Web scraping in R

Emily Burchfield

2017-07-11

The data

In the olden days, you'd come across a page with say, air quality data for counties in the United States from 1980 to 2017. Each year of data is stored in a separate link to a zip file. You'd click each link, download each Excel file, and copy paste everything into a single Excel file. This process might take you a few hours. Ahhhh, but not any more! Using `rvest` and a few other tools, the while process is very quick and easy. Let's visit aforementioned dataset... you can find it AT THIS LINK.

Scraping link names from a website

You can use the piping function[1] to access more precise aspects of the website of interest. In our case, we start with the website (`page`) and zoom in to the zipped files of interest (AQI by county). We'll also use the `stringr` package which helps us work with regular expression.[2] Lucky for us, we don't have to travel far into the labyrinth of regex:

[1] *Ceci n'est pas une pipe*, or rather, it is a link to a website explaining pipes `%>%`. And for the art fans in the room, CHECK THIS OUT.

[2] Thanks to this awesome post on StackExchange for the recs on how to most efficiently script this

```
library(rvest)
library(stringr)

page <- read_html("https://aqsdr1.epa.gov/aqsweb/aqstmp/airdata/download_files.html")

fn <- page %>% html_nodes("a") %>% html_attr("href") %>%
    str_subset("annual_aqi_by_county")
```

So what's going on here... we create a `page` object that holds the link to the website of interest. `html_nodes` then uses CSS Selectors (these help you quickly identify all items on a web page meeting some criteria, in our case, `a` indicates an active link on the page). The `html_attr` function then pulls the URLs from these active links. Then we use the `str_subset` function from the `stringr` package to find the links of interest, i.e. annual AQI measurements for each county. The output of this pipe is a list of the links of interest. Now to download this list of zip files...

Downloading ZIP files

```
path <- "https://aqsdr1.epa.gov/aqsweb/aqstmp/airdata/"

datalist <- list()

for (i in 1:length(fn)) {
    temp <- tempfile()
    download.file(paste0(path, fn[i]), temp)
    data <- read.csv(unz(temp, gsub(".zip", ".csv",
        fn[i])), header = T)
    datalist[[i]] <- data
    unlink(temp)
}

county_aqi <- do.call(rbind, datalist)
```

Here we loop through the zip file names, download them, and append them to our final `data.frame` that's called `county_aqi`. Let's go through this step by step. First, we provide the `path` to the zipped files. We create a `temp` directory to store the zip file we download. We don't actually want to keep the zip file, just import the data into `R` to create our final `data.frame`. We then use the `download.file` function to pull the zip file off of its website using by pasting the pathname and the zip file name together using `paste0`.[3] We add that `header=T` to tell the `read.csv` function to read the first line of the `.csv` file we're unzipping as a header (i.e. lots of variable names).[4] We then append the `data` variable to the `datalist` using the index of the for loop `i` and, importantly, `unlink` (i.e. delete) the temporary file we've created to store our zip. Finally, we bind the rows of all datasets in our `datalist` into a single, lovely `data.frame` called `county_aqi`.

[3] We could have used `paste` too, but we'd have to specify `sep = ""`. `paste0` is a sort of shortcut.

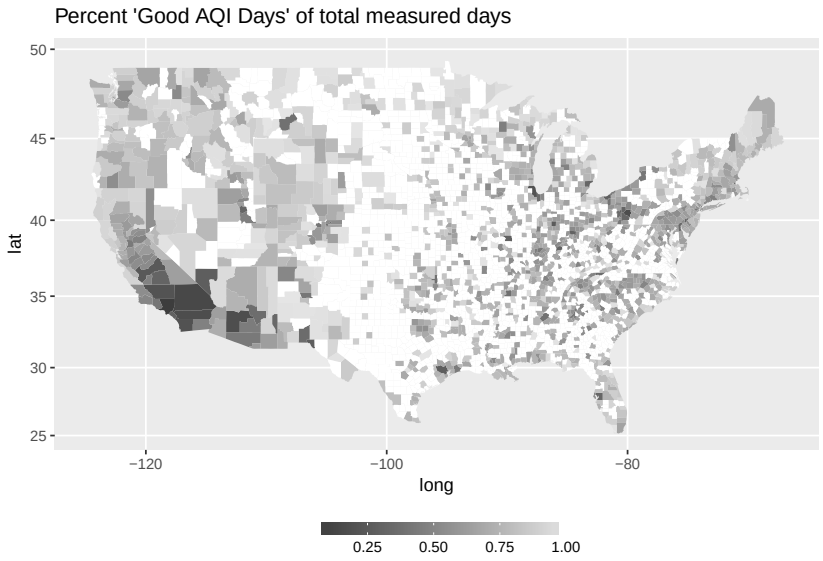
[4] It's helpful to download one file before beginning to inspect for specificities you may have to address, i.e. file type, missing header, missing data, inconsistencies across files.

