

# Activity 1: Intro to R and GitHub

*Heather Kropp*

*GEOG 331: Environmental Data Science, Colgate University*

## Instructions

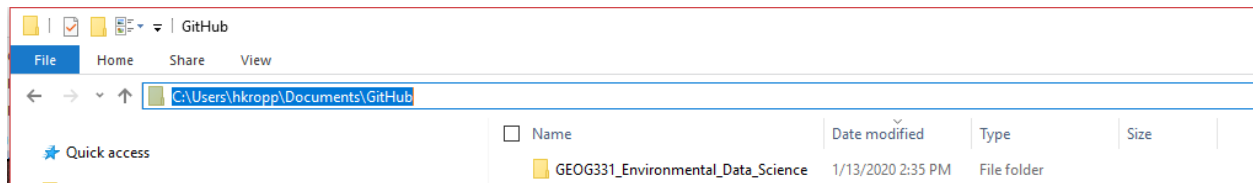
There are 4 questions to this activity. Save your answers in word document that you will hand in on Moodle using a .pdf extension. Keep your script file in your GitHub folder and make sure that all changes are pushed to GitHub. You will include a link to this file as a part of your final question in this activity.

## Learning objectives

1. Learn about file systems
2. Learn how to do version control
3. Introduction to R

## Files in computers

An essential part of working with data is thinking about how its stored on the computer. Chapter 1 and 2 cover some of the fundamentals of data in a computing environment. Here we will focus on how you will interface with data in R. In this class, you will use files from the internet, locally, and on the Geography department server. A good first step is to understand how to tell R to find a file on your computer or the server. Follow the instructions on moodle to map the Geography server (*geogsv02*) to your computer using the y drive. Click on the folder icon in the start bar to use the windows explorer to navigate around folders on the computer. Click on the Documents folder in the left Navigation bar. Right click to make a new folder and title it *GitHub*. We don't always have to click to find things in folders. Computers can reference where files are located using what's called a **file path**. You must reference where a file is located from the drive on the computer to each successive folder all the way to the folder it is located in. You can click on the drop down at the top of windows explorer to display the full file path as seen below.



The *GitHub* folder I just made has the file path “*c:\Users\hkropp\Documents\GitHub*”. All Colgate Colgate computers will automatically create a Users folder with your Colgate handle from your login and email so your path will be different than mine.



### Question 1:

Explore other folders on your computer. Keep track of the file paths. Describe the difference between each path.


## Intro to Git

Git and the user interface GitHub allow you to keep track of all changes to your code. You just have to indicate points where you want to save the state of your code and give it a tag. Here, we'll go through the basics of Git and GitHub. A **repository** groups together code and any documentation files for a users project. You can keep track of any changes to files in each repository to document each version of your code. **NEVER, NEVER, NEVER, put username and password info in files on GitHub (can't emphasize this enough).** This is the easiest way to get hacked.

Go to github.com and sign in. Click on the Repositories tab and click the New green button. Create a repository called *GEOG331* (yes, exactly that name with no spaces). Make it public and make sure the README box is checked as shown below.


### Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner:  kroppheather / Repository name \*:

Great repository names are short and memorable. Need inspiration? How about [stunning-goggles?](#)

Description (optional):

