Lab 1a: Getting started with R and RStudio

Stat 131A, Spring 2019

Learning Objectives:

- Get started with R as a scientific calculator
- Understand pane layout of RStudio
- Distinguish between categorical and quantitative variables;
- Get to know markdown syntax
- Distinguish between categorical and quantitative variables

General Instructions

- This lab consists of two parts. The first part (first 25 mins approx) involves a first contact with R and RStudio. The second part (second 25 mins) involves answering questions about types of variables.
- Your GSI will guide/help you with the *knitting* process to get the html file that you will submit for the second part of this lab.

Part 1: R and RStudio

We are going to use mainly RStudio, but you should know that R is not the same as RStudio. The latter is just an integrated development environment (IDE). Launch RStudio and locate the following panes (or panels); their position will vary depending on your customization settings:

- Console (default location: entire left)
- Environment/History (default location: tabbed in upper right)
- Files/Plots/Packages/Help (default location: tabbed in lower right)

FYI: you can change the default location of the panes, among many other things: Customizing RStudio. If have no experience working with R/RStudio, you don't have to customize anything right now. It's better if you wait some days until you get a better feeling of the working environment. You will probably be experimenting (trial and error) with the customizing options until you find what works for you.

First contact with the R console

Let's start typing basic things in the *console*, using R as a scientific calculator.

For instance, consider the monthly bills of Leia (a stats undergrad student): cell phone $80, transportation $20, groceries $527, gym $10, rent $1500, other $83. You can use R to find Leia's total expenses:

```
# total expenses
80 + 20 + 527 + 10 + 1500 + 83
```

```
## [1] 2220
```

Often, it will be more convenient to create **objects** or **variables** that store one or more values. To do this, type the name of the variable, followed by the assignment operator `<-`, followed by the assigned value. For example, you can create an object `phone` for the cell phone bill, and then inspect the object by typing its name:

```
phone <- 80
phone
```

```
## [1] 80
```

All R statements where you create objects, "assignments", have this form:

```
object <- value
```

this means you assign a `value` to a given `object`; you can read the previous assignment as "phone gets 80".

RStudio has a keyboard shortcut for the arrow operator `<-`: `Alt` + `-` (the minus sign).

Notice that RStudio automagically surrounds `<-` with spaces, which demonstrates a useful code formatting practice. So do yourself (and others) a favor by ALWAYS surrounding an assignment operator with spaces.

- Make more assignments to create variables `transportation`, `groceries`, `gym`, `rent`, and `other` with their corresponding amounts.
- Now that you have all the variables, create a `total` object with the sum of the expenses:
- Assuming that Leia has the same expenses every month, how much would she spend during a school "semester"? (assume the semester involves five months).
- Maintaining the same assumption about the monthly expenses, how much would Leia spend during a school "year"? (assume the academic year is 10 months).

Object Names

There are certain rules you have to follow when creating objects and variables. Object names cannot start with a digit and cannot contain certain other characters such as a comma or a space. You will be wise to adopt a convention for demarcating words in names.

```
i_use_snake_case
other.people.use.periods
evenOthersUseCamelCase
```

The following are invalid names (and invalid assignments)

```
# cannot start with a number
5variable <- 5

# cannot start with an underscore
_invalid <- 10

# cannot contain comma
my,variable <- 3

# cannot contain spaces
my variable <- 1
```

This is fine but a little bit too much:

```
this_is_a_really_long_name <- 3.5
```

Functions

R has many functions. To use a function type its name followed by parenthesis. Inside the parenthesis you pass an input. Most functions will produce some type of output:

```
# absolute value
abs(10)
abs(-4)

# square root
sqrt(9)

# natural logarithm
log(2)
```

Comments in R

All programming languages use a set of characters to indicate that a specifc part or lines of code are **comments**, that is, things that are not to be executed. R uses the hash or pound symbol `#` to specify comments. Any code to the right of `#` will not be executed by R.

```
# this is a comment
# this is another comment
```

```r
2 * 9

4 + 5  # you can place comments like this
```

Case Sensitive

R is case sensitive. This means that `phone` is not the same as `Phone` or `PHONE`

```r
# case sensitive
phone <- 80
Phone <- -80
PHONE <- 8000

phone + Phone
```

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

```r
PHONE - phone
```

```
## [1] 7920
```

Getting help

Because we work with functions all the time, it's important to know certain details about how to use them, what input(s) is required, and what is the returned output.

