Leaflet In Class Exercises

Swarnima Sircar, Gabriel Petrov, Chandler Beyer
Aiman Imtiaz, Terence Choo, Haaken Bungum

13/04/2019

Task 0

Set the working directory to your folder containing the "CAIT Country CO2 Emissions" csv file.

Task 1

Load the emissions data here. You may use the code from exercise 17 of this class. Use `countrycode()` to add a column with ISO Alpha-3 country codes to the data frame. Then, using `joinCountryData2Map()`, turn the data frame into a spatial data frame. We'll be looking at 2014 again this time.

Task 2

Let us draw our basic world map. Use the `leaflet()` function. Edit `setView()` to position the map so that when it loads, we only see Singapore. After doing so, zoom out and inspect the map. Try adding the polygons created by your spdf.

Task 3

Let us now add some colour to the map. - We want the break points we used in our previous maps. As the notes noted, there are more than 9 bins RColorBrewer's Palette. Write a `pal` so that the colour gradient of our palette separates at the points specified by `breaks` in Prof's code. Refer to the notes to remind yourself how `pal` works. - Set the fill opacity to something that looks good - Make the nation borders thicker with an appropriate degree, opaqueness and color. White worked well for us. Experiment with different types of dashed patterns.

Task 4

Our map now looks something like what we want. But we want to make the polygons to highlight as we pass over them. Hint: Check the readings. Play around with it. We want to highlight it such that when our mouse hovers over a particular country, the borders are - some color. grey looked good, but feel free to experiment with different hexadecimals - thicker (choose a value that is pleasing to the eye) - We want the highlight itself (i.e. not the borders) to be somewhat opaque

Task 5

To populate this map and actually make it useful, let us create some data labels. We would like every time we go over a country to get a label which has - on the first line - that country's name and on the second it ought to say "CO_2 levels: X", where X is that specific country's CO_2 output for 2014. Create a variable `labels` of class "list" that contains all of these variables. Pass it through html, so that Leaflet knows to treat each label as HTML instead of as plain text.

Task 6

Add the labels to your map. Optional: play around with the `labelOptions()`, set the text size to something appropriate, set a preferred direction, and more. Here's a bit of HTML also that you can add to the variable "style":

```
style = list("font-weight" = "normal", padding = "3px 8px")
```

We encourage you to play around with it.

Task 7

Let us now add a legend to our map. Use the `addLegend()` function. Set its opacity to something nice, give it a meaningful title and position it in an appropriate position on your map.

Task 8

Give the map a title using the following recipe. Use the `addControl` function to add it to the top right of the map. You may need to load a couple of libraries. Make the title say "CO_2 Emissions around the Globe in 2014". Hint: Google how to add a title in `leaflet()`. Look for `tags$div`.

Bonus:

Hmmmmm... Something's not quite right. You see that disgusting grey area on top? And notice where our title is. It's outside the map! It's like one of those movies that don't quite fit the screen. Terrible. How would we change that? Consult our group's representative for a hint, but we're curious as to what you'll come up with on your own. If we're low on time, he'll help you out once you get here. But try and have an idea first. The way you should be checking whether it works is by kniting, not in your .Rmd file.

N.B. You may have to copy some of your previous code, but seeing as we were going to make you do that anyway below, why not just go ahead and do so now.

Task 9

And that's all! Combine and add comments to all your code here. Should look something like Figure 1 below.

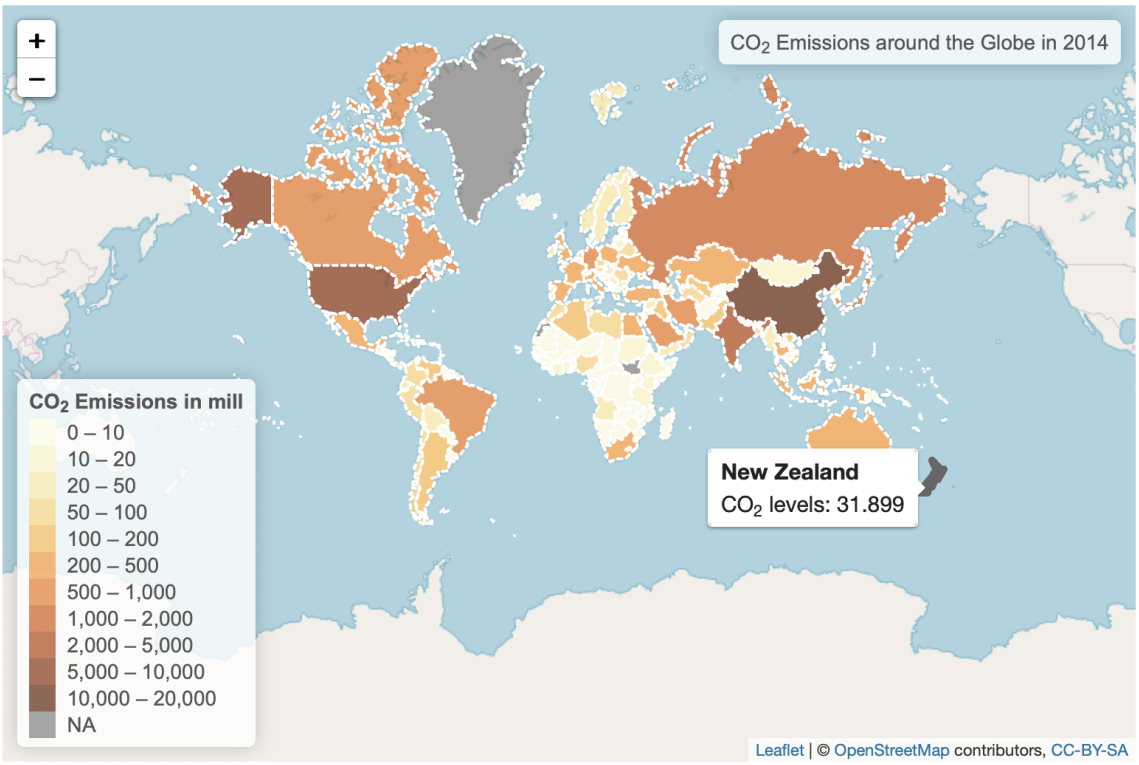

Figure 1: Result