Web Scraping in R using Docker and Selenium

Cengiz Zopluoglu

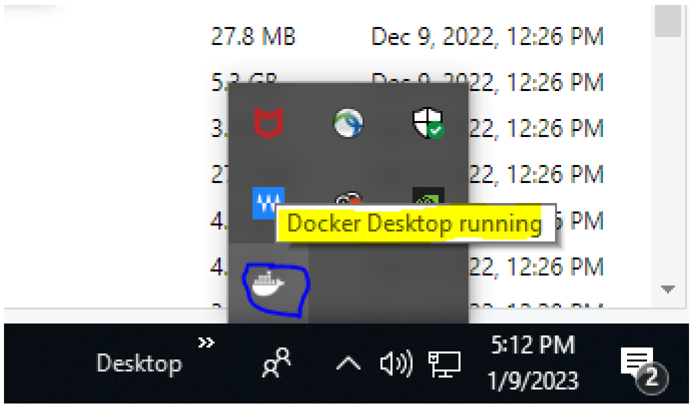
2023-01-9

This is a brief tutorial on how to scrape data from webpages if you would like to work with unstructured data for data visualization tasks. There are two tools you should install before you can run the R code I will provide for demonstration.

Docker Desktop

The first tool is **Docker**. In simple terms, Docker is a software platform that simplifies the process of building, running, managing and distributing applications. It does this by virtualizing the operating system of the computer on which it is installed and running. You can download and install it from the following link: https://docs.docker.com/desktop/. You should choose the appropriate version based on your operating system (Mac, Windows, Linux).

Once you install it, it will restart your computer and you should see an image like below indicating that Docker is installed properly running once your computer restarts. You can always stop and close Docker Desktop, and re-start it when you need it.



SelectorGadget - Chrome Plugin

Another tool is a Chrome plugin for CSS generator. After having installed the extension, go to any webpage and launch it using the Chrome browser. A box will open in the bottom right of the website. Click on a page element that you would like your selector to match (it will turn green). SelectorGadget will then generate a minimal CSS selector for that element, and will highlight (yellow) everything that is matched by the selector. Now click on a highlighted element to remove it from the selector (red), or click on an unhighlighted element to add it to the selector. Through this process of selection and rejection, SelectorGadget helps you come up with the perfect CSS selector for your needs.

You can download it from the following link.

https://chrome.google.com/webstore/detail/selectorgadget/mhjhnkcfbdhnjickkkdbjoemdmbfginb?hl=en

There are also Youtube videos demonstrating how to use SelectorGadget to extract CSS elements from a webpage.

Scraping data from Eugene Police Department Dispatch Logs

The code below demonstrate how to use Docker and RSelenium to scrape data from webpages. First, you should install and load the three packages: `RSelenium`, `rvest`, and `xml2`.

