Instructor: McHale, Michael, Research Hydrologist, USGS

Code transcribed by: Charles Stoll

R-class Date: 02.23.2018

Course: Intro to ggmap package

Introduction to the ggmap package. This R package is related to ggplot in its functionality; both were written by Hadley Wicham. Because of this, you have the full power of ggplot and it uses the same arguments as ggplot. Other than the ggmap vignette, there currently isn't a lot of documentation for ggmap. Data and supplementary materials supplied by course instructor (M. McHale, Research Hydrologist, USGS). Reference R script for specific code and additional commentary.

This lesson introduces the user to the ggmap() package for R. The lesson gives a general overview of basic functionality, including methods for acquiring a map, creating a map, map layering, and saving map documents as JPG or PDF. This summary serves as a walk through for the introductory lesson R script that was developed during this workshop. It is also encouraged that you play around with the functions used in this lesson to learn the additional functionality of the package; for example, additional exercises could be done using other maps types, or symbolizing schema, than what are used in this demonstration.

Lastly, the script is written into sections so that if you are using Rstudio you can collapse the sections of code that you are not working in. This allows you to more readily access, organize, and work with individual sections of code.

**Script Work Flow**

Section:GGMAP Notes

This section of the script covers some initial notes about the R package. Many of the details presented in the script are also at the top of this summary description. Of special note is the description given in the GGMAP Notes section of the fundamental ideas behind how ggmap package works; it is recommended that you take a quick second to review these statements before proceeding through the script; this will give you a better understanding of ggmap principles.

Section: Setting working directory, Open libraries, and Import data

These three sections are commonplace to all scripts. They establish the parent directory that the script is operating in, open the libraries necessary to run the functions being used, and read the data that is going to be used into a data frame (df). As an aside, there are alternative methods to using the **setwd()** function; for example, if you are taking data from a folder not located in the same parent directory as where you are outputting data, you can alternatively create an input\_data\_location and output\_data\_location variable that you then use in the **paste()** function with the name of input/output files. With respect to the libraries being used in this lesson script, you may need to download the ggmap package prior to beginning running this script; this can be accessed quickly using the package install function in Rstudio. Lastly, it should be noted that there are also alternatives to how you choose to import data and read it into a data frame and that depending on the needs of a project, you may want to research these.

Section: Different ways to Enter a location for a map

This section covers the three basic ways that a user can enter spatial data to create a map; it was stated during the lesson that is possible to import polygon shapefiles that can be used for this task, however, this was not covered in the lesson. The three basic ways of determining the area your map focus on are as follows:

1. Users can enter an address; the example used in this demonstration was “Albany, NY.” The level of specificity that this entry method can use was not discussed in detail; street addresses, etc., will need to be tested with the different map types to determine if and where they would work.
2. A Longitude/Latitude value. Users can enter a single point location value to determine the central location of a map.
3. A bounding box. Users can enter the longitude/latitude combinations to “box” out the area you would like to map.

Section: Create and display a map using stamen map source type

This section introduces the method of using a variable associated with user location(s) to generate a map plot. This section introduces users to the four map source types available for use with this package, and then focuses on creating a map plot using the “stamen” map source type. Some of the input options for the **get\_map()** function can be a little confusing at first, but they will become more clear with continued use, especially if you are accustomed to using **ggplot()**. One example of how they can be confusing is that in the **get\_map()** function the “source =” and “**maptype =**” input parameters; “**source =**” is where users select the map source type (e.g. stamen or google), and “**maptype =**” is where users select the “type” of map they would like to see made. This section uses the stamen map source type and makes a “toner” type of map; there are many different “**maptypes =”** to choose from, it is recommended you review the package help documentation, as well as, the supplementary material for this lesson and experiment with what these different “**maptype =”** are.

Section: Create and display a map using a Google map sour type; use a more specific **get\_map()** function (i.e. **get\_googlemap()**)

This section repeats some of what the previous section does with the exception of modifying the **get\_map()** function to **get\_googlemap()**. There are alternatives to the more generalized **get\_map()** function in ggmap; these alternatives are specifically designed to individually access each map source type’s server whereas the **get\_map()** function is more of a “catch-all” general version. This section also introduces the “**zoom =**” input parameter into the **get\_map()** (**get\_googlemap()**) function. This allows the user to define the tile resolution of the map document that is pulled from the map source type server. It is advised to use this setting carefully; the higher the zoom number, the longer it will take to retrieve the tiles necessary to make the map. If the end goal of increasing the level of “**zoom =**” is to increase the scale, it is advised to do this by adjusting you input location parameter (e.g. decrease bounding box size) rather than “**zoom =”**; it may be required, depending on the map resolution needs of the project, to adjust both input location size and “**zoom =**” level to achieve desired results.

Section: Adjust map properties

This section focuses in on adding the map to a variable, then layering data over the map by “adding (+)” to the map variable. Specifically, this section creates point features from the supplementary data table and adds those to the map. It then proceeds to demonstrate a couple of the ways that a user can adjust these symbology of those features by adjusting the input parameters associated to them. It changes the size of points, and colors them according to the coordinate system information stored in the data table. This section also covers how to change the labels associated to the x and y axis of the map; these adjustments function and plot like in ggplot. Similarly, this section also demonstrates how to add a title to map document. Additional elements critical to map integrity, such as North Arrow, scale bar, etc., were not covered here, but should be researched before using these map plots in any publications. Users are also encouraged to research the input parameters to these layering functions as there are many functionality aspects not covered by this lesson; for example, the “**alpha =**” input parameter controls feature transparency, which could be of interest if working with polygon features.

Section: Save map as multiple outputs

This last section covers how to create a file name and extension for you map and then save it to your output folder. The two examples used here were saving the map as a JPG and PDF. Of note, if a user is generating multiple maps simultaneously, such as through a looping function, it is possible when saving as a pdf to place multiple maps into a pdf as separate pages (likely similar to ggplot). Conversely, it is not possible to achieve this with JPG; each map will become a distinct JPG.