LAB 1 - AN INTRODUCTION TO R

MATH 1170

20 AUGUST 2018

In the first week, we'll establish basic proficiency in the tools needed for lab for the duration of this class, the primary of which is the R programming language. By the end of the lab, you should be able to[1]

- Open R and/or RStudio
- Use R to perform arithmetic
- Define and store variables
- Define vectors and plot functions
- Save and submit your work
- Save, edit, and run R code to perform simulations

[1] if for whatever reason, you know all of this already, feel free to skip ahead!

history lesson: in prehistoric times (1970s), Bell Labs created a series of languages for statistics, including *S*, *Old S*, and *New S*. Eventually they decided to get more creative with the naming scheme. What comes after S? **R**!

Opening R and RStudio

You have two options: use R on the lab computers or install R on your own laptop, the latter of which is strongly encouraged.

Lab Computers

The only tricky thing about this option is logging in. If your machine seems to be asleep, yell at it[2] and jiggle the mouse to wake it up. Next, you must decipher the puzzle of your login name.

[2] this probably won't actually help

All login names from classes begin with "c-". The rest of your **username** is determined by the following the recipe of

```
c-(first letter of last name)(last letter of last name)(first letter of first name)(middle initial)
```
[3]

[3] if you don't have a middle name, your login will only have 3 initials, i.e. `c-xxx`.

Your initial **password** follows a similar recipe, but just removes the "-c" and adds the last 4 of your uID, resulting in

```
(first letter of last name)(last letter of last name)(first letter of first name)(middle initial)(last 4 of uID)
```

For example, if you were, say, Donald J. Trump and had a uID of 06781234, your login name would be `c-tpdj` and password would be `tpdj1234`.

The lab computers already have R and RStudio installed, so you're now good to go.

Laptop Installation

This is also relatively straightforward, but a little trickier. Ultimately, you need to install *both* R and RStudio. First, to install R, navigate to `https://cran.r-project.org/` and hit "Download R for

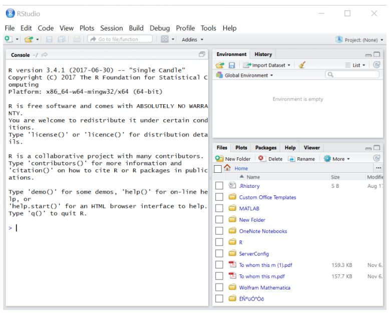
…" for your appropriate operating system. For Macs, this involves navigating to `R-3.5.1.pkg` and downloading it and for Windows `R-3.5.1-win.exe`.[4] Follow the instructions and everything should go okay.

[4] if you're using Linux, you probably don't need these instructions

Next, install RStudio, which can be found at `https://www.rstudio.com/products/rstudio/download/`. Specifically, you'll want to use the files under the "Installers for Supported Platforms" section for whatever your appropriate operating system is. Again, follow the installer and go with whatever the default options are.

RStudio

R is the **language** we'll be using and RStudio is the **editor**. Basically, RStudio is just a convenient way of programming in R. Whether you're using the math computers or your own laptop, open RStudio and you should see something like the figure to the right. If you do, celebrate.



```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical C
omputing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRA
NTY.
You are welcome to redistribute it under certain cond
itions.
Type 'license()' or 'licence()' for distribution deta
ils.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in public
ations.

Type 'demo()' for some demos, 'help()' for on-line he
lp, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Name	Size	Modifie
.Rhistory	5 B	Aug 1
Custom Office Templates		
MATLAB		
New Folder		
OneNote Notebooks		
R		
ServerConfig		
To whom this m (1).pdf	159.3 KB	Nov 6,
To whom this m.pdf	157.7 KB	Nov 6,
Wolfram Mathematica		
ÉÑ°üÓ°Òô		

This program looks overwhelming at first, but we'll talk about each of the windows when they come up. For now, focus on the window that says **Console**, probably in the bottom left. It should say something like:

```
R version 3.4.1 (2017-06-30) -- "Single Candle"
...
>
```

fun fact: each version of R references the cartoon *Peanuts*. This version is called "Single Candle".

The > means that R is waiting for you to do something!

WHAT'S THIS?
I HAVE HEARD THAT IT IS BETTER TO LIGHT A SINGLE CANDLE THAN TO CURSE THE DARKNESS
THAT'S TRUE..ALTHOUGH THERE WILL ALWAYS BE THOSE WHO WILL DISAGREE WITH YOU...
YOU STUPID DARKNESS!
SCHULZ

Arithmetic in R

In general, computers are really stupid[5] and will do **exactly** as you tell them. This sounds great, but ultimately is a blessing and curse. We'll first warm up by giving R some basic orders in the console. [6]

In their stupidity, computers (and R) love doing basic arithmetic. Try entering the following commands[7]

```
> 2-3
> 2^23
> 22*33
```

[5] don't tell them this

[6] later we'll write more complex programs that require more than one command at a time, and that's what the top left window in RStudio is for.

[7] and race R to compute them

don't be intimidated by the colors I've used here. This is called *syntax highlighting*, which simply to make our life easier. Numbers will always be in blue. Commands or "special words" that R knows will always be in green.

R also has a lot of built-in mathematical functions.

```
> sin(3)
> exp(3)
> log(3)
```

when we type `log(3)` in R, it uses the natural log, $\ln(3)$

Storing Variable Names

Although you should be convinced that R is really good at arithmetic, you shouldn't be impressed yet. In general, numbers represent something, so we'll give them a label. Say, I had 5 cats and I wanted to store this quantity in R. What is a natural name for this variable? Let's say `cats`. How do we tell R to store 5 under the name `cats`?

```
> cats <- 5
```

After you type this, in the top right corner of RStudio under the **variables** section, you should see that R is now storing the value 5 under the name `cats`. We can also see this via the console, where if we type the name of our variable, it spits out the value

```
> cats
[1] 5
```

[8] ignore the `[1]` in the output here. One of (many) weird quirks of R.

[8] Once a variable is assigned, it is not set in stone. We can happily override it with a new value

```
> cats <- 3
```

Now, the value of `cats` is 3. Assigning variables can also be done with other variables:

```
> dogs <- 1 #simply means we only have 1 dog
> total <- dogs + cats #add two variables to make a new
```

here I've used a comment, shown in gray. The general syntax is `#comment`, where anything you write after the #, R ignores and is just there for humans.

One last thing to note is that we can also use = to assign variables, so

```
cats = 5 #assigns 5 to 'cats'
```

however, this is somewhat confusing because this is **not** the same as using ==

```
cats == 5 #checks if cats and 5 are equal
```

For this reason, I'll use the <- operator for assignments. Ultimately, a good thing to remember: the arrow always points to the thing it is assigning. [9]

[9] for example, a<-b means a becomes the value stored in b.

Plotting functions

Some functions are horrible, gross, and too confusing for our tiny human brains to sort out. However, R is amazing at dealing with these. Take, for instance,

$$y = f(x) = e^{\sin\left(x^2-3\right)}$$

.

Because R is stupid, it doesn't know what x is in our plot. Let's suppose that we want to plot our function for $x = 0$, $x = 1$, all the way to $x = 10$.

We can use the `c` command, short for *concatenate*[10] to create a list or **vector**[11] of numbers

[10] this is just a fancy word for stick stuff together

[11] this is just a fancy word for list

```
> x <-c (0,1,2,3,4,5,6,7,8,9,10)
```

Now we can calculate the y values for these values of x.

```
>y <- exp(sin(x^2-3))
```

note what R is doing in this line. It knows that x is a vector and it spits out a vector by doing stuff to each element. How smart.

Then we can use the built-in plot function. The plot function requires at least 2 inputs: a list of x values, and a list of y values. We (conveniently) have these defined as x and y.