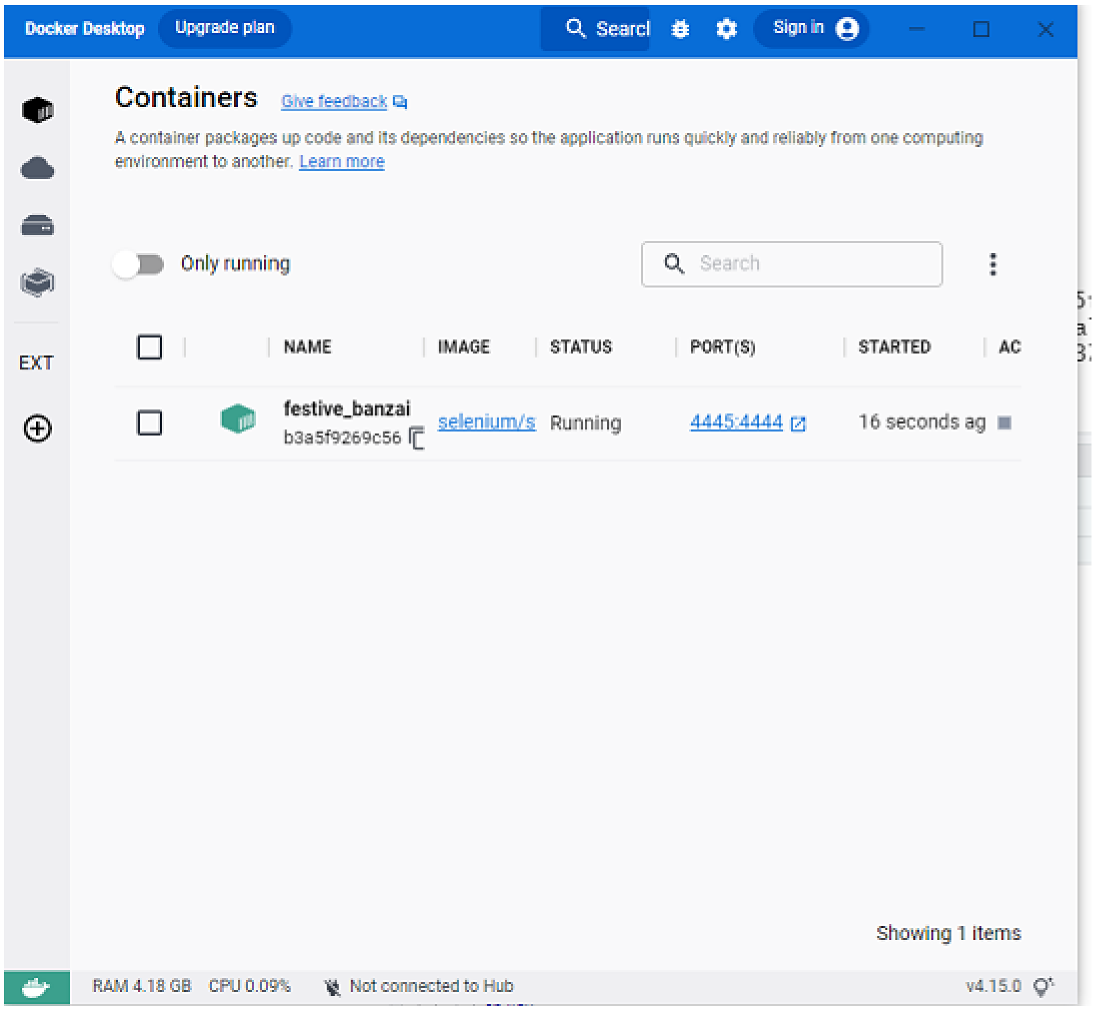
```r
# install.packages('RSelenium')
# install.packages('rvest')
# install.packages('xml2')

require(RSelenium)
require(rvest)
require(xml2)
```

Then, run the following code to start a Docker container (a virtual machine) in your computer.

```r
system('docker run -d -p 4445:4444 selenium/standalone-chrome')
```

```
## [1] 125
```

After this, you should see a container running in your Docker Desktop. The name is always random.



A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

	NAME	IMAGE	STATUS	PORT(S)	STARTED	AC
	festive_banzai b3a5f9269c56	selenium/s	Running	4445:4444	16 seconds ag	

Showing 1 items

RAM 4.18 GB CPU 0.09% Not connected to Hub v4.15.0

Now, we will start a Chrome Browser in this virtual machine and ask to navigate to the target url we want to scrape data. In this case, the url is http://coeapps.eugene-or.gov/EPDDispatchLog/Search

```r
remDr <- RSelenium::remoteDriver(remoteServerAddr = "localhost",
                                 port = 4445L,
                                 browserName = "chrome")
remDr$open()
```

```
## [1] "Connecting to remote server"
## $acceptInsecureCerts
## [1] FALSE
##
## $browserName
## [1] "chrome"
##
## $browserVersion
## [1] "108.0.5359.124"
##
## $chrome
## $chrome$chromedriverVersion
## [1] "108.0.5359.71 (1e0e3868ee06e91ad636a874420e3ca3ae3756ac-refs/branch-heads/5359@{#1016})"
```

