CME 195/Stats 195 HW#3
Due April 26, 2016
Instructions:
1. Upload R script and any other files to Coursework dropbox using the filename <your SUNetid>stats195hw3.R. E.g., SUNetid "hgm7" gives filename "hgm7stats195hw3.R"
2. Follow any naming indications below.

Problem 1

We will write a function `my.geocoder` that, when given an address, will print its approximate latitude and longitude. We will use Google's geocoding API to perform this lookup. The geocoding API is a service that allows you to request a webpage containing among other things the latitude and longitude information. The webpage's url/address will contain the address you need to look up. For example, if I want information on Stanford, CA, I point my browser to:

`http://maps.google.com/maps/api/geocode/xml?address=Stanford, CA`

You should do this programmatically. Specifically, your function, when given an address string (e.g., `Stanford, CA`), should first paste it to the base url,

`http://maps.google.com/maps/api/geocode/xml?address=`

Next, it should read in the webpage at that address. Next, it should find the lines in the webpage containing latitude and longitude information, extract those coordinates from those lines, and print them out.

Here are two issues you will encounter. First, the address string (e.g., `Stanford, CA`) needs to be formatted as a valid url. For our purposes, this means you must replace all spaces with the string `%20` (e.g., `Stanford,%20CA`). (If you put the above full url into your browser, you may have noticed your browser making that substitution for you.) Second, when you read in the webpage, there may be multiple lines containing latitude and longitude data. You only need to print out one of these pairs.

```
> my.geocoder('Stanford, CA')
[1] "  Stanford, CA, USA"
[1] "    37.4241060"
[1] "    -122.1660756"
> my.geocoder('97 fuller st., boston, ma')
[1] "  97 Fuller Street, Boston, MA 02124, USA"
[1] "    42.2824030"
[1] "    -71.0700653"
```

Problem 2

We will plot aggregate tax collection information on a map of the US. Usually plotting on a map requires information containing the shapes and arrangement of the locales, but we will use a quicker and more accessible approach.

2(a)

Read in the IRS's tax data at

`http://www.irs.gov/file_source/pub/irs-soi/12cyallnoagi.csv`

The variables we will focus on are `STATE`, `COUNTYNAME`, and `A00100`. The last variable gives the aggregate adjusted gross income for the corresponding county, state pair. Since aggregate AGI varies so much we will work with logarithms: create a new variable `lagi` containing the logarithms of the values in `A00100`.

2(b)

Next, we would like a way to map a county, state pair to its latitude and longitude. You could try to use your function from Problem 1 but Google will probably rate limit you well before you get data for all the counties in the IRS data. Instead, use the CSV at

`http://notebook.gaslampmedia.com/wp-content/uploads/2013/08/zip_codes_states.csv,`

which contains the longitude and latitude for the many city centers in the US. This data set typically contains the latitude and longitude of multiple cities for each county, state pair. You should use the mean of all these latitudes and longitudes to represent the latitude and longitude for the county, state pair that they lie in. If you get stuck on this part, you can download a CSV with the required data from Coursework in order to complete the rest of the problem.

2(c)

Create a data frame `dat` containing the `STATE`, `COUNTYNAME`, and `lagi` columns from the IRS data and the corresponding `latitude` and `longitude` data from the geographical data set. This operation is known as a "join": You look for rows in the IRS data and the geographical data set that have the same state, county values, and then you combine the other variables on those matching rows from the two data frames. In this way you end up with a data frame containing both the log AGI for a given county as well as its latitude and longitude information. You may find that some county, state pairs exist in one data set but not the other. Or you may notice some nonsensical values in one of the data sets. These are common situations so try to think of a practicable, consistent approach. If you get stuck on this part, you can download a CSV with the required data from Coursework in order to complete the rest of the problem.

