Lab 4 - Part 1

Yaqi Shi, 1003813180

2023-02-04

The lab exercise results are in the end of this file and it contains only Q1 to Q3

Extracting data on opioid prescriptions from CDC

We're going to grab some data on opioid prescription rates from the CDC website. While the data are nicely presented and mapped, there's no nice way of downloading the data for each year as a csv or similar form. So let's use `rvest` to extract the data. We'll also load in `janitor` to clean up column names etc later on.

Getting the data for 2008

Have a look at the website at the url below. It shows a map of state prescription rates in 2008. Let's read in the html of this page.

```
cdcpage <- "https://www.cdc.gov/drugoverdose/rxrate-maps/state2008.html"
cdc <- read_html(cdcpage)
cdc
```

```
## {html_document}
## <html lang="en-us" class="cdc-2022 theme-purple cdc-page-type-content">
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
## [2] <body class="no-js cdc-page">\r\n\t<div id="skipmenu">\r\n\t\t<a class="s ...
```

Note that it has two main parts, a head and body. For the majority of use cases, you will probably be interested in the body. You can select a node using `html_node()` and then see its child nodes using `html_children()`.

```
body_nodes <- cdc %>%
 html_node("body") %>%
 html_children()
head(body_nodes)
```

```
## {xml_nodeset (6)}
## [1] <div id="skipmenu">\r\n\t\t<a class="skippy sr-only-focusable" href="#con ...
## [2] <div class="header-language-bar container text-right pt-1 pb-1 fs0875">\r ...
## [3] <header id="page_banner" role="banner" aria-label="Banner"><div class="co ...
## [4] <div class="container d-flex flex-wrap body-wrapper bg-white">\r\n\t\t\t< ...
## [5] <footer class="" role="contentinfo" aria-label="Footer"><div class="conta ...
## [6] <script src="https://www.cdc.gov/config/cdc_config.js"></script>
```

Inspecting elements of a website

The above is still fairly impenetrable. But we can get hints from the website itself. Using Chrome (or Firefox) you can highlight a part of the website of interest (say, 'Alabama'), right click and choose 'Inspect'. That gives you info on the underlying html of the webpage on the right hand side. Alternatively, and probably easier to find what we want, right click on the webpage and choose View Page Source. This opens a new window with all the html. Do a search for the world 'Alabama'. Now we can see the code for the table. We can see that the data we want are all within `tr`. So let's extract those nodes:

```
head(cdc %>% html_nodes("tr"))
```

```
## {xml_nodeset (6)}
## [1] <tr>\n<th>State</th>\n<th>State Abbreviation</th>\n<th>Opioid Dispensing  ...
## [2] <tr>\n<td>Alabama</td>\n<td>AL</td>\n<td>126.1</td>\n</tr>\n
## [3] <tr>\n<td>Alaska</td>\n<td>AK</td>\n<td>68.5</td>\n</tr>\n
## [4] <tr>\n<td>Arizona</td>\n<td>AZ</td>\n<td>80.9</td>\n</tr>\n
## [5] <tr>\n<td>Arkansas</td>\n<td>AR</td>\n<td>112.1</td>\n</tr>\n
## [6] <tr>\n<td>California</td>\n<td>CA</td>\n<td>55.1</td>\n</tr>\n
```

Great, now we're getting somewhere. We only want the text, not the html rubbish, so let's extract that:

```
table_text <- cdc %>%
  html_nodes("tr") %>%
  html_text()

head(table_text)
```

```
## [1] "State\nState Abbreviation\nOpioid Dispensing Rate per 100\n"
## [2] "Alabama\nAL\n126.1\n"
## [3] "Alaska\nAK\n68.5\n"
## [4] "Arizona\nAZ\n80.9\n"
## [5] "Arkansas\nAR\n112.1\n"
## [6] "California\nCA\n55.1\n"
```

This is almost useful! Turning it into a tibble and using `separate` to get the variables into separate columns gets us almost there:

```
rough_table <- table_text %>%
  as_tibble() %>%
  separate(value, into = c("state", "abbrev", "rate"), sep = "\n", extra = "drop")
head(rough_table)
```

```
## # A tibble: 6 x 3
##   state      abbrev             rate
##   <chr>      <chr>              <chr>
## 1 State      State Abbreviation Opioid Dispensing Rate per 100
## 2 Alabama    AL                 126.1
## 3 Alaska     AK                 68.5
## 4 Arizona    AZ                 80.9
## 5 Arkansas   AR                 112.1
## 6 California CA                 55.1
```