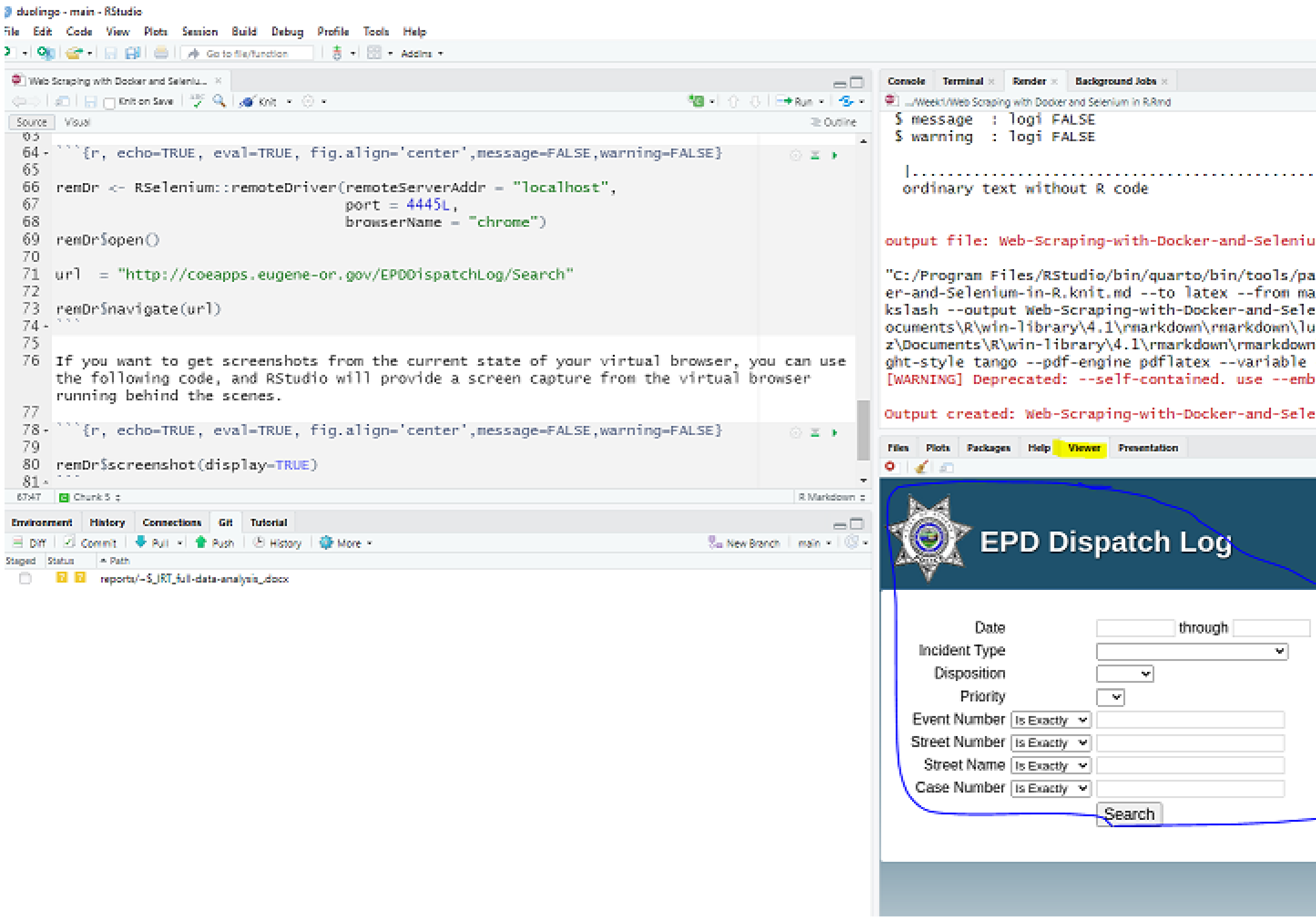
```
##
## $chrome$userDataDir
## [1] "/tmp/.com.google.Chrome.56JPcG"
##
##
## $`goog:chromeOptions`
## $`goog:chromeOptions`$debuggerAddress
## [1] "localhost:32821"
##
##
## $javascriptEnabled
## [1] TRUE
##
## $nativeEvents
## [1] TRUE
##
## $networkConnectionEnabled
## [1] FALSE
##
## $pageLoadStrategy
## [1] "normal"
##
## $platform
## [1] "ANY"
##
## $platformName
## [1] "ANY"
##
## $proxy
## named list()
##
## $`se:bidiEnabled`
## [1] FALSE
##
## $`se:cdp`
## [1] "ws://172.17.0.2:4444/session/d88f403666dc320915a6fb9d1c48bb93/se/cdp"
##
## $`se:cdpVersion`
## [1] "108.0.5359.124"
##
## $`se:vnc`
## [1] "ws://172.17.0.2:4444/session/d88f403666dc320915a6fb9d1c48bb93/se/vnc"
##
## $`se:vncEnabled`
## [1] TRUE
##
## $`se:vncLocalAddress`
## [1] "ws://172.17.0.2:7900"
##
## $setWindowRect
## [1] TRUE
##
## $strictFileInteractability
## [1] FALSE
```

```
##
## $timeouts
## $timeouts$implicit
## [1] 0
##
## $timeouts$pageLoad
## [1] 300000
##
## $timeouts$script
## [1] 30000
##
##
## $unhandledPromptBehavior
## [1] "dismiss and notify"
##
## $version
## [1] ""
##
## $`webauthn:extension:credBlob`
## [1] TRUE
##
## $`webauthn:extension:largeBlob`
## [1] TRUE
##
## $`webauthn:virtualAuthenticators`
## [1] TRUE
##
## $id
## [1] "d88f403666dc320915a6fb9d1c48bb93"
```

```
url  = "http://coeapps.eugene-or.gov/EPDDispatchLog/Search"

remDr$navigate(url)
```