☒  **Public**  
Anyone can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

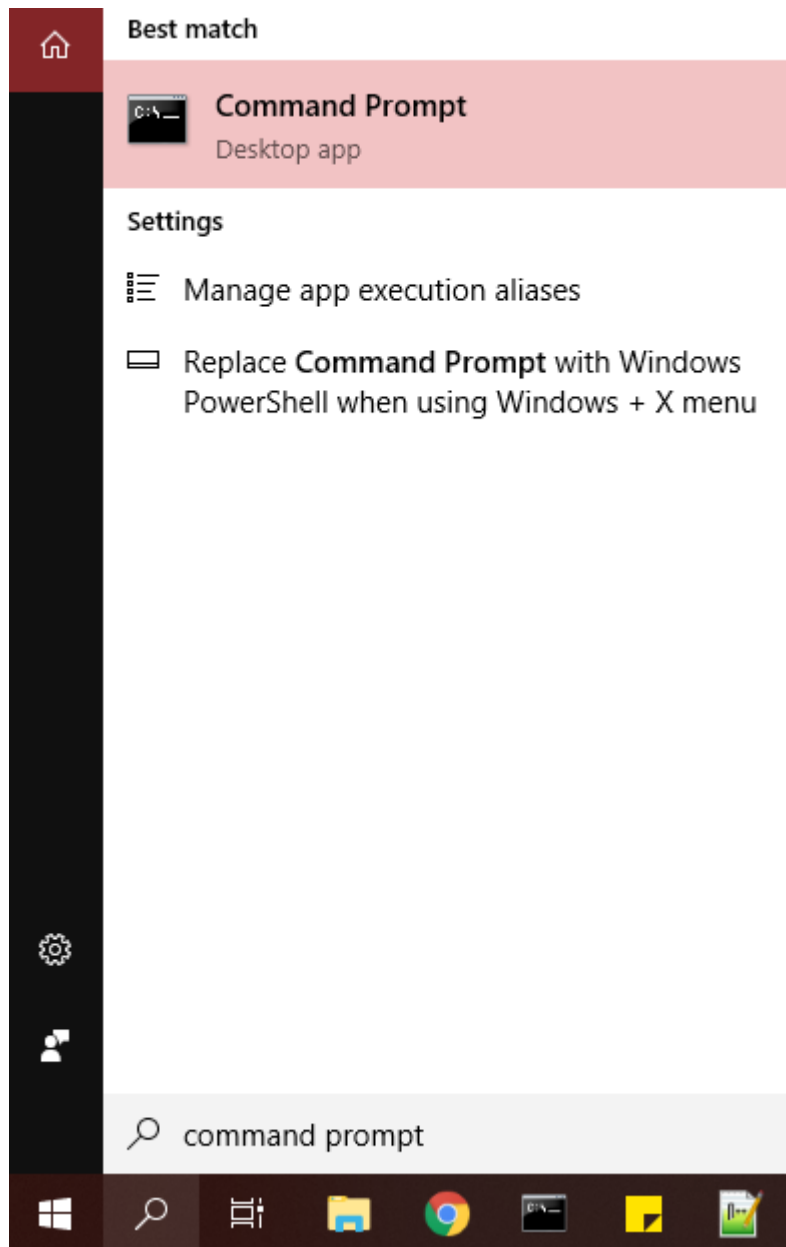
Skip this step if you're importing an existing repository.

☒ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** | Add a license: **None** ⓘ

[Create repository](#)

Once you click on the create repository button, we will want to clone the repository in your local computer. Click the green Clone button and click on the clipboard to copy the link. The universal way to interface with git and communicate with github is to use the command line. Search for the command line (as shown below) and click on it.



You will see an interface that has a drive name. It may default to your student drive (n:). If so type c: and hit enter to change it to your computers drive. Now you'll want to always work from repositories from a local folder on your computer. One of the benefits of github is that you can load a repository on any computer, push changes back to the github cloud, and then load your repository on any computer. I always create a GitHub folder (like you did above!) in the Documents folder on any computer I work from. You'll want to navigate to that folder you created in the command line so you can tell git to work in that folder. You can change the directory with the **cd** command and enter your filepath to your GitHub folder. You will see your new file path indicated on the left of the > sign. See below for an example.

```
CA. Command Prompt
Microsoft Windows [Version 10.0.17134.753]
(c) 2018 Microsoft Corporation. All rights reserved.

N:\>c:

C:\>cd c:\Users\hkropp\Documents\GitHub

c:\Users\hkropp\Documents\GitHub>
```

Git is a program that runs from the command line and is already installed on the computer. Type `git` and hit enter. You will see a list of options for running functions in git. You want to clone your *GEOG331* repository to your GitHub folder. Type `git clone` then right click to paste the repository URL into the command line and hit enter (Pay attention to spaces here. There's one after `git` and `clone`).

```
c:\Users\hkropp\Documents\GitHub>git clone https://github.com/kroppheather/GEOG331.git
Cloning into 'GEOG331'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.

c:\Users\hkropp\Documents\GitHub>
```

You will see some print out text. Go to the GitHub folder in windows explorer. You will see a new folder called *GEOG331*. This is your repository! Any files you save, folders you make, or changes in files will be tracked. Make a folder called *activity1* in you *GEOG331* folder. Let's create an R file to save in there. Open notepad++ (search the start menu). Click the file with the green plus button. Save your file as *activity1\_script.r* in your *GEOG331\activity1* folder making sure that you select the R programming language file extension. In this file type the following:

```
# Activity 1
print("hello R world")
```

Click the save button. Now we need to tell git to document these changes. Go back to the command line. You will need to make sure you are now in the *GEOG331* repository to document these change. Change your directory to the repository folder. Mine looks like this.

```
c:\Users\hkropp\Documents\GitHub\GEOG331>
```

Now the next steps will label your changes for tracking. You need to follow all three steps in order for your changes to be properly tracked.

