Using APIs in R

Promothesh Chatterjee*

*Copyright© 2024 by Promothesh Chatterjee. All rights reserved. No part of this note may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the author.

Application Programming Interface

We use an application-programming interface (API) to communicate between software programs. How is this useful? Assume you want a table of economic growth for last five years from World Bank website. You can go about it in a few ways:

1) You can go to the website, find the relevant data, the manually copy-paste the data into a spreadsheet and then import into R. This method is tedious, error-prone and inefficient.

2) You can use some of the web-scraping techniques, we have talked about previously. The issue here is if World Bank changes it's website, your code has to be adjusted.

3) You can use wbstats package in R as it allows researchers to quickly search and download the data of their particular interest in a programmatic and reproducible fashion from the World Bank's API. The benefit of this strategy is even if World Bank's web structure changes, it won't affect your code. Thus,where recurring data retrieval is needed for analysis, APIs provide a superior and consistent method. The number of organizations that provide APIs is growing very fast. For instance, there are almost 23,000 organizations that share their API (https://apilist.fun/). Even though, main objective of APIs is to give access to data, many now analyze data as well such as facial recognition APIs, voice to text APIs, data visualization APIs etc.

Fundamentals

A few fundamental things before you start working with an API.

1) We need to know the URL (Uniform Resource Locator or the web address) of the organization and what data (content, specific variables etc.) we need.

2) To access an API, we will need to provide some kind of identification and/or authorization. These are done using:

a) API key- Typically a key or token can be often obtained by giving name and email to the organization.

b) OAuth- OAuth is an open standard for authorization and provides credentials. R package httr has greatly simplified the complications underlying OAuth, so we will use an example to explain and demonstrate this. Let's first look at how we can use utilize readymade R packages for APIs.

R Package for API data feeds

The first step in working with APIs is to check if there is an R package available for it. This greatly simplifies our work. You check for the R API packages here- CRAN Task View: Web Technologies and Services web page,and on the rOpenSci web page. Let's take the example of getting financial data from a package called Quantmod.Let's see how APIs are utilized within the `quantmod` package to fetch and manipulate financial data. Note: Quantmod does not need API key and OAuth but most other R API packages do.

Introduction to APIs with Quantmod

```
# Install and load the quantmod package
if (!require(quantmod)) install.packages("quantmod")
library(quantmod)
```

Note: Initially, we ensure that `quantmod` is installed and loaded. The `quantmod` package in R is designed to interact with financial and economic data APIs. It abstracts API requests, making data retrieval straightforward without requiring detailed knowledge of how APIs work.

```
# Set a symbol for a stock (e.g., AAPL for Apple Inc.)
symbol = "AAPL"
```

Note: Here, `symbol` represents the ticker symbol for Apple Inc. This symbol is used as an identifier in API calls to retrieve financial data specifically for Apple's stock.

```
# Get stock data from Yahoo Finance
getSymbols(symbol, src = "yahoo", from = "2020-01-01", to = "2023-12-31")
```

```
## [1] "AAPL"
```