```
> plot(x,y)
```

Now, the bottom right corner of RStudio should display something vaguely resembling a plot. It certainly doesn't very look pretty, though.[12] This is because we plotted this function for pretty stupid x values. We really don't want to just use $0, 1, 2, 4, \ldots, 10$ but a lot of numbers in-between. To do this, we use the `seq` command, which gives us a *sequence* of numbers

[12] beauty is in the eye of the beholder

```
> x< -seq(0,10,0.01)
```

the general syntax for `seq` is `seq(low, high, step)`.

Take a look at the result by typing

```
> x
```

Now, try to plot our function again.[13] Did you get an error? This is because the vectors x and y are of different lengths. We have to update y the same way as before to calculate it at all of our intermediate points. Re-entering the plot command should now display a lot more points that seem to lie on a curve. R will connect these points with lines for you if you enter

[13] *pro tip:* The up arrow will show you your previously entered commands!

```
>plot(x,y,type='l')
```

`type='l'` is just short for "line", instead of the default scatter plot

You can add a title and label the x and y axis, like so.

```
>title(main="This sweet function",xlab="x",ylab="y")
```

here we've introduced our last color. *Strings are in purple.* You can think of a string as a collection of letters (and spaces), which is very different than anything to do with numbers but strings like `'abc123'` can have numbers in them. Confusing, I know.

Saving Plots

R can export graphics in many different formats, but I'm going to recommend we save our figures as `png` files. There is a way to do this via the console[14] but using RStudio, you can just click the "Export" button above the plot and then choose "Export as Image". This is way easier and strongly encouraged. From here, you can paste the image into, say, Microsoft Word or any other word processor.

[14] Google it (and anything else), if you want to know

Running R code

As we mentioned earlier, some programs are too complex to be just typed into the console line-by-line. I have provided a file named `number_game.r`. Now, we use the final window of RStudio by going to "File, Open" and then find the file you just saved. Now you should be able to see the code for this program. You can run it by pressing the "Source" button in RStudio. If you do so, you should see

```
> source('(path of the folder you saved)/number_game.r')
Guess a number between 0 and 100.
Enter an integer:
```

In other words, R ran *all* of the commands in the file at once. By doing a sequence of stupid things, R can do very smart things, like this game! Try to win the game.[15]

[15] it's very easy to cheat if you've been paying attention to the other stuff going on in RStudio

We'll in general write `.r` files with a sequence of R commands and edit them directly in RStudio, which is convenient because we can also immediately run them.

Assignment for this week

1. Plot a different scary function than the one we did in the lab:

$$y = f(x) = \frac{\sqrt[3]{x}}{\log(x+3) - 2}$$

between -2 and 2 in same way we plotted the function in this lab. Give your plot axis labels and a title. Even better, you could try learning (via the internet) how to customize your plot with colors, fonts, and more![16]

[16] one thing R is "famous" for is having great packages to do stuff for you. Basically, people share their R code and you can use it. I encourage you to look up one called `ggplot2` for pretty plots if you're feeling ambitious

2. Although `numbers_game.r` has a scary portion, where it makes sure you input nice things, you can ignore this and focus on the second half of the code. Read over and see if you can figure out roughly what it is doing.

   (a) Modify the original code so the computer only guesses between 0 and 50.

   (b) Modify the original code so that if your guess is above 100, the computer prints an angry message of your choosing (chance to be creative!).

Submissions

To submit, save all of your work (figures, code, output, your feelings) in a **single** PDF file. You can generally do this by pasting everything into a Word document. For my sanity, please use following naming convention:

```
lab#_(lastname)(firstinitial).pdf
```

so The Donald's first submission would be, for example, `lab1_trumpd.pdf`.

Every week's lab is due at the beginning of the lab. Late submissions for any reason will not be accepted.