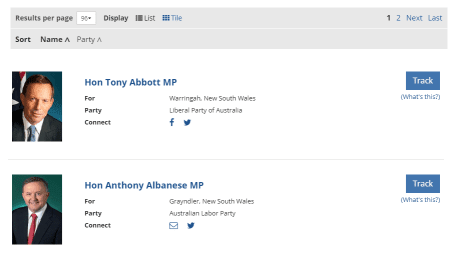
While there is more and more data available in structured formats (CSV, JSON) through initiatives like OpenData, sometimes nicely formatted data still not publicly available.

When I decided to conduct a little study of what Australian politicians from the major party post in Twitter. So I decide to find list of Twitter accounts of the Members of Parliaments with the references to what party they belong. Unfortunately my search for this data in any structured format wasn’t successful (if you know where to get it – you are welcome to comment).

The information I needed can be taken from [Parliament of Australia](https://www.aph.gov.au/Senators_and_Members/Parliamentarian_Search_Results?q=&mem=1&par=-1&gen=0&ps=96) website. The issue is that it is presented as HTML pages like one below:



It contains all the information I needed. Let’s see how it can be extracted using RSelenium package.

First we need to install the package and load it.

install.packages("RSelenium")  
library(RSelenium)

Also make an empty dataframe for the data I want to collect

mps <- data.frame(matrix(ncol = 4, nrow = 0))  
colnames(mps)<- c('name', 'electorate', 'party', 'twitter')

Next thing we need to do is to initialize Selenium driver. Selenium is different from other R packages used to scrap web. It actually loads web pages into real browser and allows you to work with that pages. Selenuim use is much wider than data scrapping: for example it is a very popular tool for automated testing of websites.

The code below initializes Google Chrome driver on local machine and open browser session controlled in our case from R.

rD <- rsDriver()  
remDr <- remoteDriver(port = 4567L, browserName = "chrome")  
remDr$open()

This is the only way I could run it on Windows 10 machine without use of Docker containers and other virtualization mechanisms. Firefox didn’t work for me. I admit that haven’t tried Safari and IE/Edge.

You may also run RSelenium sessions from remote machine including such services as BrowserStack.

Next prepare a function for scrapping the data about individual MP – name, electorate, party, twitter handle.

onempdata <- function (webElems, ref) {  
one <- data.frame(matrix(ncol = 4, nrow = 1))  
colnames(one)<- c('name', 'electorate', 'party', 'twitter')  
name <- webElems[[ref]]$findChildElement(using='class', value='title')  
one$name <- as.character(name$getElementText())  
electorate <- webElems[[ref]]$findChildElement(using='css selector', value='dd:nth-child(2)')  
one$electorate <- as.character(electorate$getElementText())  
## party data is badly structured, will get it another way  
### getting twitter, since not all MPs have it, we need to catch errors to avoid stop  
twitter <- tryCatch({  
suppressMessages({  
webElems[[ref]]$findChildElement(using = 'css selector', value = '.fa-twitter')  
})  
},  
error = function(e) {  
NA\_character\_  
}  
)  
# only collect Twitter handle if it exists  
if (class(twitter)!='character'){  
twitter$clickElement()  
Sys.sleep(4)  
windows <- remDr$getWindowHandles() #get list of open windows  
remDr$switchToWindow(windows[[2]]) # switch to Twitter window  
tt <- as.character(remDr$getCurrentUrl())# collect URL  
remDr$closeWindow() # Close Twitter window  
remDr$switchToWindow(windows[[1]]) #switch back to main window  
one$twitter <- as.character(tt)  
}  
else one$twitter <- NA\_character\_ #if no Twitter return NA  
# return one row dataframe with all data about one MP  
return(one)  
}

Several notes about the code above:

* remDr$findElements is the main function to search web page content. The search can be done using xpath, css selector, classes. Variation of the function called like webElems[[ref]]$findChildElement look for the child objects within webElems[ref] object. There is a nice Chrome plugin to find web page elements xpath or class called [SelectorGadget](https://selectorgadget.com/)
* To extract text getElementText() < function that can be applied to webElement class of objects used
* We need to process exceptions to avoid function stopping when no Twitter handle available (yes, not all MPs have it)
* I was unable to find a way to extract Twitter handle directly from the web page. Maybe it is possible, but I haven’t found a way to get it. So I found a work around. R code controlled browser user actually visits Twitter accounts if they exists and we capture that page URL to extract Twitter handle later

