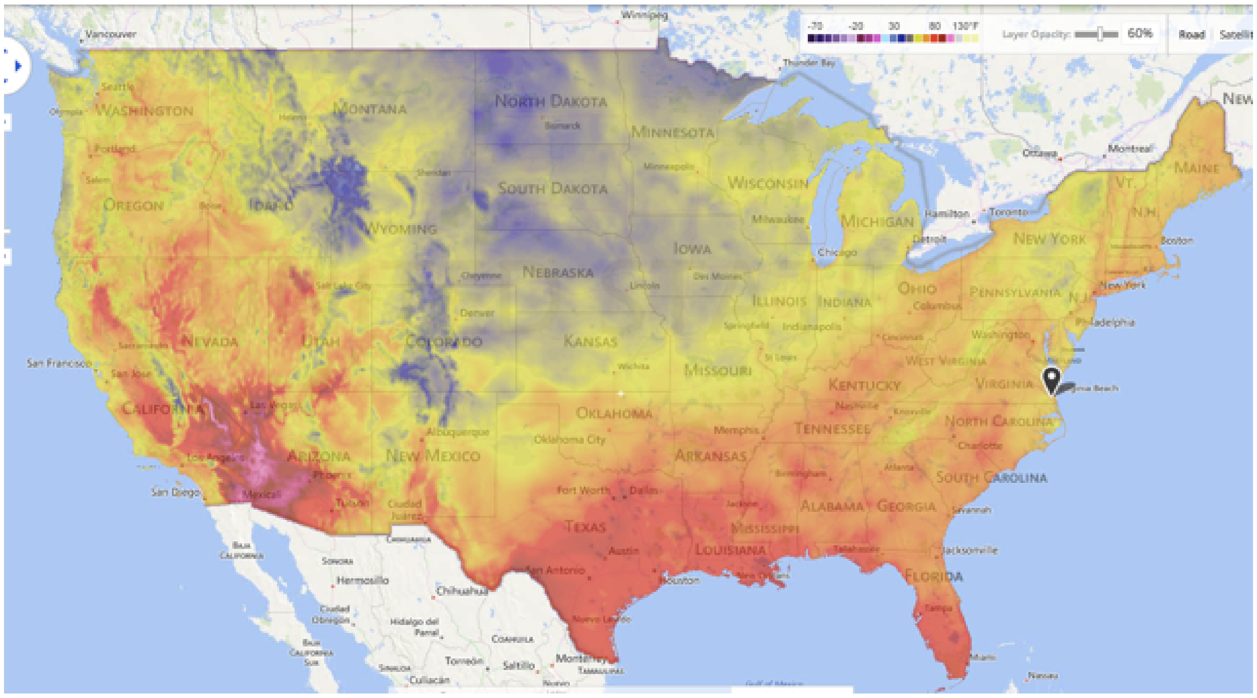
Introductory Lab. A Gentle Introduction to Data

Due by noon on Friday, August 28th



"Everything is related to everything else, but near things are more related than distant things."

Tobler, 1970

Laboratory in Brief:

The purpose of this laboratory is to introduce you to *the world of tomorrow*. More directly, you'll be learning a few of the fundamentals you'll need later in the course. This is **NOT** a programming course (I promise!), but you'll need to at least be able to type a few things in here and there that may - deceptively - look a little bit like programming. To get you started off, we're going to very, very slowly introduce you to a few concepts used in the statistical language *R*.

Goals of this Laboratory:

1. To learn how to install and load "packages" - magic tools other people have written that will let us do incredible things.
2. To learn how to make a map using *R*! You'll be the envy of your friends.

Session 1: Wednesday, August 26th

Acquire all the Data! Kind of...

1. First lets open RStudio. You'll find it as a link in your start menu (if you're really brave, you could even install this at home!).
2. One of the most basic features of R is a package. A package is an external source of code that can be installed into your local system that has specific features for executing a particular function.
3. Using a package is a two-step process. First, just like any other app, you need to install. Second, you need to "load" it inside of RStudio (just like you have to launch an app every time you use it).
4. Installing the package only needs to be done one time, while loading it needs to be done each time we want to use it in R. To do this first we install the package by typing the following command:

   ```
   install.packages("ggmap", dependencies=TRUE)
   ```

   This particular package, ggmap, will let us make our map! Adding the command `dependencies=TRUE` tells R to install all the other applications `ggmap` needs to function properly. You don't *always* need this, but better safe than sorry.
5. Once you have installed the package `ggmap`, next you will need to load it into R. To do this type the following command (and then, if you haven't figured it out yet, hit enter).

   ```
   library(ggmap)
   ```

   and your package is now active and ready for use.

6. I know that was exciting, but now we're headed to the *really* fun part. Just like a word document, you can save all the things you do in R. To do this, lets create a new file (File -> New File -> R Script). Now, click file -> Save. Choose your H drive and save it as something you'll remember later!

7. Now it's time to get coding. Let's go *CRAZY*, and type the following into our new document (R Script):

   ```
   1 + 1
   ```

   Highlight that entire thing (yes, ALL of 1+1), and then click the 'run' button in the upper-right hand corner. Just like that, we've tricked you into becoming a programmer.

8. Now, lets make a "quick map" in R by assigning a place to a variable. Since we are going to create a map of our favourite University, I will use one of my favourite places.

   ```
   gt = 'Georgia Tech'
   ```

   To understand what we have done, simply type `gt` and have a look at the output from R.

