API Calls with R

Yue Xiong, LMU, yue.xiong@stat.uni-muenchen.de

Chapter 03 API Calls with R

APIs (application programming interfaces) are hosted on web servers. When you type www.google.com in your browser's address bar, your computer is actually asking the www.google.com server for a webpage, which it then returns to your browser. APIs work much the same way, except instead of your web browser asking for a webpage, your program asks for data. This data is usually returned in JSON format. To retrieve data, we make a request to a webserver. The server then replies with our data. In R, we will use the `httr` and `jsonlite` packages to deal with this.

R Setup

```
# install.packages(c("httr", "jsonlite", "tidyverse"))
library('httr')
```

```
## Warning: package 'httr' was built under R version 4.0.5
```

```
library('jsonlite')
```

```
## Warning: package 'jsonlite' was built under R version 4.0.5
```

```
library('tidyverse')
```

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

```
## Warning: package 'purrr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## Warning: package 'stringr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter()  masks stats::filter()
## x purrr::flatten() masks jsonlite::flatten()
## x dplyr::lag()     masks stats::lag()
```

Task example: Pull patents for Stanford University

Let's go to the Patents Endpoint (http://www.patentsview.org/api/patent.html) and find the appropriate field for the organization's name.

The variable that we need is called `"assignee_organization"` (organization name, if assignee is organization)

> *Note*: **Assignee**: the name of the entity - company, foundation, partnership, holding company or individual - that owns the patent. In this example we are looking at universities (organization-level).

Step 1. Build the URL query

Let's build our first URL query by combining the base url with one criterion (name of the `assignee_organization`)

base url: `http://www.patentsview.org/api/patents/query?q=` + criterion: `{"assignee_organization":stanford university"}`

```
# save the URL as a variable
indicator = "covid"
type = "daily"
country = "Germany"
region = "all"
daterange = "20210501-20210516"

path = paste0("https://covidmap.umd.edu/api/resources?indicator=", indicator,
                  "&type=", type, "&country=", country, "&daterange=", daterange)
```

Step 2. Get the response

Now let's get the response using the URL defined above, using the `httr` library.

```
request = GET(url = path)
```

Step 3. Check the Response Code

Before you can do anything with a website or URL in Python, it's a good idea to check the current status code of said portal.

The following are the response codes for the PatentsView API:

200 - the query parameters are all valid; the results will be in the body of the response

400 - the query parameters are not valid, typically either because they are not in valid JSON format, or a specified field or value is not valid; the "status reason" in the header will contain the error message

500 - there is an internal error with the processing of the query; the "status reason" in the header will contain the error message

Now, let us check the status of our response.

```
request$status_code  # should be 200, why 400 with the original url
```

```
## [1] 200
```

The status code is 200, which suggests valid query parameters. The results can be shown normally.

In the next steps, we are ready to get the content from the corresponding url,

Step 4. Get the Content

After a web server returns a response, you can collect the content you need by converting it into a JSON format.

JSON is a way to encode data structures like lists and dictionaries to strings that ensures that they are easily readable by machines. JSON is the primary format in which data is passed back and forth to APIs, and most API servers will send their responses in JSON format using the `jsonlite` package.

```
response = content(request, as = "text", encoding = "UTF-8")
head(response) # view json
```

```
## [1] "{\"data\":[{\"pct_covid\":0.005157,\"covid_se\":0.001358,\"pct_covid_unw\":0.005399,\"covid_se_
```

Step 5. Convert JSON to a normal R dataframe

```
df_survey <- fromJSON(response, flatten = TRUE) %>% data.frame()
head(df_survey, n = 10)  # check the dataframe
```

```
##    data.pct_covid data.covid_se data.pct_covid_unw data.covid_se_unw
## 1        0.005157      0.001358           0.005399          0.001123
## 2        0.004224      0.001060           0.004406          0.000983
## 3        0.006768      0.001987           0.005092          0.001108
## 4        0.005229      0.001260           0.005398          0.001122
## 5        0.006222      0.001581           0.005895          0.001200
## 6        0.004973      0.001236           0.005414          0.001178
## 7        0.003960      0.001117           0.004286          0.001037
## 8        0.005395      0.001439           0.005106          0.001168
```

```
## 9        0.005680      0.002062           0.004516          0.001093
## 10       0.002840      0.000856           0.004000          0.001031
##    data.sample_size data.country data.iso_code data.gid_0 data.survey_date
## 1              4260      Germany           DEU        DEU         20210501
## 2              4539      Germany           DEU        DEU         20210502
## 3              4124      Germany           DEU        DEU         20210503
## 4              4261      Germany           DEU        DEU         20210504
## 5              4071      Germany           DEU        DEU         20210505
## 6              3879      Germany           DEU        DEU         20210506
## 7              3966      Germany           DEU        DEU         20210507
## 8              3721      Germany           DEU        DEU         20210508
## 9              3764      Germany           DEU        DEU         20210509
## 10             3750      Germany           DEU        DEU         20210510
##     status
## 1  success
## 2  success
## 3  success
## 4  success
## 5  success
## 6  success
## 7  success
## 8  success
## 9  success
## 10 success
```

Descriptive stats based on the retrieved data.

```
summary(df_survey)
```

```
##  data.pct_covid     data.covid_se      data.pct_covid_unw data.covid_se_unw
##  Min.   :0.002840   Min.   :0.000856   Min.   :0.003403   Min.   :0.000946
##  1st Qu.:0.003894   1st Qu.:0.001039   1st Qu.:0.004376   1st Qu.:0.001035
##  Median :0.005065   Median :0.001248   Median :0.005099   Median :0.001115
##  Mean   :0.004833   Mean   :0.001339   Mean   :0.004934   Mean   :0.001107
##  3rd Qu.:0.005781   3rd Qu.:0.001615   3rd Qu.:0.005493   3rd Qu.:0.001171
##  Max.   :0.006768   Max.   :0.002062   Max.   :0.006012   Max.   :0.001250
##  data.sample_size data.country       data.iso_code       data.gid_0
##  Min.   :3526     Length:16          Length:16          Length:16
##  1st Qu.:3810     Class :character   Class :character   Class :character
##  Median :3958     Mode  :character   Mode  :character   Mode  :character
##  Mean   :3979
##  3rd Qu.:4140
##  Max.   :4539
##  data.survey_date      status
##  Length:16          Length:16
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

I will stop here as I am not sure whether I should be doing the checkpoint 1 and etc as the original url is invalid. Let's talk about this in future meetings.