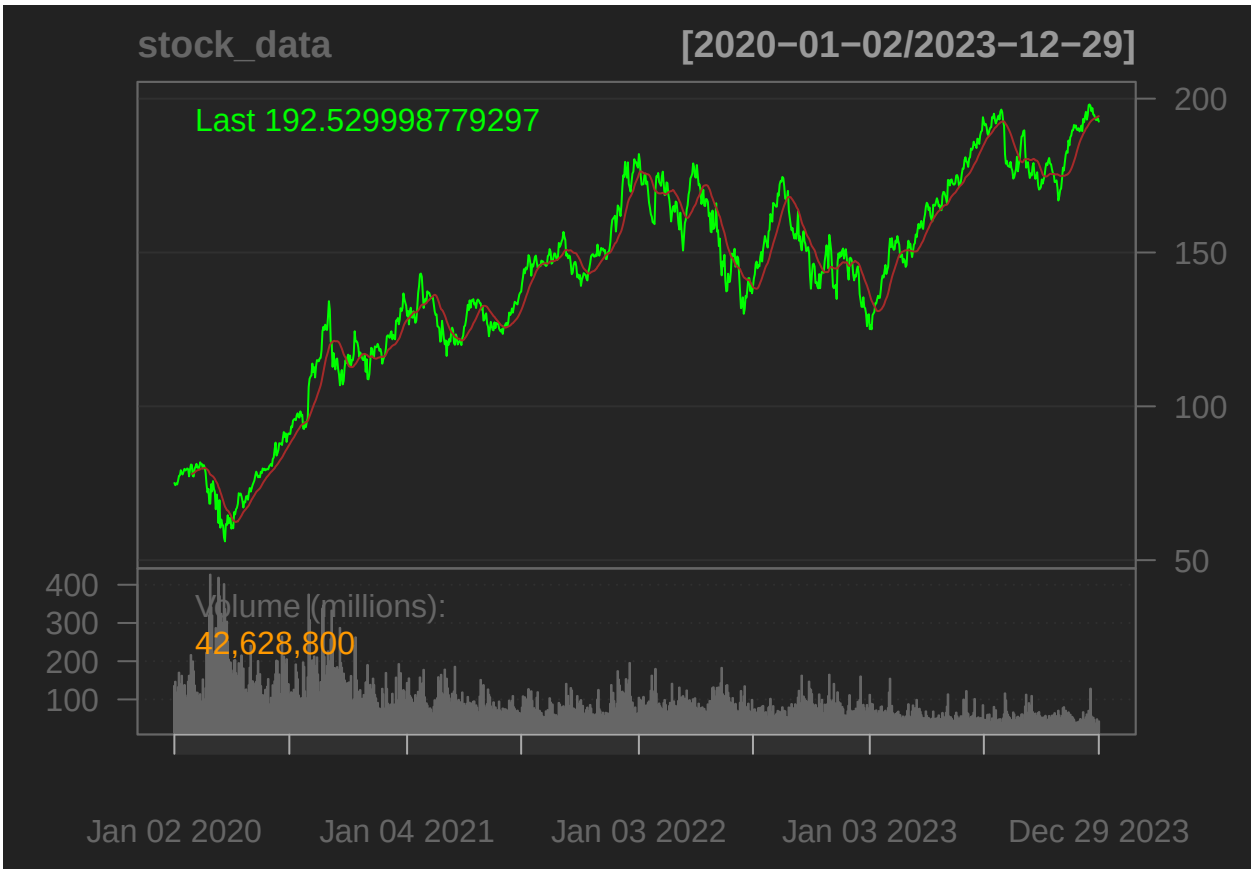
Note: `getSymbols()` is a function that makes an API call to Yahoo Finance (specified by `src="yahoo"`). It requests historical stock data for the given symbol (`AAPL`) over a specified date range. This function showcases how APIs are utilized to fetch data directly from financial databases without manual intervention.

Accessing and Visualizing Data

```
# Access the retrieved data
stock_data <- get(symbol)
```

Note: After data retrieval, `get(symbol)` accesses the fetched data stored in the R environment. This step illustrates how data returned from an API is managed and utilized within an analytical framework.

```
# Plot the closing prices
chartSeries(stock_data, type = "line", TA = "addVo();addSMA(20)", main = "AAPL Stock Price")
```



