Lab Assignment 2 - Ingesting Data from APIs

Instructions

Please complete the exercises below. Submit your completed assignment as a PDF, HTML or Word document outputted by knitr or compiled manually showing both the code and output. (It should be in a similar format to this document).

Note that in this document code blocks are shown with a grey background and output from running the code blocks is displayed with ## preceding the output.

There may be some packages used in this assignment that you have not yet installed. In many cases the instructions are to "modify" the code provided, and it is implied in all cases that you should make sure the modified code successfully runs on your computer.

There are thousands of APIs in existence, many of which make data available that can be useful for behavioral research. Let's get started by making a connection to one using R, and downloading some data. The first API we'll connect to is the Guardian API (the Guardian is a daily UK newspaper).

1. Go to the documentation for the Guardian API and explore it. Use that website to answer the following questions:

- Which endpoint would we use to search for Guardian articles that contain some keyword?
- What is the URL to access this endpoint (including the protocol, domain, and path)?
- What is the daily rate limit for free use?
- What format are the results returned in?

The code below is based on the api_access_template.R with an additional loop at the end to combine the results into a dataframe. You can use my API key for this assignment, but the whole class is using it so please do not make more than 200 calls/day.

2. Familiarize yourself with what each line in the code chunk below is doing and *adjust the code to search for a keyword of your choice*. Make sure the code runs on your computer.

```
library(RCurl)#to make http connection to API
library(RJSONIO)#to transform JSON response to data frame

#input API key
api_key <- "30c3044f-edf3-4aed-8e73-c8a74814a410"

#documentation on parameters and endpoints here:
  #https://open-platform.theguardian.com/documentation/

#set parameter
parameter1 <- "covid"

#set parameter
parameter2 <- ""

#construct the URL for the request
url <- paste0("https://content.guardianapis.com/search?api-key=", api_key,
              "&q=", parameter1)
#url

#make the request
raw <- getURL(url)
#View(raw)#to view if you'd like to

#transform to dataframe
df <- fromJSON(raw, nullValue = NA)#convert from JSON to dataframe (and set null as NA)

length <- length(df$response$results)#check the number of results
length
```

```
## [1] 10
```

```
#combine all the results into a single dataframe
news_data <- data.frame()
for(i in 1:length)
{
  new_row <- as.data.frame(df$response$results[[i]])
  #this df$response$results is where the data we want is
  try({news_data <- rbind(news_data, new_row)}, silent=T)
  #the try() function allows it to fail and continue if needed
}
head(as.character(news_data$webTitle), 3) #show the first 3 titles
```

```
## [1] "What we actually know about Covid-19"
## [2] "Oxford University resumes Covid-19 vaccine trials"
## [3] "Israel enters unpopular second Covid lockdown"
```

Next you will modify the code to change the page size (i.e. how many results are returned per query). In the API documentation you can find that the name of the page size parameter is "page-size" and the maximum page size is 50.

3. Modify the code to return 50 results per query instead of 10.

---

4. Modify the code so there is a variable in R (such as "parameter2") that controls the target page number. Set this variable so the query requests page 2 of the results.

Note: you can consult the API documentation to find the name of the parameter that controls what page is returned. Hint: you'll need to modify the URL with the new parameter value.

---

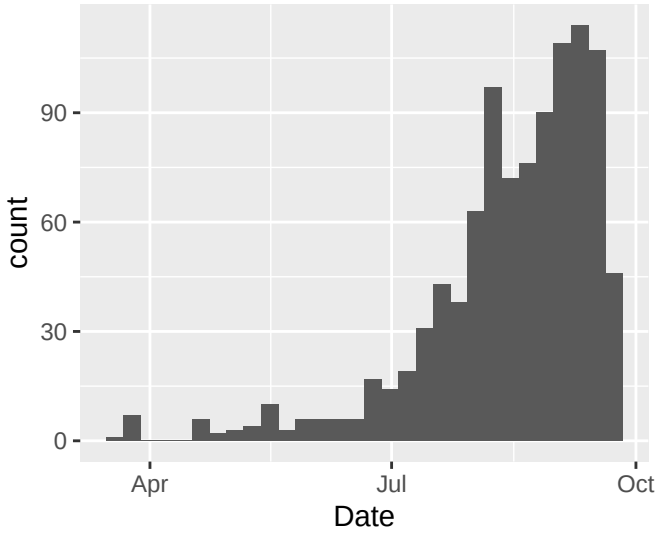
5. Now put the modified code into a loop to collect 20 "pages" worth of data.