If you want to get screenshots from the current state of your virtual browser, you can use the following code, and RStudio will provide a screen capture from the virtual browser running behind the scenes.

```
remDr$screenshot(display=TRUE)
```


duolingo - main - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Web Scraping with Docker and Seleniu...
Source Visual
Console Terminal Render Background Jobs
.../Week1/Web Scraping with Docker and Selenium in R.Rmd
Environment History Connections Git Tutorial
reports/~$_IRT_full-data-analysis_.docx
Files Plots Packages Help Viewer Presentation
EPD Dispatch Log
Search

~~~
```{r, echo=TRUE, eval=TRUE, fig.align='center',message=FALSE,warning=FALSE}

remDr <- RSelenium::remoteDriver(remoteServerAddr = "localhost",
                                 port = 4445L,
                                 browserName = "chrome")
remDr$open()

url  = "http://coeapps.eugene-or.gov/EPDDispatchLog/Search"

remDr$navigate(url)
~~~

If you want to get screenshots from the current state of your virtual browser, you can use
the following code, and RStudio will provide a screen capture from the virtual browser
running behind the scenes.

```{r, echo=TRUE, eval=TRUE, fig.align='center',message=FALSE,warning=FALSE}

remDr$screenshot(display=TRUE)
```
```

```
 $ message  : logi FALSE
 $ warning  : logi FALSE

  |.............................................
  ordinary text without R code

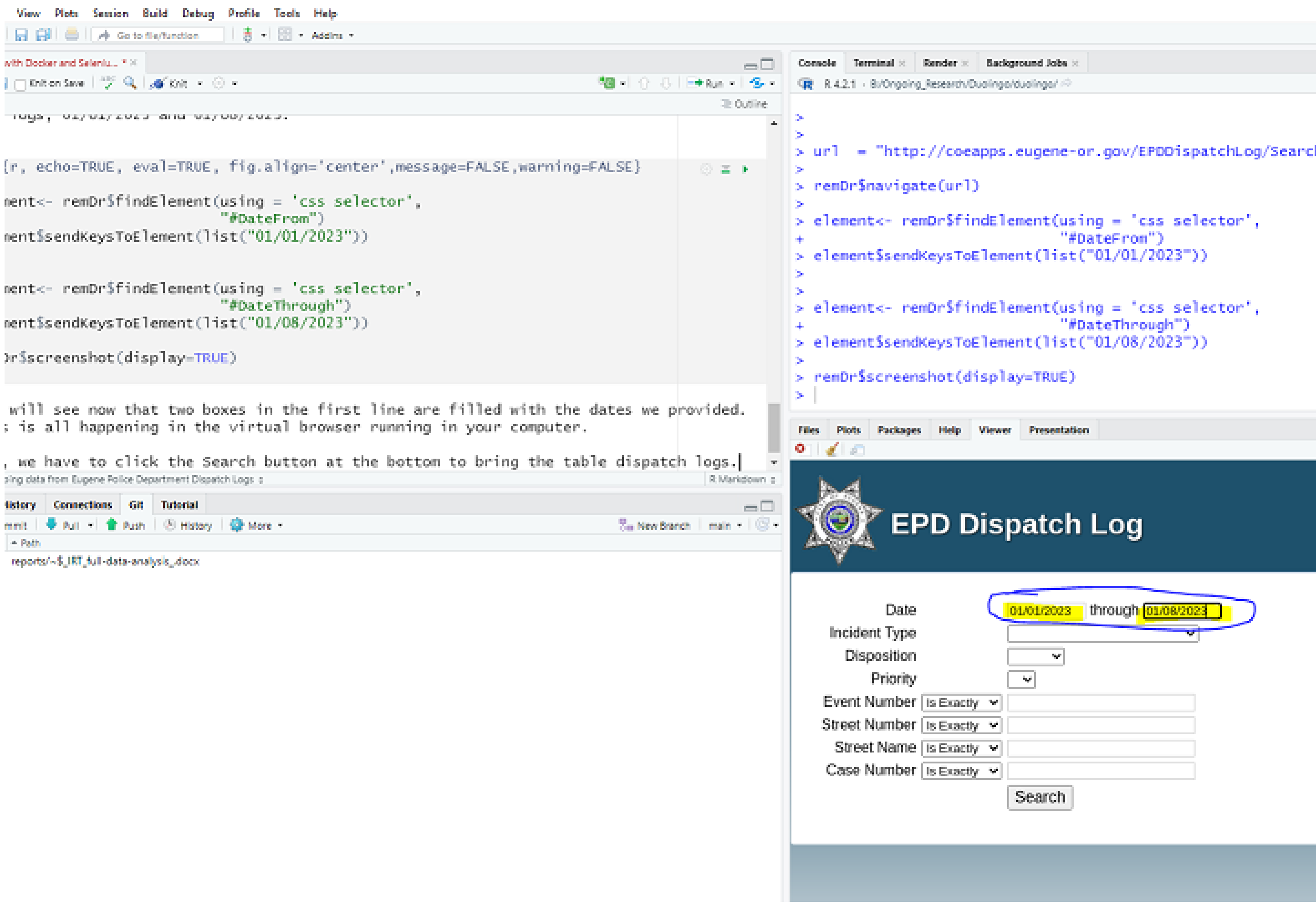
output file: Web-Scraping-with-Docker-and-Seleniu

"C:/Program Files/RStudio/bin/quarto/bin/tools/pa
