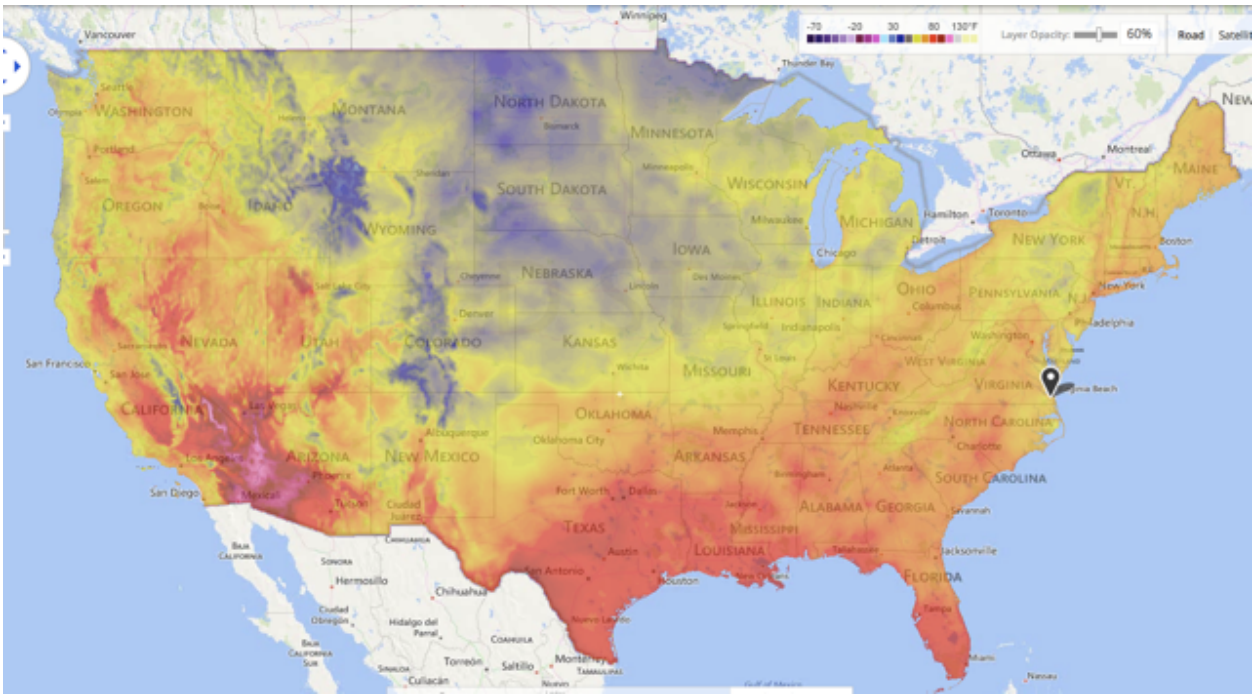
Introductory Lab. Some Basic Maps

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 ...

Goals of this Laboratory:

1. To learn how to install and load packages
2. To learn how to make a map using commands with an interpretive programming language

Session 1: Wednesday, August 26th

Step by Step Instructions: First we Acquire the Data

1. First lets open R
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. First we need to install the package onto our local system, then we will need to load the package into our current session. 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.

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

   Adding the comment in the command `dependencies=TRUE` will require R to install all the other packages `ggmap` is dependent upon in order for it to function properly.

3. Once you have installed the package `ggmap`, next you will need to load it into your current active R session. To do this type the following command

   ```
   R> library(ggmap)
   ```

   and your package is now active and ready for use.

4. Before we start writing code, we need to open a script file, then name and save it. From the R console, under the File menu pull down, choose New Document. Go ahead and save your document and give it a name. We will be typing our R commands into our script file and then sending those commands to the R console. This enables us to save our commands for reuse over and over again, as well as to revise the code, in much the same way we might edit a paper. To send a command from your script to the R console hold the `command` key and press `return` at the same time. For starters we can type

   ```
   R>  1 + 1
   ```

in our script and send the command to the R console.

5. To start out lets make a "quick map" in R by assigning a place to a variable. In order to do this we will use the arrow operator in R which is a combination of a less-than sign and a hyphen <-. The arrow operator is used to assign output from a function to an object, or to write over an existing object, or parts of an existing object. To start lets create and object by giving it a name. Since we are going to create a map of our favourite University, I will use one of my favourite places.

   ```
   R>  gt <- 'Georgia Tech'
   ```

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

6. Now lets execute our first function in R. The package `ggmap` has a quick map function that takes a named object and attempts to "know" what you are interested in mapping, and then creates a map for you.

   ```
   R>  qmap(gt, zoom = 14)
   ```

   We can also play around with the qmap function by adding some other qualifying comments.

   ```
   R>  qmap(gt, zoom = 13, source = "osm")
   R>  qmap(gt, zoom = 15, source = "stamen", maptype =
   "watercolor", crop = FALSE)
   R>  qmap(gt, zoom = 16, source = "stamen", maptype =
   "toner", crop = FALSE)
   ```

7. 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.

8. ***Challenge: Now create a map of your favourite school or University***

9. 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.

   ```
   R>  santos <- get_map(location = "Santos", zoom = 12)
   ```

   Again we have created an object in R, although this time we have used the `get_map()` function. 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.

```
R>  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.

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

10. 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.

11. ***Challenge: Now create a map of your favourite place or a place you would like to visit one day.***

12. 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`.

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

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

13. Now lets read our shape file into R and create a new spatial object. 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.

    ```
    R> 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.

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

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

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.

```
R> 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.

```
R> qtm(shp = acr, fill = c("SLUM_INDEX", "N_HH_POP"), fill.palette = c
```

Challenge: Find a shape file with a spatial statistic that geographically describes a place, location or region and create a spatial map.

Final Output for Submission

Due by noon on Friday, August 28th:

... Make certain the Final Report meets the following criteria.

- The Final Report for this Intro Lab should include your name
- A Map of your favorite School or University
- A Map of your favourite place or a place you would like to visit one day
- A Map created from a shapefile that describes a place, location or region statistically

Grading

Generate 3 maps