# Leaflet In Class Exercises

*Swarnima Sircar*

*13/04/2019*

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

```
# Task 0
```

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.

```
# Task 1
```

Set the colour palette to the one we've been using in class.

```
# Task 2
```

Let us draw our basic world map. Use the `leaflet()` function The basic recipe is already below. Edit `setView()` to position the map so that when it loads, we only see Singapore.

```
# Task 3
m <- leaflet(spdf) %>%
  setView() %>%
  addTiles() %>%
  addPolygons()
m
```

Now zoom out and inspect the map. As you can see, it has the borders of all the countries, and you can move around.

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 R's ColorBrewer Palette. Write a `pal` so that the colour gradient of our palette separates at the following points: `(0, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000)`. Refer to the notes to remind yourself how `pal` works. - Set the fill opacity to 0.7 - Make the blue nation borders thicker to an appropriate degree, opaque, white, and dashed.

```
# Task 4

# Cut points for turning numerical values into categories on the map.
breaks <- c(0, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000, 20000)
pal <-
```

Our map now looks something like what we want. But we want to make the polygons highlight as we pass over them. Hint: the `addPolygons()` function has a `highlight` function. Play around with it. We want to highlight it such that when our mouse hovers over a particular country, the lines are - grey (#666) - thicker (choose a value that is pleasing to the eye) - opaque - the lines come to the front (Note that each `addPolygons()` function writes over itself, so you have to keep reproducing your code from above.)

```
# Task 5
```

To populate this map and actually make it useful, to us create some data labels. `sprintf` is a function that returns a character vector containing a formatted combination of text and variable values. - `%s` and `%g%` simply reference the data values we want to put into the labels. - `<br/>` inserts a line break - `<sub>2</sub>` makes the "2" print as subscript. - Optional: Figure out how to make the country names print in bold. We pass it through `lapply(htmltools::HTML)` so that Leaflet knows to treat each label as HTML instead of as plain text.

```
# Task 6
labels <- sprintf(
  "<strong>%s</strong><br/>CO<sub>2</sub> levels: %g",
  spdf$Country, spdf$CO2) %>%
  lapply(htmltools::HTML)
```

Add the labels to your map. (Note that each `addPolygons()` function writes over itself, so you have to keep reproducing your code from above.) Optional: play around with the `labelOptions()`, set the text size to 15px, set a preferred direction, and more.

```
# Task 7
```

Let us now add a legend to our map. Use the `addLegend()` function. Set its opacity to 0.7, give it a meaningful title and position it in the bottom left of 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.

```
# Task 9
rr <- tags$div(HTML("CO<sub>2</sub> Emissions around the Globe in 2014"))
m <- m %>% addControl()
m
```

And that's all! Combine and add comments to all your code here.

```
# Task 10
```