DCF Week 4 Activity I

Data and Computing Fundamentals

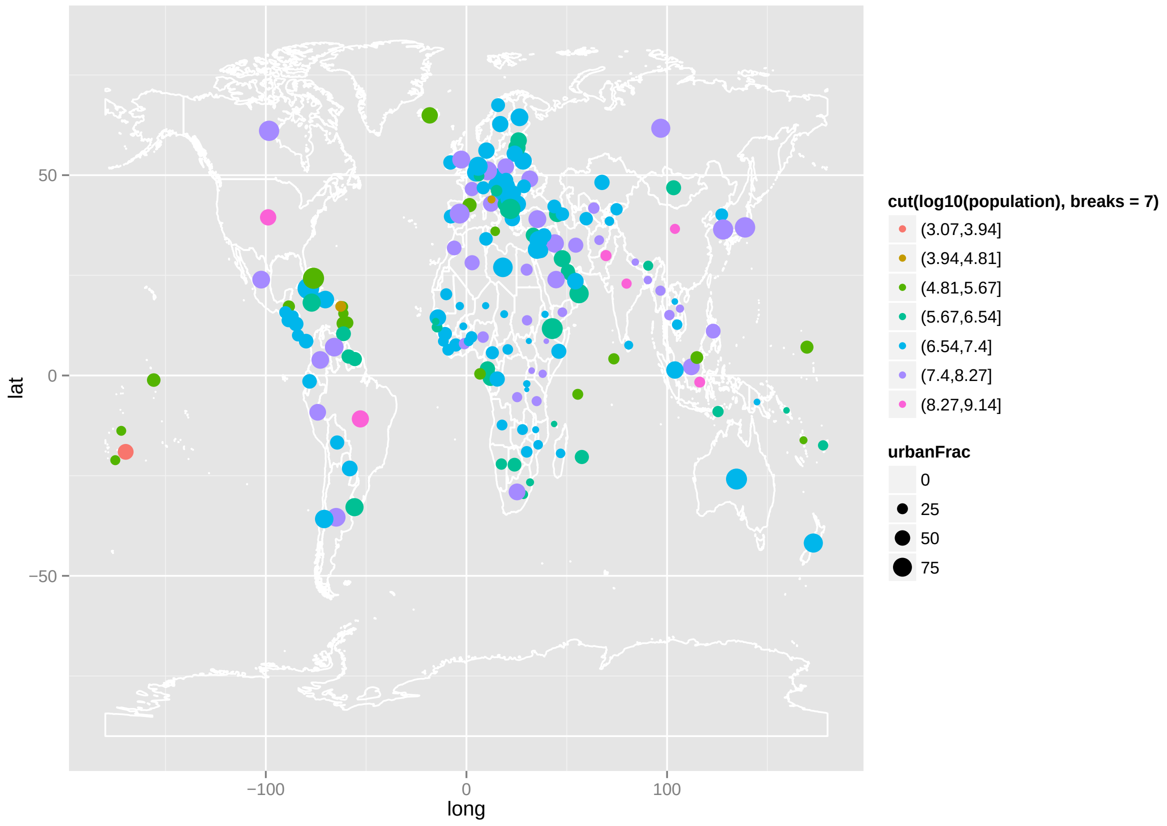
Do this work with one or more partners. One person should compose the Rmd file, with the others helping to construct the commands, debug, etc. Put the names of the people in the group and their email addresses in the Rmd file, like this.

- Ben Franklin, bfranklin76@gmail.com
- Tom Jefferson, monticello@yahoo.com
- Abby Adams, aa@baycolony.org

To get the email to show up as a link, enclose it in angle bracket, e.g., `<bfranklin76@gmail.com>`.

Joining Multiple Files

The `WorldCities` data table gives the population of each city. The `CountryData` data table gives, among other things, the population of each country. `CountryCentroids` gives the latitude and longitude of the center of each country. In this activity, you will be joining these sources of information together in order to create a glyph-ready data table showing the total population of each country as the size of a dot and the fraction of the country's population in cities as a dot color. (Note: Don't worry about the country borders for now. You'll see how in the final step.)



One: Envision the Output

What will the glyph-ready data look like to produce the plot?

Start by analyzing the graphic.

- What variables form the frame?
- What is the glyph? (Ignore the country borders, which are really a guide, not a glyph.)
- What graphical properties does the glyph have?

Which variables are mapped to those properties? Sketch out, on paper, the table by writing the variable names and a case or two of made-up data in the usual rectangular format.

Take a picture of your sketch and send it to yourself so that you can include your sketch of the table in your Rmd document. (If you don't remember how to include a graphic in an Rmd file, ask!)

Two: Look at the Starting Data

Look at each of `CountryCentroids`, `WorldCities`, and `CountryData` to find what "raw material" you have to generate the glyph-ready data. Note that the identifier for the country is called `name` in one of the data tables and `country` in the other two.

Decide whether the case in each data table is already in the form it will need to be in for the glyph-ready data.

- `CountryCentroids` yes or no
- `CountryData` yes or no
- `WorldCities` yes or no

Select (that is, `select()`) the variables from each data set that you're going to need to create the glyph-ready data. (This holds particularly for `CountryData` which has dozens of variables that you don't need in the glyph-ready data.)

As you do the selecting, rename variables as appropriate so that you can join the tables later. Remember, joins match up cases in the two tables being joined. The `inner_join()` function does this by looking for matching variable names whose values can be matched.

To rename a variable using `select()`, use a command like this:

```
NewTable <-
  select( InputTable, newname=oldname,
          ... other vars you want to include )
```

Three: Matching Countries

Note that `WorldCities` represents the country in a different encoding than `CountryCentroids` and `CountryData`. You'll see the acronym ISO in some of the variable names. This stands for the International Standards Organization, which publishes such things as standard country abbreviations.

You'll have to translate the `WorldCities` country into a form compatible with `CountryData` and `CountryCentroids`, or *vice versa*. To do this, you may find the `Codes` and `Synonyms` data tables useful. Load them with these commands.

```
Codes <-
  fetchData("DCF/CountryCodes.csv") %>%
  select(ISO2, ISO3)
Synonyms <-
  fetchData("DCF/CountrySynonyms.csv") %>%
  mutate( country=as.character(country))
```

Write down on paper a plan for joining the information in the various tables in order to carry out the translation. This should show, at each stage, which data tables you need to join and what variable(s) they should be joined on. Give a sketch of what you plan the output from the join to look like. Remember, `inner_join()` requires the matching variables to have the same name, so you may need to rename some variables. Include this on your plan.

Four: Urban Population

Carry out the appropriate operation to transfigure `WorldCities` into a form whose cases/variables you want to join with the other data.

Five: Getting Glyph-Ready

One of the variables you will be plotting is "urban fraction of the total population," a number between 0 and 100%. Perform the appropriate transfiguration to create this variable.

Final Step: Make the graph

This is a matter of *mapping* variables in your glyph-ready data into graphical aesthetics.

If you want to add country borders, adding this layer to `ggplot()` will do it:

```
geom_polygon( data=mosaic:::World_Countries_df,
                color="white", fill=NA,
                aes( x=long, y=lat, group=group ))
```