er-and-Selenium-in-R.knit.md --to latex --from ma
kslash --output Web-Scraping-with-Docker-and-Sele
ocuments\R\win-library\4.1\rmarkdown\rmarkdown\lu
z\Documents\R\win-library\4.1\rmarkdown\rmarkdown
ght-style tango --pdf-engine pdflatex --variable
[WARNING] Deprecated: --self-contained. use --emb

Output created: Web-Scraping-with-Docker-and-Sele
```

Date		through
Incident Type		
Disposition		
Priority		
Event Number	Is Exactly	
Street Number	Is Exactly	
Street Name	Is Exactly	
Case Number	Is Exactly	

Using the SelectorGadget, we can figure out the CSS element for the two boxes appearing in the first line, **Date**. The code below is going to type two dates we want to search the logs, 01/01/2023 and 01/08/2023.

```
element<- remDr$findElement(using = 'css selector',
                             "#DateFrom")
element$sendKeysToElement(list("01/01/2023"))

element<- remDr$findElement(using = 'css selector',
                             "#DateThrough")
element$sendKeysToElement(list("01/08/2023"))

remDr$screenshot(display=TRUE)
```

You will see now that two boxes in the first line are filled with the dates we provided. This is all happening in the virtual browser running in your computer.
```



```
{r, echo=TRUE, eval=TRUE, fig.align='center',message=FALSE,warning=FALSE}

nent<- remDr$findElement(using = 'css selector',
                          "#DateFrom")
nent$sendKeysToElement(list("01/01/2023"))


nent<- remDr$findElement(using = 'css selector',
                          "#DateThrough")
nent$sendKeysToElement(list("01/08/2023"))