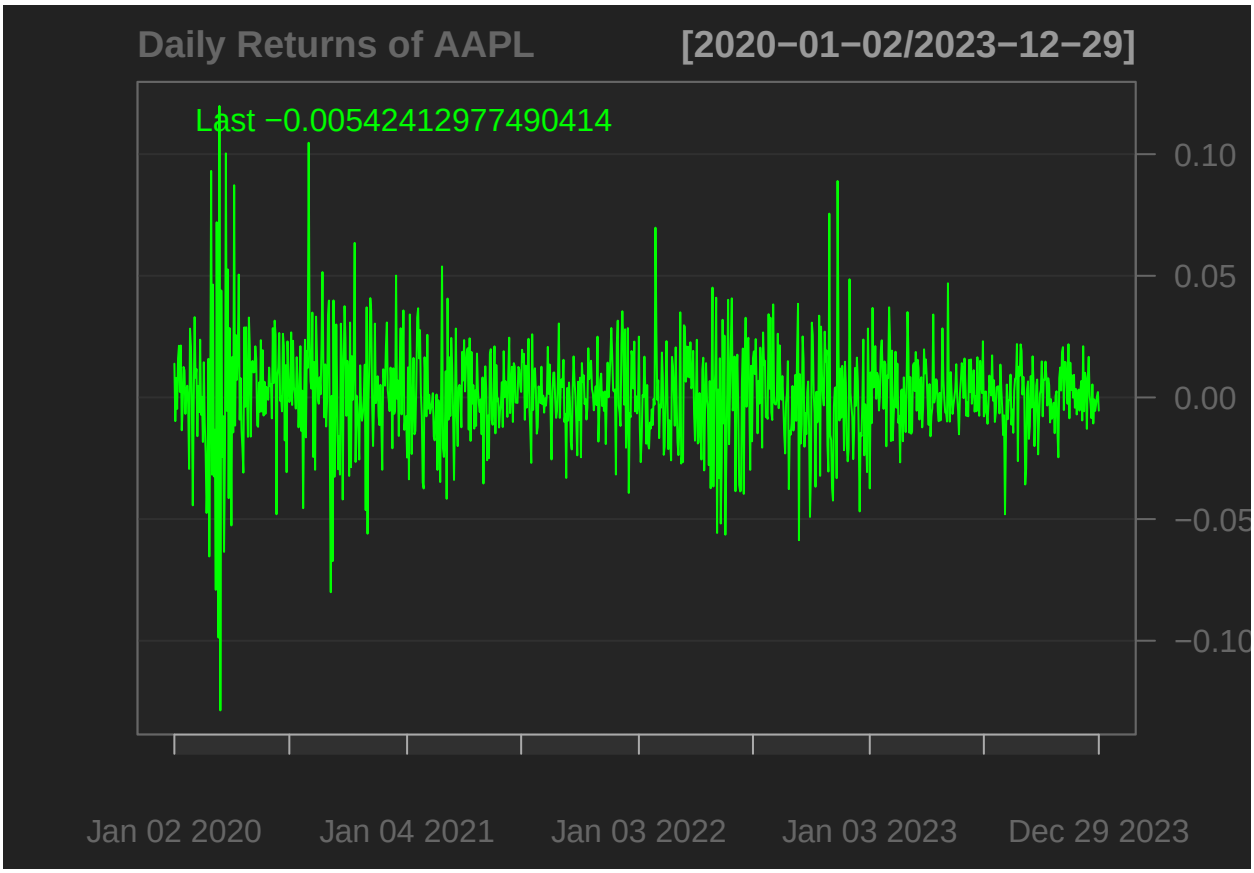
Note: Visualizing data is crucial for analysis. `chartSeries()` uses the data retrieved via the API to create a line chart of Apple's closing stock prices. Additional technical analysis overlays, like volume (addVo) and a 20-day simple moving average (addSMA), are added to the plot.

Advanced Financial Analysis

```
# Calculate daily returns
returns <- dailyReturn(AAPL)
```

This demonstrates the calculation of daily returns from the stock prices retrieved via the API. This step is essential for understanding the day-to-day performance and risk associated with the stock.

```
# Plot the returns
chartSeries(returns, name="Daily Returns of AAPL")
```