1. Type **git add -A**
  - This will tell git to add changes to all files for tracking.
2. Type **git commit -m "description of changes"**
  - This command helps label all of your changes
3. Type **git push**
  - This command sends the changes to github
  - You will have to enter your username and password for GitHub here

```

c:\Users\hkropp\Documents\GitHub\GEOG331>git add -A

c:\Users\hkropp\Documents\GitHub\GEOG331>git commit -m "started a script for activity 1"
[master e8ad999] started a script for activity 1
 1 file changed, 2 insertions(+)
 create mode 100644 activity1/activity1_script.r

c:\Users\hkropp\Documents\GitHub\GEOG331>git push
fatal: HttpRequestException encountered.
An error occurred while sending the request.
Counting objects: 4, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 384 bytes | 384.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/kroppheather/GEOG331.git
   bb62613..e8ad999  master -> master

```

Go to the repository page on GitHub. You should see your changes and you can view your scripts from the browser. If you ever need to pull changes from a repository that aren't on an existing computer, you can type **git pull** and it will update the files in your repository.

**Congratulations! You can now use version control!**



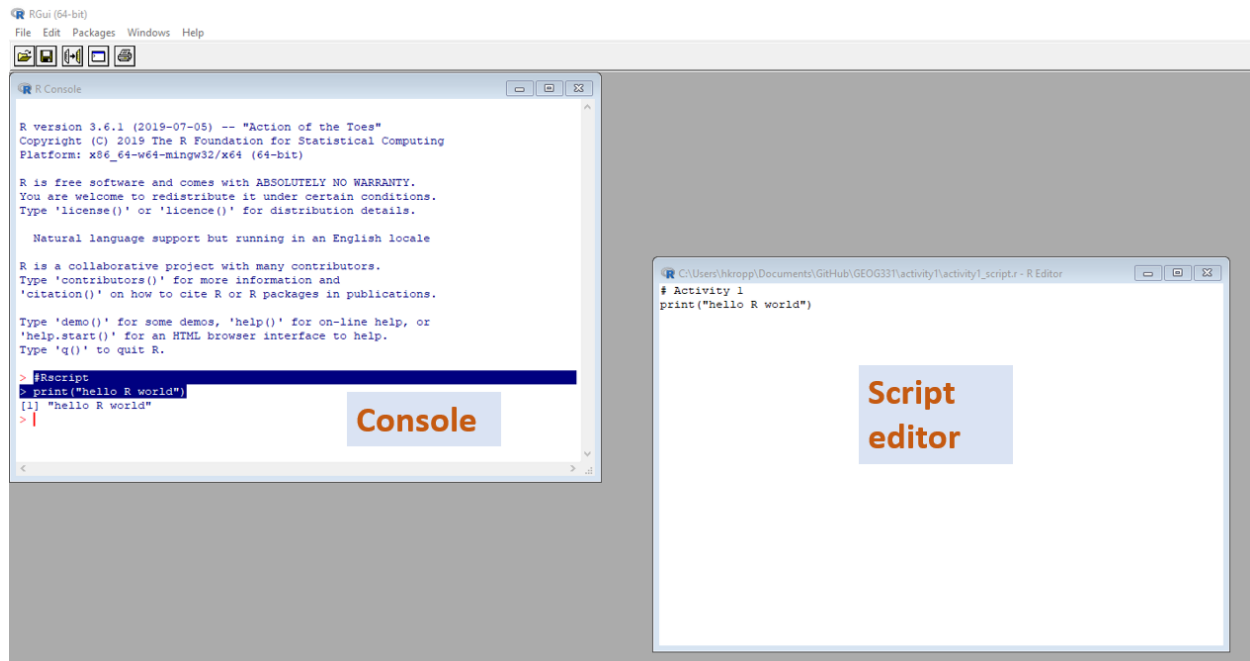
### Question 2:

Take a screenshot of your repository with your new commit. Write a brief description of the three steps to committing a change.