Dr$screenshot(display=TRUE)
```

will see now that two boxes in the first line are filled with the dates we provided.
s is all happening in the virtual browser running in your computer.

, we have to click the Search button at the bottom to bring the table dispatch logs.

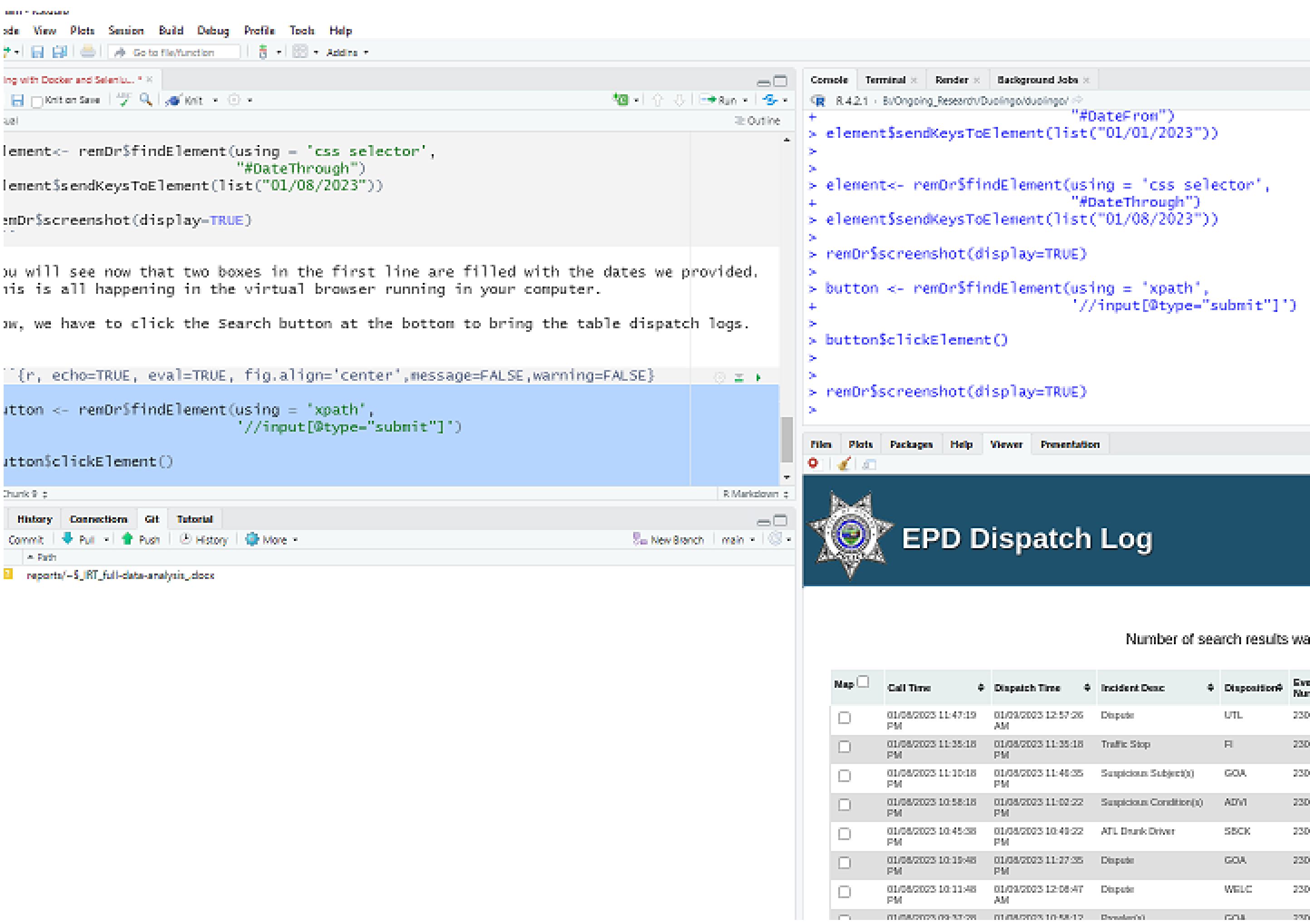
```
>
>
> url  = "http://coeapps.eugene-or.gov/EPDDispatchLog/Searc
>
> remDr$navigate(url)
>
> element<- remDr$findElement(using = 'css selector',
+                               "#DateFrom")
> element$sendKeysToElement(list("01/01/2023"))
>
>
> element<- remDr$findElement(using = 'css selector',
+                               "#DateThrough")
> element$sendKeysToElement(list("01/08/2023"))
>
> remDr$screenshot(display=TRUE)
>
```

EPD Dispatch Log

Date	01/01/2023 through 01/08/2023
Incident Type	
Disposition	
Priority	
Event Number	Is Exactly
Street Number	Is Exactly
Street Name	Is Exactly
Case Number	Is Exactly

Now, we have to click the Search button at the bottom to bring the table dispatch logs.