2(d)

Using your data set from 2(c), plot latitude versus longitude and set the colors to correspond to the log AGI. Longitude goes on the horizontal axis (see figures). Save the image and upload it with your script to Coursework under the name <your SUNetid>stats195hw3prob2d.jpg (the extension can vary if you are using a different graphics format).

2(e)

Extra Credit. Counties vary widely in size and population. Create a similar map but normalize `lagi` by dividing it by the county population.

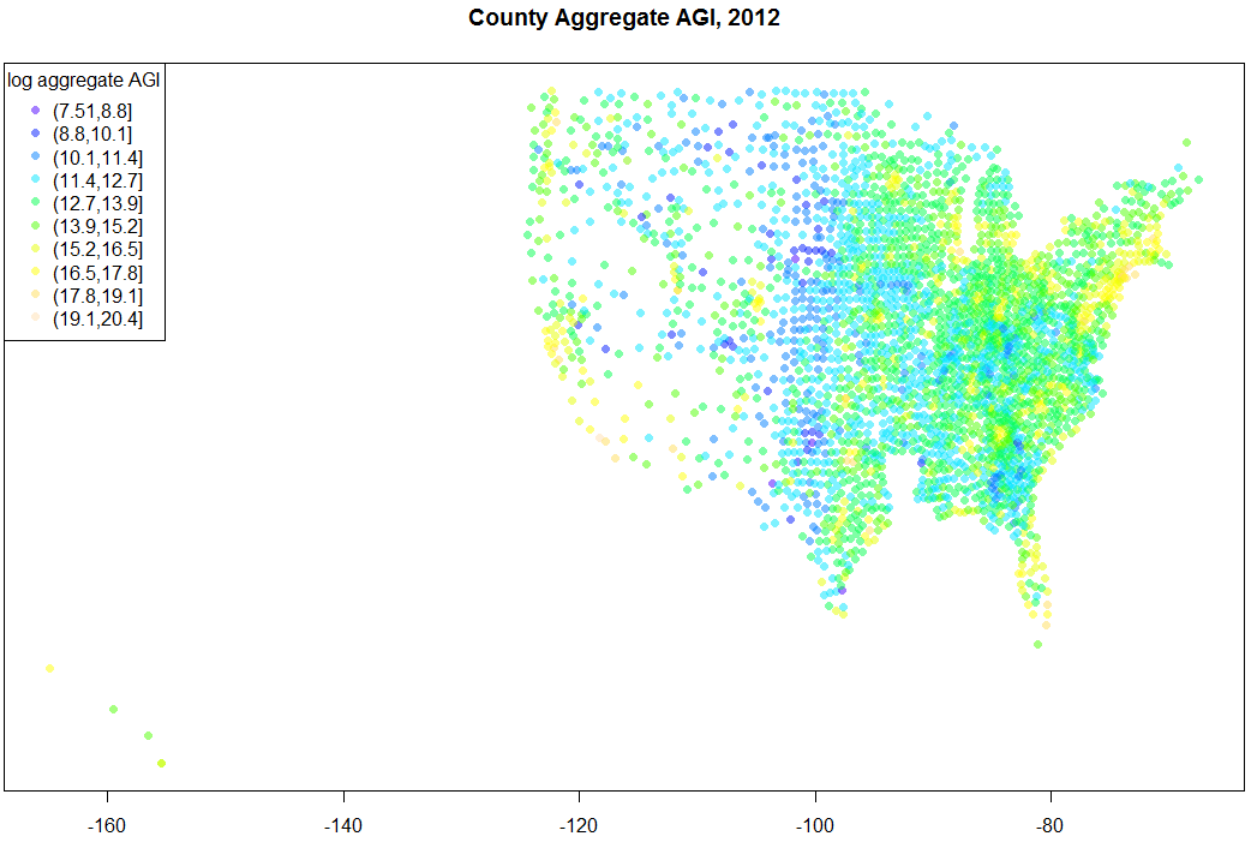

Figure 1: Problem 2(d) using `base` graphics

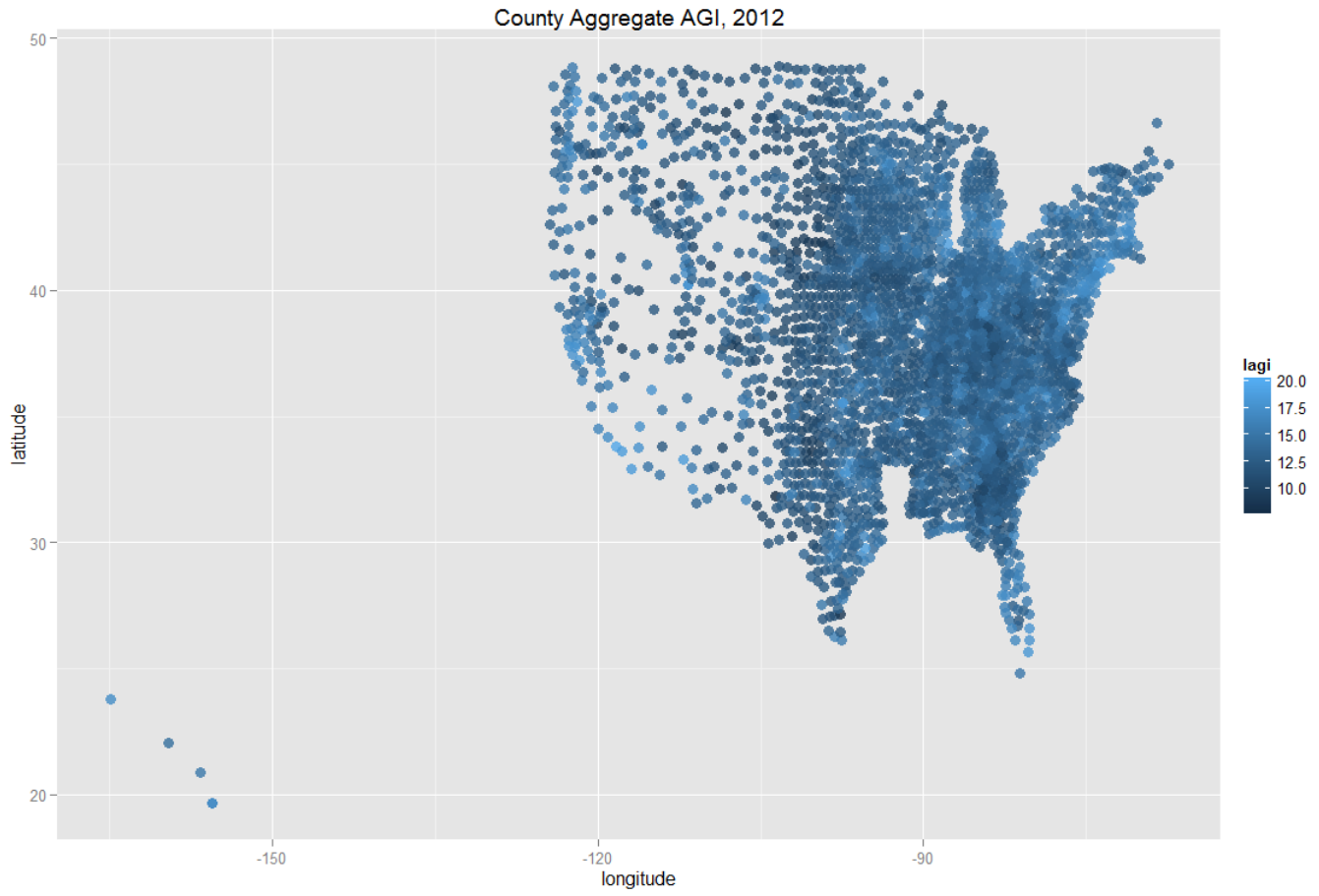

Figure 2: Problem 2(d) using `ggplot2`