There are several ways to get help.

If you know the name of a function you are interested in knowing more, you can use the function `help()` and pass it the name of the function you are looking for:

```r
# documentation about the 'abs' function
help(abs)

# documentation about the 'mean' function
help(mean)
```

Alternatively, you can use a shortcut using the question mark `?` followed by the name of the function:

```r
# documentation about the 'abs' function
?abs

# documentation about the 'mean' function
?mean
```

- Understand and learn how to read the manual (i.e. help) documentation

- Title
- Description
- Usage of function
- Arguments
- Details
- See Also
- Examples!!!

`help()` only works if you know the name of the function your are looking for. Sometimes, however, you don't know the name but you may know some keywords. To look for related functions associated to a keyword, use double `help.search()` or simply `??`

```
# search for 'absolute'
help.search("absolute")

# alternatively you can also search like this:
??absolute
```

Notice the use of quotes surrounding the input name inside `help.search()`

Your turn

- Take your objects (i.e. variables) `phone`, `transportation`, `groceries`, `gym`, `rent`, and `other` and pass them inside the *combine* function `c()` to create a vector `expenses`

```
expenses <- c(phone, transportation, groceries, gym, rent, other)
```

- Now, use the graphing function `barplot()` to produce a barchart of `expenses`:

```
barplot(expenses)
```

- Find out how to use `sort()` to sort the elements in `expenses`, in order to arrange the elements in expenses in decreasing order.
- Find out how to use `sort()` and `barplot()` to produce a bar-chart with bars in decreasing order.
- Optional: see if you can figure out how to display the names of the variables below each of the bars.

Introduction to Markdown and `Rmd` files

You will be using a specific type of source files known as *R markdown* files. These files use a special syntax called **markdown**.

In the menu bar of RStudio, click on **File**, then **New File**, and choose **R Markdown**. Select the default option (Document), and click **Ok**.

Rmd files are a special type of file, referred to as a *dynamic document*, that allows to combine narrative (text) with R code. Because you will be turning in most homework assignments as `Rmd` files, it is important that you quickly become familiar with this resource.

Locate the button **Knit HTML** (the one with a knitting icon) and click on it so you can see how `Rmd` files are renderer and displayed as HTML documents.

R markdown (`Rmd`) files use markdown as the main syntax to write content. It is a very lightweight type of markup language, and it is relatively easy to learn.

Part 2: Variables

- Write your solutions in an `Rmd` (R markdown) file.
- Name this file as `lab01a-first-last.Rmd`, where `first` and `last` are your first and last names (e.g. `lab01a-gaston-sanchez.Rmd`).
- Knit your `Rmd` file as an html document (default option).
- Submit your `Rmd` and `html` files to bCourses, in the corresponding lab assignment.

Problem 1

Identify the type of variable (qualitative or quantitative) for the list of questions from a survey applied to college students in a statistics class.

a. Name of student.

b. Birth date (e.g. 10/21/1995).

c. Age (in years).

d. Home Address (e.g. 1234 Shattuck Ave).

e. Telephone number (e.g. 510-123-4567).

f. Major field of study.

g. College year-grade: freshman, sophomore, junior, senior.

h. Score on midterm test (based on 100 possible points).

i. Overall grade: A, B, C, D, F.

j. Length of time—in minutes—to complete Stat 131A final test.

k. Number of siblings.

Problem 2

Choose an object (any object, e.g. animals, plants, countries, institutions, etc) and come up with a list of 14 variables: 7 quantitative, and 7 categorical.

Problem 3

Consider a variable with numeric values describing electronic ways of expressing personal opinions: 1 = Twitter; 2 = email; 3 = text message; 4 = Facebook; 5 = blog. Is this a quantitative or a qualitative variable? Explain.

Problem 4

For each research question, (1) identify the individuals of interest (the group or groups being studied), (2) identify the variable(s) (the characteristic that we would gather data about), and (3) determine whether each variable is categorical or quantitative.

a. What is the average number of hours that community college students work each week?

b. What proportion of all U.S. college students are enrolled at a community college?

c. In community colleges, do female students have a higher GPA than male students?

d. Are college athletes more likely than non-athletes to receive academic advising?

Problem 5

If we gathered data to answer the previous research questions, which data could be analyzed using a histogram? How do you know?

Don't forget to submit your `Rmd` and knitted `html` file to bCourses.