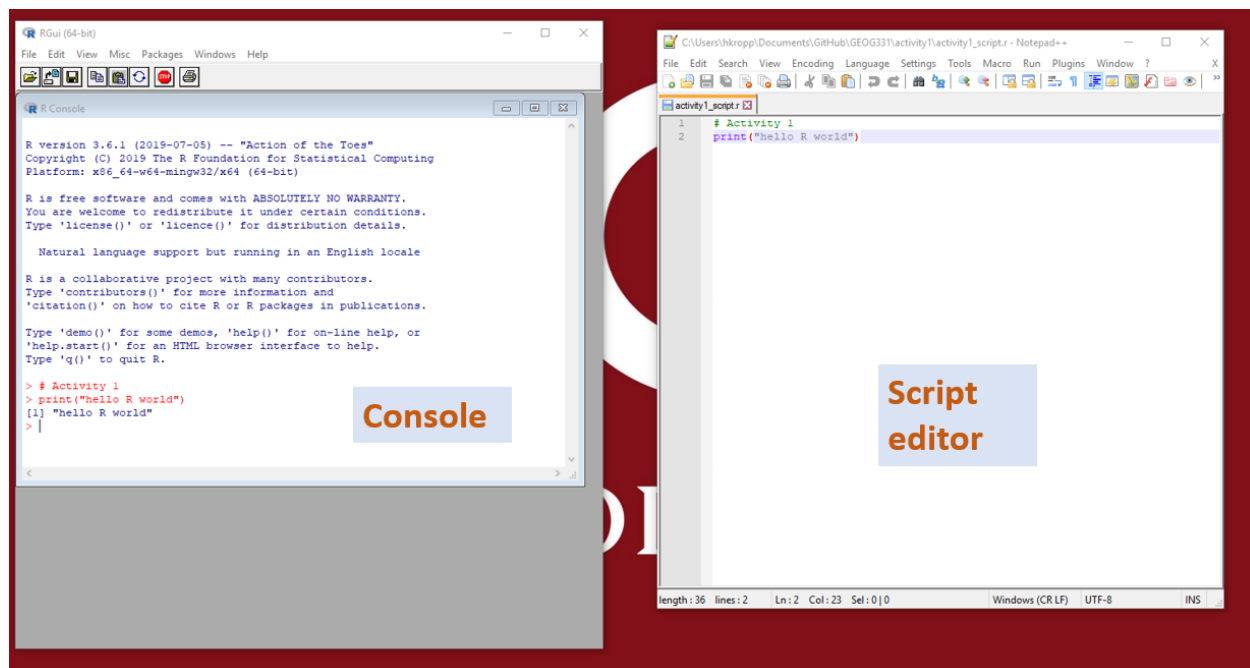
## Introduction to R

R is essentially a giant calculator with many options for plotting and built in functions. You just started writing your first script for R. As we discussed in class, R scripts allow you to save code in a text file to run in R. R scripts all have a `.r` extension. By itself, the script will do nothing. You need to actually tell R to run your code. The **console** runs your R code. It's the calculator! You can type code into the console and it will run. However, you won't be able to access that code later. That's why we use scripts. Anything that has run in the console is a part of your **working environment**. The working environment saves your calculations in the computer's temporary memory. When you close R, your working environment will disappear. However, when you write good script, you can rerun the code in the script and all of the items in your working environment will be the same as when you last used R.

There are many different ways to interface with R. You can even run scripts from the command line! I will work interchangeably between different R interfaces. You can choose how you would like to work with R. The base R program has an option for writing scripts and sending lines of code. However, the script editor doesn't have any formatting help or color coding of different types of code. This is often not a user friendly option.