Next we define a function to get data about all MPs who are member of particular party. We use here APH website feature that allows to filter MPs by party, so each party has specific URL like [this one](https://www.aph.gov.au/Senators_and_Members/Parliamentarian_Search_Results?q=&mem=1&par=1&gen=0&ps=96) displays only Labor MPs.  
  
collectPartyData <- function (url, party){  
remDr$navigate(url)  
# empty dataframe to collect one party data and return as a function result  
fmps <- data.frame(matrix(ncol = 4, nrow = 0))  
colnames(fmps)<- c('name', 'electorate', 'party', 'twitter')  
webElems1 <- remDr$findElements(using = 'xpath', value = '//\*[contains(concat( " ", @class, " " ), concat( " ", "padding-top", " " ))]')  
for (i in seq\_along(webElems1)){  
one <- data.frame(matrix(ncol = 4, nrow = 1))  
colnames(one)<- c('name', 'electorate', 'party', 'twitter')  
one <- onempdata(webElems1, i)  
one$party <- party  
fmps <- rbind(fmps, one)  
}  
return (fmps)  
}

Now it is time to run the data collection using the functions defined.  
## run with Labour MPS  
url <- "https://www.aph.gov.au/Senators\_and\_Members/Parliamentarian\_Search\_Results?q=&mem=1&par=1&gen=0&ps=96"  
mps <- rbind(mps,collectPartyData(url, "Australian Labor Party"))  
## run with Liberal MPS  
url <- "https://www.aph.gov.au/Senators\_and\_Members/Parliamentarian\_Search\_Results?q=&mem=1&par=2&gen=0&ps=96"  
mps <- rbind(mps,collectPartyData(url, "Liberal Party of Australia"))  
## run with Nationals  
url <- "https://www.aph.gov.au/Senators\_and\_Members/Parliamentarian\_Search\_Results?q=&mem=1&par=3&gen=0&ps=96"  
mps <- rbind(mps,collectPartyData(url, "The Nationals"))  
## run with Greens  
url <- "https://www.aph.gov.au/Senators\_and\_Members/Parliamentarian\_Search\_Results?q=&mem=1&par=9&gen=0&ps=96"  
mps <- rbind(mps,collectPartyData(url, "Australian Greens"))  
## run with independent  
url <- "https://www.aph.gov.au/Senators\_and\_Members/Parliamentarian\_Search\_Results?q=&mem=1&par=4&gen=0&ps=96"  
mps <- rbind(mps,collectPartyData(url, "Independent"))  
## run with Centre Alliance  
url <- "https://www.aph.gov.au/Senators\_and\_Members/Parliamentarian\_Search\_Results?q=&mem=1&par=25&gen=0&ps=96"  
mps <- rbind(mps,collectPartyData(url, "Centre Alliance"))  
### run with Katter party  
url <- "https://www.aph.gov.au/Senators\_and\_Members/Parliamentarian\_Search\_Results?q=&mem=1&par=15&gen=0&ps=96"  
mps <- rbind(mps,collectPartyData(url, "Katter's Australian Party"))  
### closing webdriver session  
remDr$close()   
Here it is, we now have a nice data frame with all MPs with their parties, electorate and Twitter account (if they have it).  
We can apply a quick formatting to leave just Twitter handle/account name part of URL, it is better for future use and save the results in CSV file.

mps$twitter <- gsub("https://twitter.com/","", mps$twitter)  
write.csv(mps, "mps.csv")

That’s it, with help of RSelenium we have collected the data from the website and can use it for further analysis, which will be part of my future posts.  
If you don’t really keen to explore data collection and just need the resulting CSV file you can take it from [Github](https://github.com/alevashov/aupolitics/blob/master/mps.csv).

Full code is also available in my [Github repository](https://github.com/alevashov/aupolitics), the file related with this post is [date-collection.R](https://github.com/alevashov/aupolitics/blob/master/data-collection.R)