9. Now lets execute our first function in R. The package `ggmap` has a quick map function that takes a name and attempts to figure out what you are interested in mapping, and then creates a map for you. To make this work, your R document should look like the below:

   ```
   library(ggmap)
   gt = 'Georgia Tech'
   qmap(gt, zoom = 14)
   ```

   We can also play around with the qmap function by adding some other commands, known as "arguments". Run the below three examples one at a time to see how this can work.

   ```
   Example 1: qmap(gt, zoom = 13, source = "osm")

   Example 2: qmap(gt, zoom = 15, source = "stamen",
   maptype = "watercolor", crop = FALSE)

   Example 3: qmap(gt, zoom = 16, source = "stamen",
   maptype = "toner", crop = FALSE)
   ```

10. ***Deliverable Number 1: Create a map of your home town, favourite school, or university, and copy it into a word document - full points will be awarded for using a different type of map source or map type.***

11. NOTE: The help pages in R are notoriously difficult to read, but once you get the hang of them, they can be informative. In order to find out more information about a particular function type `?qmap` and the help page for that function will pop up.

12. Next we will use the `get_map` function in `ggmap`. This function works similar to the qmap function, but now we will be able to have more control over the output. Type the following command.

    ```
    santos = get_map(location = "Santos", zoom = 12)
    ```

13. Here, we have created an object called 'santos' in R. One important convention to understand, is when R programmers refer to a function, they generally refer to it with two parentheses after the function name, for example `get_map()`. You can type the name of the object `santos` into the R console, but this isn't going to be very helpful, since it will just print the data within the object. You can look at it and see it it makes any sense to you. Once you are finished lets use the `ggmap()` command to print our map of Santos, Brazil.

14. Did you just type in ggmap and get an error? Caught you! ggmap() is a *function*, as you can see by the () after the word "ggmap". Many functions require more information before they'll work - in this case, you need the information we saved into "santos" earlier. Try this:

    ```
    ggmap(santos)
    ```

    We can begin to do more with the `ggmap` package like add labels for the X and Y axis as well as give our plot a title.

    ```
    ggmap(santos) + labs(x = 'Longitude', y = 'Latitude')
    + ggtitle('The Home of Pele')
    ```

15. As before, in order to find out more information about a particular function type `?ggmap` and the help page for that function will pop up.

16. ***Deliverable Number 2: Now create a map of a place you would like to visit one day, and copy it into your word document. Make something that is visually interesting to you!***

Final Output for Submission

That's it! This assignment will be turned in for a grade, and is *due by 11:59PM EST on Friday, August 28th - note, Blackboard will NOT accept submissions after this deadline!:*

Make certain the word document you turn in on blackboard has the following:

- Your name!
- A Map of your favourite School, University, or home town.
- A Map of a place you would like to visit one day
- A copy of the code you used to produce each of the above maps.
- Any stretch goals you choose to include (see the next page).

There are few wrong answers to most labs in this course, but creativity is heavily encouraged to reach the highest grades. Make your visuals look good, use your data to make compelling arguments, and don't worry about "getting it wrong"!

Stretch Goals

Most labs in this course will have "stretch goals" - challenges that can be turned in for extra credit on your lab (note your grade cannot exceed 100!). These goals are designed to be more difficult than the mandatory assignment, and may require more critical thinking (or, even googling!) to solve. Good luck, and never hesitate to come talk with a TA or professor!

17. Next, we will import a shape file and do some very basic spatial statistics on it. To start lets install and load the `rgdal` just as we installed and loaded `ggmap`.

```
install.packages("rgdal", dependencies=TRUE)
library(rgdal)
```

We will also want to use a new R command in order to set the working directory. The working directory is the location where R will look for files that are being referenced as part of a command. While we can always specify the full path for a file, its often just easier to set the working directory in order to prevent our programming from becoming to verbose. For example (note, you'll need to change this working directory path to where YOU have the lab downloaded. This part may be hard if you've never done it before, so don't hesitate to ask for help from a TA or professor!).

```
setwd("/Users/tyfrazier/workspace/work_life/WM/Teaching/
COLL_100/labs/lab1_basics")
```

18. Now lets read a shape file into R and create a new "spatial object". A shapefile is essentially a file that defines the lines of a map. We use the `readOGR()` function in order to accomplish this task. The `readOGR()` function takes two qualifiers the `den =` specifier and the `layer =` specifier. The `dsn =` is just the name of the folder where the shape file and its supporting files are located. The `layer =` qualifier is just the name of the shape file.

```
acr = readOGR(dsn = "data", layer = "accra")
```

This function will import the shape file named accra and create the R object named `acr`, which is simply a shortened naming convention for the capital city of Ghana.

19. Finally, we will use the `tmap` to print a few basic spatial maps that describe Accra. Again lets install and load the package.

```
install.packages("tmap", dependencies=TRUE)
library(tmap)
```

Now we will use the `qtm()` function to spatially describe a variable. One variable we can use is named `SLUM_INDEX`. So lets use it.

```
qtm(shp = acr, fill = "SLUM_INDEX",
fill.palette = "-Blues")
```

We can also plot two variables side by side and compare them. Another variable we can use gives us the number of households per enumeration area.

```
qtm(shp = acr, fill = c("SLUM_INDEX", "N_HH_POP"),
fill.palette = c("Blues"), ncol = 2)
```

20. ***Stretch Deliverable: Copy your variables in a side-by-side comparison, and show the code you used to produce them.***