You can use a **for loop** to iteratively run the code with different values set in the variable that determines what page is returned. You should have about 1,000 rows in your final dataset. Important: be sure to include the function Sys.sleep(2) so the code pauses for 2 seconds before each API call.

```
#A template for loop to get you started:
for(page in 1:20){

  Sys.sleep(2)

  "your code here"

}

nrow(news_data)#to check how many results were compiled
head(news_data$webTitle)#to see the first few titles
tail(news_data$webTitle)#to see the last few titles
```

Before we move on from the Guardian API task, let's work with the data a bit. Here is the code for generating a histogram of the dates each article was published:

```
library(ggplot2)
ggplot(news_data, aes(x=as.Date(webPublicationDate))) + geom_histogram() + xlab("Date")
```



6. Draw a histogram of the dates of the articles you retreived.

---

7. Would this likely be a valid measure of societal attention to your topic over time? Please explain your answer in some detail.

Now we turn to the Twitter API.

To access the Twitter API, there is a useful R package (rtweet) that simplifies the process in R. This one is probably the best out of a few R packages that attempt to do this.

Ultimately, the R package just interfaces with the Twitter API, so you can also consult the Twitter API documentation for details on its use.

The code below searches for the past 200 tweets that contain the keyword "covid":

```
library(rtweet)

#create API token
try({
  new_token <- get_token()#this part may only work when run manually in rstudio
           #and not in rmarkdown because it needs to load something in a browser
  saveRDS(new_token, "new_token.RDS")#I ran it in R studio and saved to load in RMD
  })
new_token <- readRDS("new_token.RDS")

#make a set of queries (this function makes more than 1 query to the API if needed)
tweets <- search_tweets(q="covid", n = 200, include_rts = FALSE, token=new_token)
head(tweets$text)#show the first six messages
```

```
## [1] "@realDonaldTrump @WhiteHouse #DemocracyIsDead\n#TrumpKilledDemocracy\n\nM..."
## [2] "#Margin recovery by Q4 &amp;amp; #Stability thereon says @TCS. #Tech #COVI..."
## [3] "Dr Fauci throws cold water on Trump's boasts that a COVID vaccine is i..."
## [4] "@PieterTrapman @AdamJKucharski @joel_c_miller @BillHanage Definitely.\n..."
## [5] "@CudiKids @Aiden_brown04 @Kwags4312 @MSHSL @MSHSLTim Playing now will ..."
## [6] "When are these COVID Marshall's coming into place?<U+0001F602><U+0001F..."
```

7. Modify the code to search using a keyword of interest to you. Also, make it return 10,000 messages instead of 200 *and* make it include the most popular tweets rather than the most recent tweets.

You will need to examine the search_tweets() documentation to determine how to change the kind of tweets we request. *Hint*: run ?search_tweets() to see the documentation in R Studio.

Note: the rate limit for this endpoint in the Twitter API is 180 calls per 15 minutes. Each 100 messages takes 1 call. So if you request 10,000 messages it will take 100 calls - therefore you can't repeat this more than once every fifteen minutes.

Now that we have some Tweets, let's work a little with the data.

8. Make a scatterplot of retweets count vs. favorites count and add a regression line using geom_smooth().

Use ggplot to do this. (See here for some help with this.)

---

9. Save both of the datasets you created today as .RDS files.

```
saveRDS(news_data, "guardian_data.RDS")
saveRDS(tweets, "twitter_data.RDS")
```

Now we turn to Google Trends data.