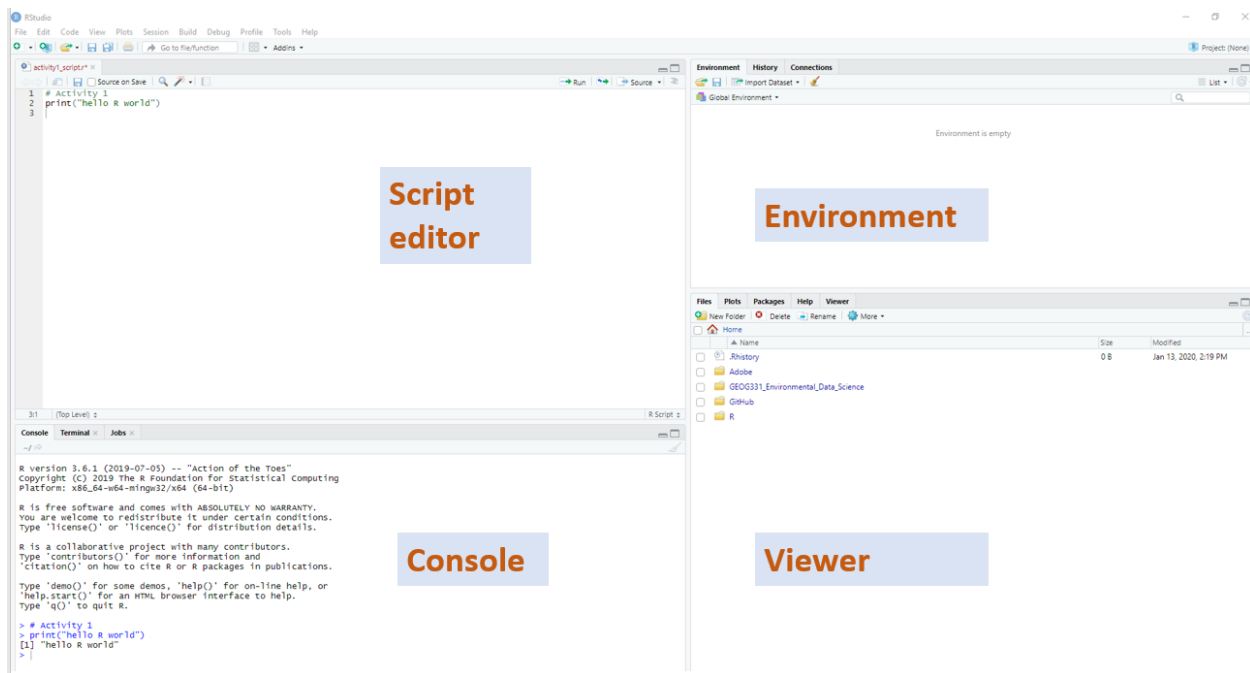
Another option is to use a code editor like notepad++ and send your code from your script to R. For these computers you have to copy and paste chunks of code. The programs NppToR that will send your code to R using designated shortcut keys. To use NppToR, just open the program and Notepad++. Hit F8 and you can send your code directly to R. It will even open R!



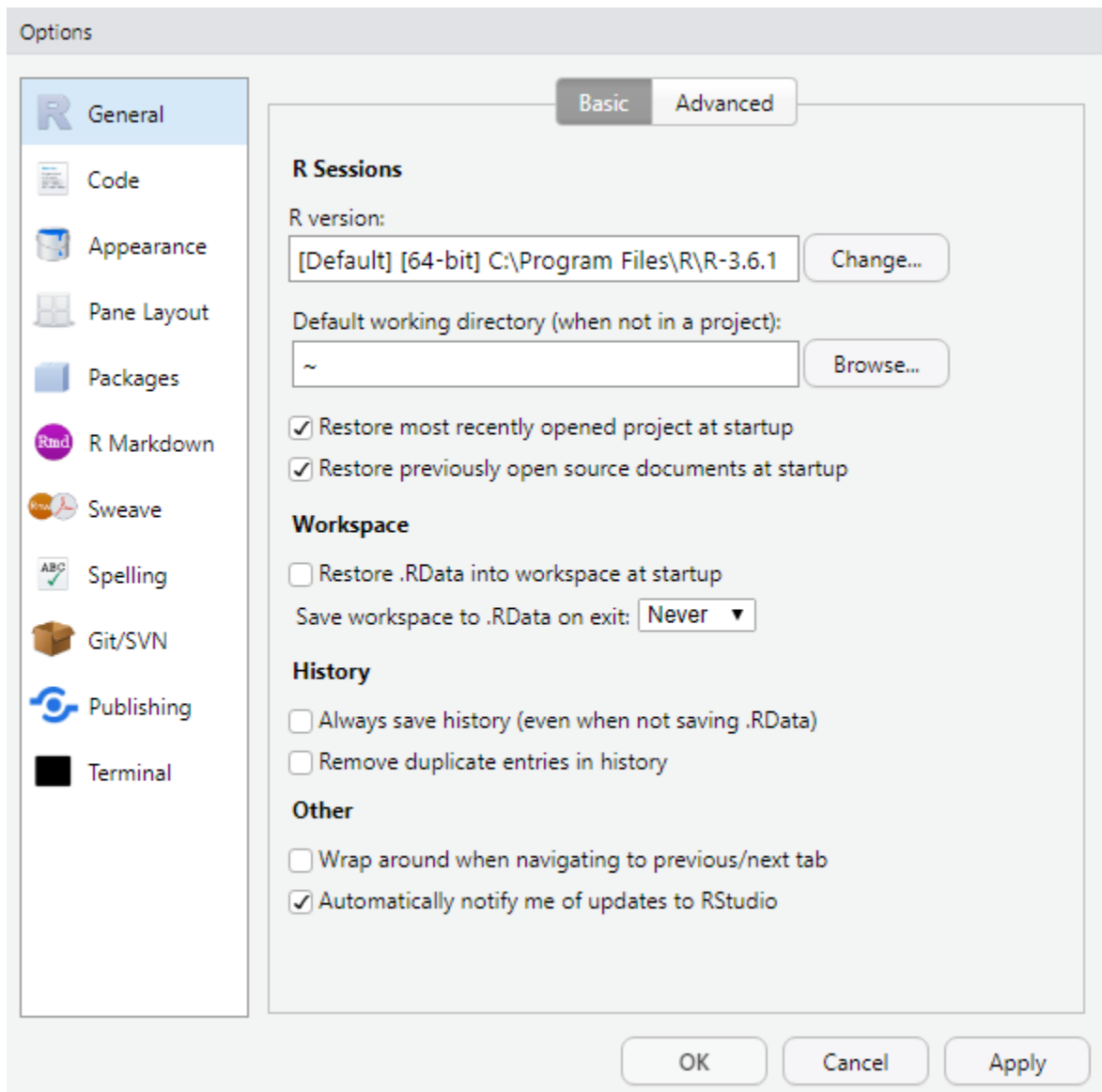
The option that many students prefer to use is Rstudio. Rstudio provides an interface for scripting, the console, and it has additional windows for visualizing objects in your working environment and viewing plots. R studio can also allow you to view data in cells that look similar to excel.