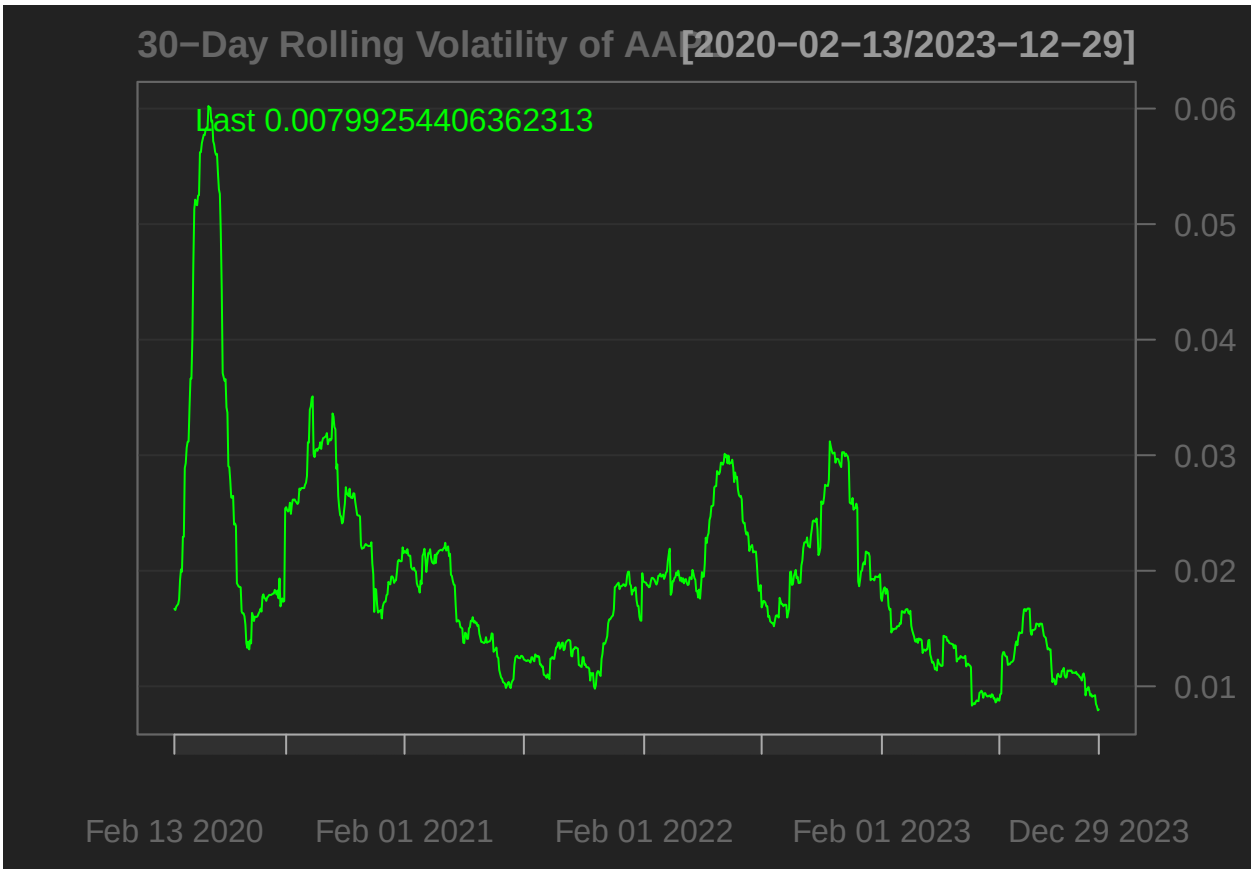


Note: Further plotting highlights the utility of these returns in visualizing financial volatility and investment risk.

```
# Calculate 30-day rolling standard deviation
rolling_sd <- runSD(returns, n = 30)
```

Computing rolling standard deviation emphasizes the analysis of stock price volatility over time, derived from API-provided data.

```
# Plot the rolling volatility
chartSeries(rolling_sd, name="30-Day Rolling Volatility of AAPL")
```



This visualization aids in understanding market dynamics and the financial stability of Apple, based on data fetched from Yahoo Finance via API.

There are many other functions in the package that you can explore: https://github.com/joshuaulrich/quantmod?tab=readme-ov-file

HTTR Package from Tidyverse

In the absence of an R package for the API of interest, we can use the HTTR package from Tidyverse to get the data of interest. The major HTTP verbs are:

1) GET: This is used to fetch data from an existing resource. We will use this mostly to get the text that we need.
2) POST: is used to create a new resource. As the name suggests, we are posting data to new resource.
3) PUT: is to update an existing resource.
4) DELETE: deletes an existing resource.

HTTP is the foundation for APIs; understanding how it works is the key to interacting with all the diverse APIs out there. An excellent beginning resource for APIs (including HTTP basics) is this simple guide: https://cdn.zapier.com/storage/learn_ebooks/e06a35cfcf092ec6dd22670383d9fd12.pdf

Let's utilize the HTTR package to search for articles from New York Times (even though there exists an R package for NY Times API). You have to create a developer account with NY Times to get a Client ID and API key.

```
library(httr)

apikey<-'omZ73xAKdSL2V3gY7DZn.....'# truncated NY Times apikey

base_url <- "http://api.nytimes.com/svc/search/v2/articlesearch.json" # Where to query
pricing <- GET(base_url, query=list(q="pricing","api-key"=apikey)) # search term -
pricing
```

```
## Response [http://api.nytimes.com/svc/search/v2/articlesearch.json?q=pricing&api-key=omZ73xAKdSL2V3gY
##   Date: 2024-04-17 07:23
##   Status: 200
##   Content-Type: application/json
##   Size: 230 kB
```

From the output of pricing, we can see that the query was successful (Status: 200), the content is in json format, and its size is 225kB. We can actually specify a range of dates for the query term.

```
pricing <- GET(base_url, query=list(q="pricing",
                                    "api-key"=apikey,
                                    "begin_date"=20230101,
                                    "end_date"=20230401))
```