```r
button <- remDr$findElement(using = 'xpath',
                             '//input[@type="submit"]')

button$clickElement()


remDr$screenshot(display=TRUE)
```



```
lement<- remDr$findElement(using = 'css selector',
                           "#DateThrough")
lement$sendKeysToElement(list("01/08/2023"))

emDr$screenshot(display=TRUE)

ou will see now that two boxes in the first line are filled with the dates we provided.
his is all happening in the virtual browser running in your computer.

ow, we have to click the Search button at the bottom to bring the table dispatch logs.

``{r, echo=TRUE, eval=TRUE, fig.align='center',message=FALSE,warning=FALSE}

utton <- remDr$findElement(using = 'xpath',
                           '//input[@type="submit"]')

utton$clickElement()
```

```
+                             "#DateFrom")
> element$sendKeysToElement(list("01/01/2023"))
>
>
> element<- remDr$findElement(using = 'css selector',
+                             "#DateThrough")
> element$sendKeysToElement(list("01/08/2023"))
>
> remDr$screenshot(display=TRUE)
>
> button <- remDr$findElement(using = 'xpath',
+                             '//input[@type="submit"]')
>
> button$clickElement()
>
>
> remDr$screenshot(display=TRUE)
>
```

Number of search results wa

Map ☐	Call Time	Dispatch Time	Incident Desc	Disposition	Eve Nur
☐	01/08/2023 11:47:19 PM	01/09/2023 12:57:26 AM	Dispute	UTL	220
☐	01/08/2023 11:35:18 PM	01/08/2023 11:35:18 PM	Traffic Stop	FI	220
☐	01/08/2023 11:10:18 PM	01/08/2023 11:46:35 PM	Suspicious Subject(s)	GOA	220
☐	01/08/2023 10:58:18 PM	01/08/2023 11:02:22 PM	Suspicious Condition(s)	ADVI	220
☐	01/08/2023 10:45:38 PM	01/08/2023 10:49:22 PM	ATL Drunk Driver	SBCK	220
☐	01/08/2023 10:19:48 PM	01/08/2023 11:27:35 PM	Dispute	GOA	220
☐	01/08/2023 10:11:48 PM	01/09/2023 12:08:47 AM	Dispute	WELC	220
☐	01/08/2023 09:37:28	01/08/2023 10:58:12	Prowler(s)	GOA	220

Finally, we can read the current html page with the table and save is as a dataframe in the R environment.

```r
html <- xml2::read_html(remDr$getPageSource()[[1]])

tab  <- as.data.frame(html_table(html_nodes(html, "table")))[,-(1:3)]

head(tab)
```

```
##                Call.Time          Dispatch.Time           Incident.Desc
## 1 01/08/2023 11:47:19 PM 01/09/2023 12:57:26 AM                 Dispute
## 2 01/08/2023 11:35:18 PM 01/08/2023 11:35:18 PM            Traffic Stop
## 3 01/08/2023 11:10:18 PM 01/08/2023 11:46:35 PM   Suspicious Subject(s)
## 4 01/08/2023 10:58:18 PM 01/08/2023 11:02:22 PM Suspicious Condition(s)
## 5 01/08/2023 10:45:38 PM 01/08/2023 10:49:22 PM        ATL Drunk Driver
## 6 01/08/2023 10:19:48 PM 01/08/2023 11:27:35 PM                 Dispute
##   Disposition Event.Number                    Location Priority Case
## 1         UTL     23006826          562 FENSTER ST, EUG        3   NA
## 2          FI     23006821      E 7TH AVE/PEARL ST, EUG        6   NA
## 3         GOA     23006814         2001 GARDEN AVE, EUG        3   NA
## 4        ADVI     23006806 E 29TH AVE/AMAZON PKWY, EUG        3   NA
## 5        SBCK     23006797          4740 ROYAL AVE, EUG        3   NA
## 6          GOA     23006789           1340 MILL ST, EUG        3   NA
```

After scraping all the data you need, you can process this data and create visualizations you are interested.