Lets open up the script that you started in the Git activity in Rstudio. In the menu, click File>Open. Navigate to your *GitHub\GEOG331\activity1* folder and choose your *activity1\_script.R* file. Your script will open in the upper left. You can run lines of code with the run button or by hitting control and enter at the

same time.



If you run your code that you wrote, you'll see "hello R world" printed in the console like my image above. Before you do anything else, you need to change some settings in Rstudio. These settings can lead to confusing errors because they save and reload old workspaces. You should be writing good scripts that easily rerun, but these settings can prevent you from checking that until after you turn in your assignment!! Furthermore they lead to a lot of confusion. I estimate that about 10% of the time I help troubleshoot problems with students, these settings are the problem! Go to the menu bar and choose *Tools>Global Options*. In Workspace, uncheck the Restore .Rdata option, change the save workspace option to Never. In History, uncheck the Always save history option. You will have to restart your R session to load these changes. Your options should look like my screenshot below.



We'll start working with R more next week and read in data. However, let's get a feel for how R works more.

Since R is just like a sophisticated calculator, you can read in numerical operations and will get the calculations as output. Type a few different operations like the one below. Note I've included both the code and outputs here for an example.

```
#remember this is a comment so it won't do anything. R just ignores it.
#you need to use these to document code and write notes
#6 raised to the 6 power
6^6
```

```
## [1] 46656
```

```
# 5 plus 210
5+210
```

```
## [1] 215
```

```
#3 minus 10
3-10
```



```
## [1] -7
```

There's a few things to pay attention to in the console. The > symbol always indicates a new line of code. The + symbol means your line is continuing. Your results will have numbers in brackets to describe the output. [1] Means I have a vector and my output starts on the first element of my vector (in this case its a vector of one).

You can also give objects names. Below is an example where I know I will want to use the number 2446 many times so I will give it a shorter name. R conducts all operations on that number. The <- symbol means you are assigning an object a **\*\*variable name\*\_\*\***

```
#name my number  
a <- 2446  
#multiply my number by 5  
5*a
```

```
## [1] 12230
```

```
#divide my number by 2  
a/2
```

```
## [1] 1223
```

R automatically does vector operations (think about code differently you Python and C coders!). You can assign two numbers to a variable name and do math operations on them. First you have to let R know you are making a vector of multiple numbers using c(...)

```
#make a vector of numbers  
b <- c(2446,40,10000)  
#divide all numbers by 2  
b/2
```

```
## [1] 1223 20 5000
```



### Question 3:

Practice writing mathematical operations in R and making vectors using variables names. Run 3 calculations, at least one with a vector with 4 numbers and a variable name



### Question 4:

Add all changes, commit, and push to GitHub. Paste the URL to your completed R script in your repository as the answer to this question.

**Great job getting started in R!!!** Next week we will pick up with coding in R.