**Background**

**Anyone who’s bought groceries online recently has seen the huge increase in demand due to the COVID-19 outbreak and quarantines. In this post, you’ll learn how to buy groceries on Amazon using R! To do that, we’ll be using the RSelenium package. In case you’re not familiar, Selenium is a browser automation tool. It works like a normal browser, except that you write code to perform operations, such as navigating to websites, filling in forms online, clicking links and buttons, etc. This way, it’s similar to writing a macro in Excel – except for a web browswer.**

**Several different languages, including Python and R, have packages that allow you to use Selenium by writing code in their language. R’s package for this, as mentioned above, is RSelenium.**

**Getting started**

**To get started, we need to install RSelenium, which we can do using the standard install.packages command. I also recommend using a newer version of R, as some of RSelenium‘s dependencies won’t work on older R versions.**

**install.packages("RSelenium")**

**Once installed, we can start coding!**

**Getting the URL of each grocery item**

**The first step we need to do is to create a driver object, which we’ll do using the rsDriver function. Running the first line below will launch a browser window. RSelenium supports several browsers, including Firefox, Chrome, and Internet Explorer.**

**Our code will mostly revolve around using driver$client to navigate around webpages. To go to a specific webpage, like Amazon’s, we can use the navigate method.**

**# create driver object**

**driver <- rsDriver(port = 2000L,browser = "firefox")**

**# navigate to Amazon's homepage**

**driver$client$navigate("https://www.amazon.com")**

**Next, let’s define a vector of general items we want to search for.**

**items <- c("flour","butter","cereal", "eggs", "milk", "apples", "sugar")**

**Now, let’s break down what we need to do. For each item, we will:**

**1) Type in the name of the item in Amazon’s search box and submit the seach**

****

**2) Find the URLs of all non-sponsored items products on the results page**

****

**3) Click on the links for the top search results for each item**

****

**4) Check if each product is in stock**

****

**5) Click “Add to cart” for each in-stock product**

****

**6) Check out**

****

**Translating that into code, the first three points are below. Let’s cover a couple key points about the code. First, to search for elements on the webpage, we use the findElement method, where we pass HTML**

**attributes of that element to the method. For example, the item search box has the id = “twotabsearchtextbox”, which we can see in the code below. We can figure that out by looking at the source code behind the webpage, or by right clicking the search box, and going to “inspect element”.**

****

* **create empty list to hold URLs for the top products for each item all\_urls <- list()**
* **loop through each item in the items vector**

**for(item in items)**

**{**

**# Find the search bar box**

**item\_box <- driver$client$findElement(using = "id", value = "twotabsearchtextbox")**

* **Clear the search box item\_box$clearElement()**
* **Type in the item name (note: text must be inside of a list) item\_box$sendKeysToElement(list(item))**
* **Submit the search**

**item\_box$submitElement()**

* **Wait for the results page to come up Sys.sleep(5)**
* **Get the links within the "rush-component" span tags**

**spans <- driver$client$findElements(using = "class", value = "rush-component")**

**links <- lapply(spans, function(span) try(span$findChildElement(using = "class", value = "a-link-normal"), silent = TRUE))**

* **Filter out errors i.e. the result of some span tags above not having links links <- links[sapply(links, class) == "webElement"]**
* **Get URLS from link objects**

**urls <- unlist(sapply(links, function(link) link$getElementAttribute(" href")))**

* **Filter out links we don't need ("sponsored" products) urls <- unique(urls[!grepl("/gp/", urls)])**
* **Add URLs to list**

**all\_urls[[item]] <- urls[1:5]**

**}**

**RSelenium returns the links as they show up on the webpage i.e. links closer to the top of the search results will show up earlier in returned links. This means if we examine the first 5 URLs for an item (as we pull**

**above), it should correspond to the first 5 products in the search results for that item.**

**Now, for our purposes, we’re only going to get one product per item, but if the first product we check is not in stock, then we want to be able to check if other products for the item are available. Again, we’ll limit our search to the first 5 products i.e. first 5 URLs associated with each item. If you’re doing this on your own, this is something you could adjust.**

**Here’s the next section of code. In this section, we go to each product’s webpage, check if it’s in stock, and then add it to the cart if it’s currently available. If a product is in stock, then we skip the rest of the products for that item. We’ll use the clickElement method to click the “add to cart” button.**

**for(urls in all\_urls)**

**{**

**text <- ""**

**for(url in urls)**

**{**

* **Navigate to url, wait for the page to fully load driver$client$navigate(url)**

**Sys.sleep(5)**

* **Look for div tag stating if item is in stock or not**

**div <- try(driver$client$findElement(using = "id", value = "availability"))**

* **If page doesn't have this tag, assume it's not in stock if(class(div) == "try-error")**

**next**

**else**

**{**

**# Scrape text from div tag**

**text <- div$getElementText()[[1]]**

**break**

**}**

**}**

**if(text == "In Stock.")**

**{**

**add\_to\_cart <- driver$client$findElement(using = "class", value = "a-button-input")**

**add\_to\_cart$clickElement()**

**}**

**Sys.sleep(5)**

**}**

**In the code above we check if the page specifically states the product is in stock. If it says it’s in stock at a later date, then this will not add that product to the cart. However, you can modify this by changing the equality operator to a regular expression, like this:**

**if grepl("in stock", text, ignore.case = TRUE)**

**{**

**...}**

**Next, now that we have added each available product to our shopping cart, we can check out! Below, we’ll go to the checkout page, and login with a username and password.**

**# Navigate to shopping cart**

**driver$client$navigate("https://www.amazon.com/gp/cart/view.html?ref\_=nav\_cart")**

**# Find "Proceed to checkout" button**

**checkout\_button <- driver$client$findElement(using = "name", value = "proceedToRetailCheckout")**

* **Click checkout button checkout\_button$clickElement()**
* **Find username box**

**username\_input <- driver$client$findElement(using = "id", value = "ap\_email")**

**# Enter username info**

**username\_input$sendKeysToElement(list("TOP SECRET USERNAME"))**

**# Submit username**

**username\_input$submitElement()**

**# Find password box**

**password\_input <- driver$client$findElement(using = "id", value = "ap\_password")**

**# Enter password**

**password\_input$sendKeysToElement(list("TOP SECRET PASSWORD"))**

**# Submit password**

**password\_input$submitElement()**

* **Wait for page to load Sys.sleep(5)**

**One note – it’s not a good idea to store credentials in a script – ever. You can avoid this by using the keyring package.**

**Next, we can place our order.**

**# Find "place order" button**

**place\_order <- driver$client$findElement(using = "name", value = "placeYourOrder1")**

* **Submit "place order" button place\_order$submitElement()**

**However, before you place your order, you might need to update your address or payment info. For example,**

**you can start the “change address” process by using the code below. You’ll just need to add a few lines to select a different address or fill in a new one. Similarly, you could do the same for payment information.**

**# Find "change address" link**

**change\_address <- driver$client$findElement(using = "id", value = "addressChangeLinkId")**

* **Click "change address" link change\_address$clickElement()**

**…**