Now we can just divert to our standard tidyverse cleaning skills (`janitor` functions help here) to tidy it up:

```
d_prescriptions <- rough_table %>%
  janitor::row_to_names(1) %>%
  janitor::clean_names() %>%
  rename(prescribing_rate = opioid_dispensing_rate_per_100) %>%
  mutate(prescribing_rate = as.numeric(prescribing_rate))

head(d_prescriptions)
```

```
## # A tibble: 6 x 3
##   state      state_abbreviation prescribing_rate
##   <chr>      <chr>                         <dbl>
## 1 Alabama    AL                           126.
## 2 Alaska     AK                            68.5
## 3 Arizona    AZ                            80.9
## 4 Arkansas   AR                           112.
## 5 California CA                            55.1
## 6 Colorado   CO                            67.7
```

Now we have clean data for 2008!

Take-aways

This example showed you how to extract a particular table from a particular website. The take-away is to inspect the page html, find where what you want is hiding, and then use the tools in `rvest` (`html_nodes()` and `html_text()` particularly useful) to extract it.

Question 1

Add a year column to `d_prescriptions`.

Answer

Add a year column to the dataset and fill in with value "2008"

```
d_prescriptions$year <- rep(2008, nrow(d_prescriptions))
```

Getting all the other years

Now I want you to get data for 2008-2019 and save it into one big tibble. If you go to cdc.gov/drugoverdose/rxrate-maps/index.html, on the right hand side there's hyperlinks to all the years under "U.S. State Opioid Dispensing Rate Maps".

Click on 2009. Look at the url. Confirm that it's exactly the same format as the url for 2008, except the year has changed. This is useful, because we can just loop through in an automated way, changing the year as we go.

Question 2

Make a vector of the urls for each year, storing them as strings.

Answer

First, I checked that for 2009, the url is "https://www.cdc.gov/drugoverdose/rxrate-maps/state2009.html" which is in the same format as the url for 2008 except that the year changes. Now we create a vector of url for each year and store it as string.

```
# Create the url lists
year_list <- 2008:2019
url_list <- rep("https://www.cdc.gov/drugoverdose/rxrate-maps/state2008.html",12)

for(i in 1:12) {
  yr <- year_list[i]
  url_list[i] <- paste("https://www.cdc.gov/drugoverdose/rxrate-maps/state", yr, ".html",sep = "")
}

# Display the results
url_list <- tibble(url = url_list, year= year_list)
url_list
```

```
## # A tibble: 12 x 2
##    url                                                     year
##    <chr>                                                  <int>
##  1 https://www.cdc.gov/drugoverdose/rxrate-maps/state2008.html  2008
##  2 https://www.cdc.gov/drugoverdose/rxrate-maps/state2009.html  2009
##  3 https://www.cdc.gov/drugoverdose/rxrate-maps/state2010.html  2010
##  4 https://www.cdc.gov/drugoverdose/rxrate-maps/state2011.html  2011
##  5 https://www.cdc.gov/drugoverdose/rxrate-maps/state2012.html  2012
##  6 https://www.cdc.gov/drugoverdose/rxrate-maps/state2013.html  2013
##  7 https://www.cdc.gov/drugoverdose/rxrate-maps/state2014.html  2014
##  8 https://www.cdc.gov/drugoverdose/rxrate-maps/state2015.html  2015
##  9 https://www.cdc.gov/drugoverdose/rxrate-maps/state2016.html  2016
## 10 https://www.cdc.gov/drugoverdose/rxrate-maps/state2017.html  2017
## 11 https://www.cdc.gov/drugoverdose/rxrate-maps/state2018.html  2018
## 12 https://www.cdc.gov/drugoverdose/rxrate-maps/state2019.html  2019
```

Question 3

Extract the prescriptions data for the years 2008-2019, and store in the one tibble. Make sure you have a column for state, state abbreviation, prescription rate and year. Note if you are looping over years/urls (which is probably the easiest thing to do), it's good practice to include a `Sys.sleep(1)` at the end of your loop, so R waits for a second before trying again.

Plot prescriptions by state over time.