To extract the text returned by this API call, you can look at the variable content (that's where text is stored). You can write it to a file to take a look at it. We will use jsonlite package here to deal with the output which is JSON (Java Script Object Notation- which is a standard way of representing output across languages). If you use the function 'head' to look at a few datapoints in content, the output is in Unicode, thus, gibberish. We need to convert it to text, using library jsonlight. After running the code, we get the output as a list. Lists are a powerful way of representing data in R, though they can quickly become complicated as lists can not only store data of various types (numeric, character etc.) but can also store list (list nested in a list). Here is quick tutorial on lists, if you need a refresher: https://data-flair.training/blogs/r-list-tutorial/. Simple way to drill down the different elements is to put $ sign after the list and different elements for selection show up in RStudio and you can keep selecting elements of interest. We can then create a dataframe (or a tibble which is specialized kind of dataframe from tidyverse) using these elements.

Look here for greater details about using the jsonlite package:

https://cran.r-project.org/web/packages/jsonlite/vignettes/json-apis.html

```
library(jsonlite)
library(tibble)
data <-  httr::content(pricing, as = "text")
```

```
data <- fromJSON(data)
my_df<-tibble(data$response$docs$headline$main,data$response$docs$section_name, data$response$docs$abst
# Rename the columns
names(my_df)[1] <- "Title"
names(my_df)[2] <- "Section"
names(my_df)[3] <- "Abstract"
names(my_df)[4] <- "Date"

knitr::kable(tibble(my_df))
```

Title	Section	Abstract	Date
California's Plan for Cheaper Insulin Collides With Big Pharma's Price Cuts	Health	The state awarded a $50 million contract to produce less costly treatments, but moves by major suppliers might undercut the initiative before any new product emerges.	2023-03-24T15:00:11+0000
Falling Lithium Prices Are Making Electric Cars More Affordable	Business Day	An unexpected decline in the price of an essential battery material, along with those of other commodities, is good news for buyers. But experts disagree on how long low prices will last.	2023-03-20T09:00:24+0000
Are Big Profits Keeping Prices High? Some Central Bankers Are Concerned.	Business Day	Companies' rising profit margins could be contributing to persistent inflation, a European Central Bank policymaker says.	2023-03-31T13:19:52+0000
M.T.A. Plans to Use Congestion Pricing Funds to Address Bronx Pollution	New York	Transit officials are thinking of spending millions of dollars to fight pollution in some of New York's poorest and most contaminated neighborhoods.	2023-03-28T22:39:59+0000
Sanofi Plans to Cut the Price of Insulin	Business Day	The company is the third of the three major insulin manufacturers that dominate the U.S. market to announce such a move this month.	2023-03-16T22:22:09+0000
Novo Nordisk Says It Will Slash the Price of Insulin	Business Day	The company's decision followed a similar move by its rival Eli Lilly this month.	2023-03-14T16:40:56+0000
The Fed's Preferred Inflation Gauge Cooled Notably in February	Business Day	A closely watched measure of price increases provided encouraging news as the Fed considers when to stop raising rates.	2023-03-31T13:09:45+0000
A Federal Tool Could Soon Make It Easier to Compare Credit Cards	Your Money	The Consumer Financial Protection Bureau, which already provides some card details, is asking banks for more information about who qualifies for which cards.	2023-03-31T13:00:07+0000

Title	Section	Abstract	Date
How a TikToker Brought Hundreds of Transplants to a Midwestern City	Real Estate	People hoping to become homeowners have answered the call to relocate to Peoria, Ill., where the housing is affordable and a one-person welcoming committee awaited them.	2023-03-29T09:00:39+0000
9 New Books We Recommend This Week	Books	Suggested reading from critics and editors at The New York Times.	2023-03-23T16:49:36+0000

Resources from where these notes have borrowed generously:

1) https://github.com/jennybc/yelpr

2) https://cran.r-project.org/web/packages/httr/vignettes/quickstart.html

3) https://afit-r.github.io/scraping

4) https://www.dataquest.io/blog/r-api-tutorial/

5) https://cfss.uchicago.edu/notes/write-an-api-function/

6) https://www.r-bloggers.com/accessing-apis-from-r-and-a-little-r-programming/

7) https://rtweet.info/articles/intro.html

8) https://themockup.blog/posts/2020-05-22-parsing-json-in-r-with-jsonlite/#dont-bury-the-lede