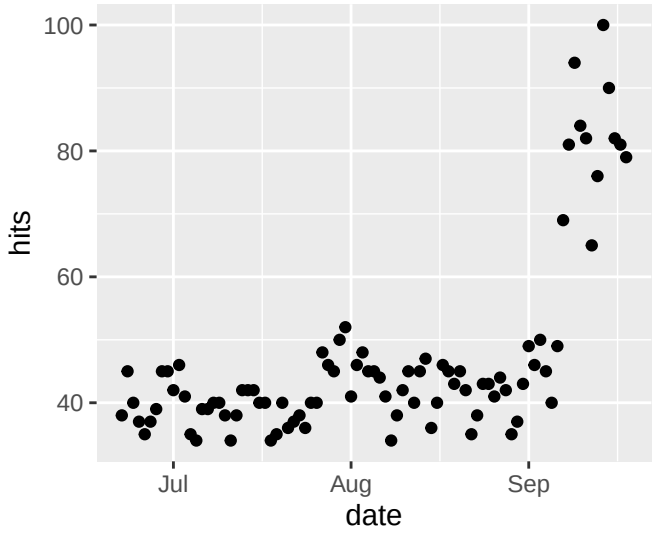
Here is some code to make a simple Google Trends query and extract the time series results. We'll use the gtrendsR package to do this.

```
library(gtrendsR)
#run the query (this one looks over past 3 months in UK)
gtrends <- gtrends(keyword="covid", geo="GB", time="today 3-m")

#extract the time series data frame
gtrends_by_time <- gtrends$interest_over_time
tail(gtrends_by_time)
```

```
##          date hits keyword geo      time gprop category
## 84 2020-09-13   76   covid  GB today 3-m   web        0
## 85 2020-09-14  100   covid  GB today 3-m   web        0
## 86 2020-09-15   90   covid  GB today 3-m   web        0
## 87 2020-09-16   82   covid  GB today 3-m   web        0
## 88 2020-09-17   81   covid  GB today 3-m   web        0
## 89 2020-09-18   79   covid  GB today 3-m   web        0
```

10. Run a gtrends query with the same keyword you used for the Guardian data collection. Store the result as shown in the code directly above and visualize the search volume over time using ggplot.

Hint: use the "date" column as the x variable and "hits" as the y variable Here's what it looks like for the keyword "covid":



OPTIONAL:

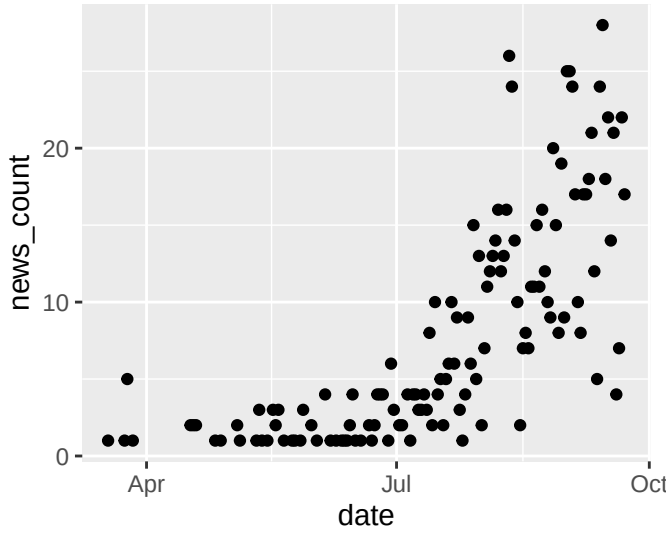
Try to merge the Guardian data and Google Trends data you collected. I'll walk you through it.

First, we need aggregate the Guardian data at a daily level to match the Google Trends data. We can use the package *dplyr* to do this.

```
#create a variable that holds the day each tweet was created
news_data$date <- as.Date(news_data$webPublicationDate)

#aggregate at the daily level using dplyr:
library(dplyr)
news_daily <- news_data %>% group_by(date) %>% summarise(news_count = n())
tail(news_daily)
```