Looking at our data

I hope I've convinced you that this approach is much faster and easier than manually downloading zip files. Let's generate a few plots to see what this dataset has to say.

```
library(tidyverse)

cty_AQI <- county_aqi %>% group_by(State, County) %>%
    summarize(mean_AQI = mean(Good.Days/Days.with.AQI)) %>%
    select(County, State, mean_AQI)

us <- map_data("county")
```

```
us$State <- str_to_title(us$region)
us$County <- str_to_title(us$subregion)

us_AQI <- merge(cty_AQI, us, by = c("State", "County"),
    all.x = T, all.y = T)
us_AQI <- us_AQI[order(us_AQI$order), ]

us_map <- ggplot(us_AQI, aes(x = long, y = lat,
    group = group)) + geom_polygon(aes(fill = mean_AQI)) +
    coord_map() + theme(legend.position = "bottom") +
    scale_fill_continuous(low = "gray24", high = "gray90",
        na.value = "white") + guides(fill = guide_colorbar(barwidth = 10,
    barheight = 0.5)) + labs(fill = "") + ggtitle("Percent 'Good AQI Days' of total measured days")

us_map
```



I used a number of functions in `dplyr` to group data by county and compute the average number of good days out ot the total number of measured days from 1980 to 2017.[5] I use the `stringr` `str_to_title` function to capitalize "State" and "County" in the `map_data` county spatial object I pull from `ggplot`. This allows me to merge this spatial dataset with the `cty_AQI` values I calculated. And then the

[5] Note that this visualization masks the variability of good days through time, an important piece of info to consider!

mapping... I like sticking with `ggplot` as much as is possible. Try adding `geom_path(color="white")`. This adds county boundaries to the plot. I like moving the legend to the bottom of the plot using `theme(legend.position = "bottom")` and to set my own color scale.[6] In this image, the regions shaded in a darker gray are regions with lower percentage of "good days" out of the total days measured, i.e. generally a lower air quality through time.[7]

Finally, just out of interest, let's see how the darker regions correspond with large cities in the US (population greater than a million people)

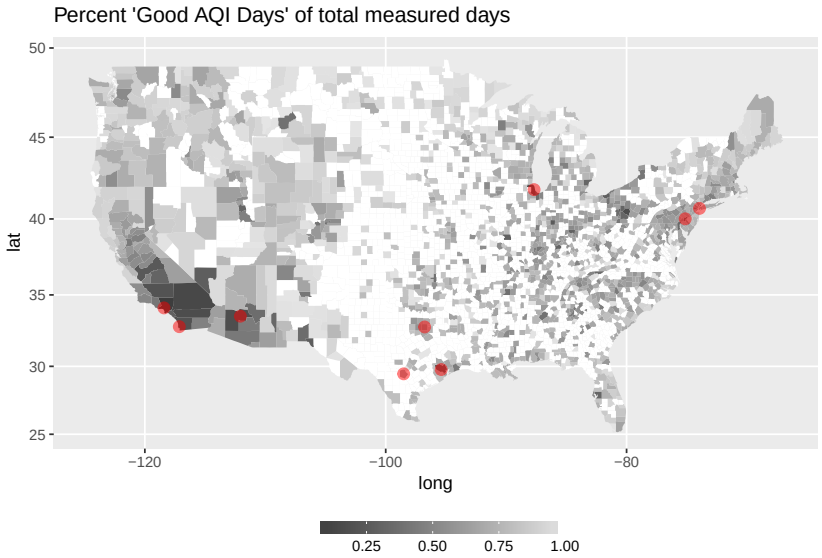
[6] If you want to go crazy with color, check out the list of color options HERE.

[7] Again, this is likely very biased based on the total number of days measured. This map is only a first visualization.

```
library(maps)
data(us.cities)

top_pop <- us.cities %>% filter(pop > 1e+06)
print(top_pop[, 1:3])
```

```
##              name country.etc     pop
## 1      Chicago IL          IL 2830144
## 2       Dallas TX          TX 1216543
## 3      Houston TX          TX 2043005
## 4  Los Angeles CA          CA 3911500
## 5     New York NY          NY 8124427
## 6 Philadelphia PA          PA 1439814
## 7      Phoenix AZ          AZ 1450884
## 8  San Antonio TX          TX 1278171
## 9    San Diego CA          CA 1299352
```

```
city_us_map <- us_map + geom_point(data = top_pop,
    inherit.aes = F, aes(long, lat), size = 3,
    color = "red", alpha = 0.5)

city_us_map
```



Hmmm… looks interesting, and certainly worth futher exploration! Voila - from zipped files hosted on a website to a map in no time!