagram")
matrix2
```

```
##       Facebook Twitter Instagram
## Sun         30      60        45
## Mon         43      32        68
## Tues        55      86        25
## Wed         89      44        76
## Thurs       71      21        50
## Fri         52      30        41
## Sat         42      28        44
```

ME3. Use the transpose operator `t()` so that the rows represent the social media outlet and the columns represent the days of the week. (You may want to look up this function by typing `?t`)

Save this as a matrix called `social`. It should have three rows (Facebook, Twitter, Instagram) and seven columns (for each day of the week).

Your answer:

```
social <- t(matrix2)
social
```

```
##           Sun Mon Tues Wed Thurs Fri Sat
## Facebook   30  43   55  89    71  52  42
## Twitter    60  32   86  44    21  30  28
## Instagram  45  68   25  76    50  41  44
```

ME4. Suppose we want to double the number of follows each day. Multiply your matrix `social` by 2 with regular multiplication (*). Does it work?

Your answer: Yes, it works.

```
social * 2
```

```
##           Sun Mon Tues Wed Thurs Fri Sat
## Facebook   60  86  110 178   142 104  84
## Twitter   120  64  172  88    42  60  56
## Instagram  90 136   50 152   100  82  88
```

ME5. Using your `social` matrix, get the Facebook follows from Wednesday. We can subset a matrix using the syntax: `my_matrix[row, column]`.

Your answer: 89

```
fb_wed <- social["Facebook", "Wed"]
fb_wed
```

```
## [1] 89
```

ME6. Use R code to print just the Monday reactions from your `social` matrix. This should be a column with 3 entries.

Your answer:

```
monday_likes <- social[, "Mon"]
monday_likes
```

```
##  Facebook   Twitter Instagram
##        43        32        68
```

ME7 OPTIONAL Intermediate Challenge. Remove the row with facebook follows from the matrix; the remaining matrix should only contain rows for twitter and instagram.

Your answer:

ME8 OPTIONAL Intermediate Challenge. (requires some knowledge of matrix multiplication): A marketer gets paid 5 cents per new follow on weekday and 8 cents per follow on weekends. Find the total weekly amount the marketer gets paid per social media outlet.

Your answer:

ME9 OPTIONAL Intermediate Challenge. How can you swap the order of the rows?

Your answer:

Data frames exercises

R has many built in data sets! Let's look at one of them.

DFE1. Look up the `iris` data set in R using the help function (or Google). What is this data? How many observations does it have?

Your answer: ris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica. The dataset contains 150 observations with 5 attributes named as Sepal width, Sepal length, Petal width, Petal length and flower type.

```
?iris
```

```
## starting httpd help server ... done
```

DFE2. Compare the results of calling `typeof()`, `class()`, and `str()` on the iris data set. Which one(s) are most useful?

Your answer: typeof() determines the (R internal) type or storage mode of any object. It is useful to see the data type of the data set. class() function is useful if we want to see the class of the data set. str() gives an overview of the data set. It is useful to see the structure of the data set.

```
typeof(iris)
```

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

```
class(iris)
```

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

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

DFE3. Print the first 3 rows of the iris data set.

Your answer:

```
head(iris,3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

DFE4. You can access just one column from a data frame using the `$` operators:

```
my_data$column_name
```

Use this to print just the “Species” column from the iris data set.

Your answer:

```
iris$Species
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor versicolor versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor versicolor
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  virginica  virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

DFE5. Write a logical expression to test whether the first plant's species in the Species column is "setosa". Then try to write a logical expression to test whether the first plant's species is setosa AND its petal length is longer than 1 cm.

Hint: You need to use the skills/code from the previous question.

Your answer: Yes, the first plant is setosa and its petal length is longer than 1 cm.

```
#First part of the question
iris$Species[1] == "setosa"
```

```
## [1] TRUE
```

```
#Second part of the question
iris$Species[1] == "setosa" && iris$Petal.Length > 1
```

```
## [1] TRUE
```

DFE6. OPTIONAL Intermediate Print only the rows from the iris data set for plants whose Species is "versicolor". Then write code to determine how many plants fit this condition.

Your answer:

OPTIONAL Intermediate List exercises (not graded; just for fun if you completed the intermediate reading)

LE1 OPTIONAL Intermediate. Suppose we have data on a book, including the title (“Game of Thrones”), year of publication (1996), and whether the series is complete (FALSE). What happens if we try to store these three pieces of information in a vector?

Your answer:

LE2 OPTIONAL Intermediate. Create a list to store the three pieces of information given. Use `list()` to create a new list and call it `book` .

Your answer:

LE3 OPTIONAL Intermediate. Edit your list `book` from the previous exercise so that the first element is called title, the second element is named year, and the third element is named is_complete.

Your answer:

LE4 OPTIONAL Intermediate. Use the book list, select just the title. How many ways can you think of to do this?

Your answer:

LE5 OPTIONAL Intermediate. Now select just the **first** available format (not the entire vector of all possible formats).

Your answer:

Hint for Matrix Exercise 1 (ME1)

Starter code to create the matrix:

```
fb <- c(30,43,55,89,71,52,42)
twitter <- c(60,32,86,44,21,30,28)

# modify to set the correct number of rows and columns
matrix1 <- matrix(c(fb, twitter), byrow = FALSE)

# add code to set the row names

# add code to set the column names
```