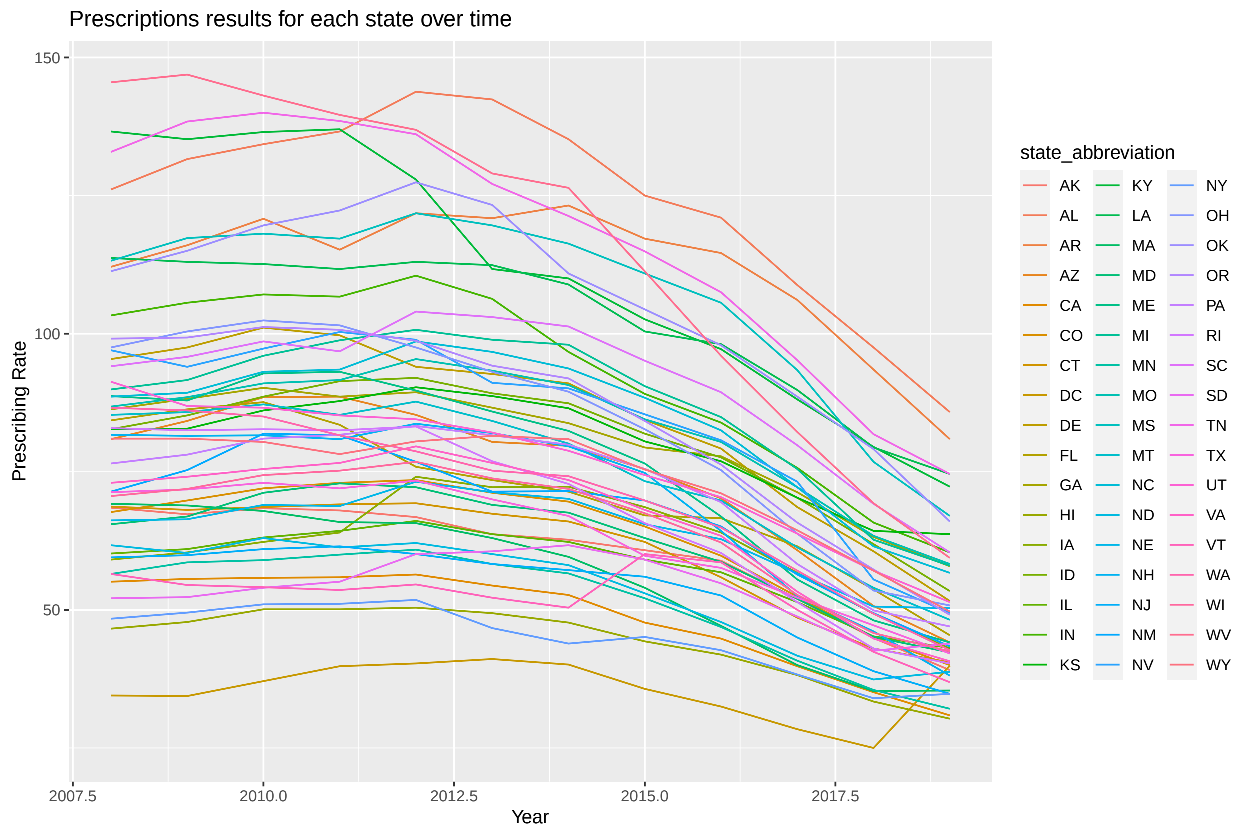
Answer

First, we extract the data from the website based on what we did for 2008 data:

```
# Set up the tibble
all_data <- c()

for(i in 1:12) {
  # Extract the information
  cdcpage <- url_list$url[i]
  cdc <- read_html(cdcpage)

  # Format the data
  table_text <- cdc %>%
  html_nodes("tr") %>%
  html_text()

  rough_table <- table_text %>%
  as_tibble() %>%
  separate(value, into = c("state", "abbrev", "rate"), sep = "\n", extra = "drop")
  d_prescriptions <- rough_table %>%
  janitor::row_to_names(1) %>%
  janitor::clean_names() %>%
  rename(prescribing_rate = opioid_dispensing_rate_per_100) %>%
  mutate(prescribing_rate = as.numeric(prescribing_rate))

  colnames(d_prescriptions) <- c("state","state_abbreviation", "prescribing_rate")

  d_prescriptions <- d_prescriptions %>%
    filter(state_abbreviation != "US")

  d_prescriptions$year <- rep(url_list$year[i], nrow(d_prescriptions))

  # Store the results
  all_data <- rbind(all_data, d_prescriptions)

}
```

Now we have extracted the data, we plot the prescriptions results for each state over time

```
ggplot(all_data, aes(x = year, y=prescribing_rate)) +
  geom_line(aes(colour=state_abbreviation)) +
  labs(title = "Prescriptions results for each state over time", x = "Year", y = "Prescribing Rate")
```


Prescriptions results for each state over time
Prescribing Rate
Year
state_abbreviation
AK
AL
AR
AZ
CA
CO
CT
DC
DE
FL
GA
HI
IA
ID
IL
IN
KS
KY
LA
MA
MD
ME
MI
MN
MO
MS
MT
NC
ND
NE
NH
NJ
NM
NV
NY
OH
OK
OR
PA
RI
SC
SD
TN
TX
UT
VA
VT
WA
WI
WV
WY