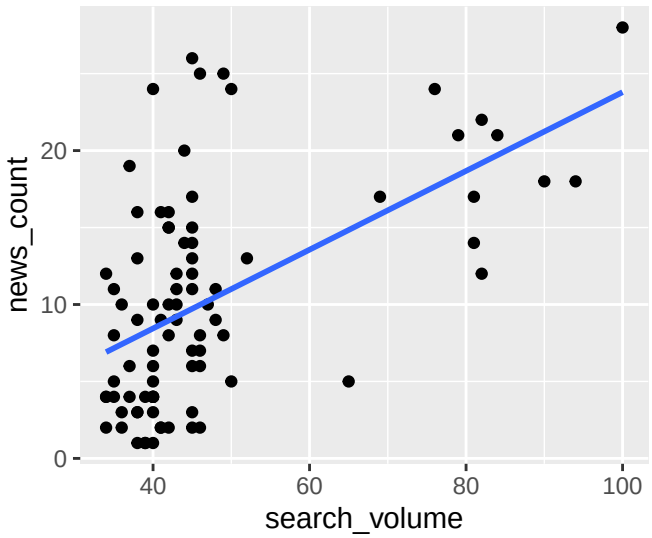
```
## # A tibble: 6 x 2
##   date       news_count
##   <date>          <int>
## 1 2020-09-17         14
## 2 2020-09-18         21
## 3 2020-09-19          4
## 4 2020-09-20          7
## 5 2020-09-21         22
## 6 2020-09-22         17
```

```
ggplot(news_daily, aes(date, news_count)) + geom_point() #plot it
```



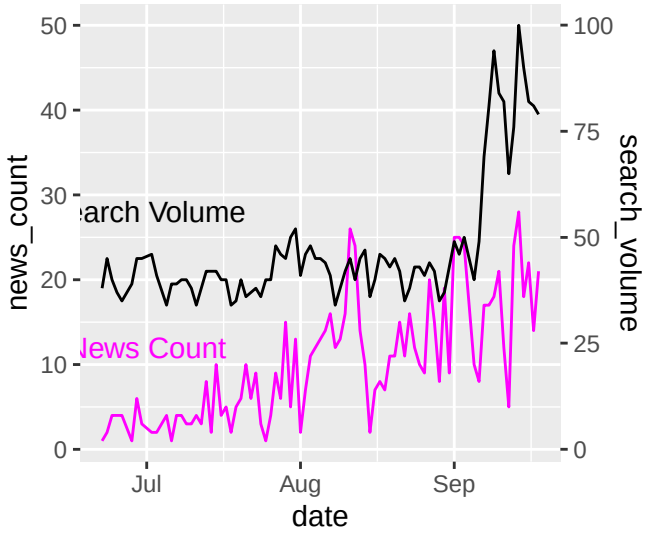
Next, we can merge the Guardian data and Google Trends data, then plot it.

```
gtrends_by_time$date <- as.Date(gtrends_by_time$date)#convert to date type for merging

merged <- merge(gtrends_by_time, news_daily, by="date")#merge

#plot the scatterplot of the search volume vs news articles:
ggplot(merged, aes(hits, news_count)) + geom_point() + geom_smooth(se=F, method="lm") +
  xlab("search_volume")
```



Now we can plot them both over time.

```
ggplot(merged, aes(date, news_count)) + geom_line(color="magenta") +
  geom_line(data=merged, aes(date, hits/2)) +
  scale_y_continuous(sec.axis = sec_axis(~.*2, name = "search_volume")) +#second y axis
  annotate("text", color="magenta", x=as.Date("2020-07-01"), y=12, label="News Count")+#labels
  annotate("text", color="black", x=as.Date("2020-07-01"), y=28, label="Search Volume")
```



Concluding remarks:

In this lab assignment, you first connected to an API (the Guardian API) without using an R package specifically designed for that API. Then, you connected to two commonly used APIs (the Twitter API and Google Trends) which each had R packages available to make the process of connecting and working with the data simpler.

I hope you feel empowered! With the knowledge of how to connect to APIs and collect data from them, a world of opportunities opens up to you regarding new datasets you can build.

Keep in mind that getting connected to an API and bringing in some data is the first step. The next step often is to develop a system that initiates scripts on a schedule to automate this process so you can build sizable datasets by running 10,000s or 100,000s of queries (more queries than you could expect to run in one session without failing). Some examples of such systems will be presented in lecture.