First steps

Francesco Bailo

2022-03-03

Contents

1 About

2 First steps

2.1 Packages

The R code relies on generalist packages to access the API and manipulate the response.

```
library("httr")
library("jsonlite")
library("dplyr")
```

If you don't have these packages already install, you need to run

```
install.packages("httr")
install.packages("jsonlite")
install.packages("dplyr")
```

2.2 Credentials

Let's store our bearer token in an environment variable (let's call it `BEARER_TOKEN`)

```
Sys.setenv(BEARER_TOKEN = "copy-your-bearer-token-here")
```

We are then able to get the token back with

```
Sys.getenv("BEARER_TOKEN")
```

The idea is to run `Sys.setenv()` from our console *before* running our scripts (that is, every time!) so that our token is never added to a script file. Of course, if you don't care you can just store it in a regular variable.

2.3 Interrogating the Twitter API

The Twitter API accept two methods to exchange information: POST and GET. Intuitively, with the POST method we send information to a server while with the GET method we retrieve information. With the Twitter API, the GET method is used more frequently. Still, we need to use the POST method to define our search rules before we GET the Filtered stream.

This is how a GET request using the httr package looks like:

```
httr::GET(url,
          httr::add_headers(.headers = headers),
          query = params)
```

The `url` is a simple character variable while `headers` and `params` are lists.

But let's send a GET request!

We need first to set the URL, specify our request headers (these are not going to change, so you can place at the top of your document) and set the parameters fo the query.

```
url <-
  "https://api.twitter.com/2/tweets/counts/recent"

headers <-
  c(`Authorization` = sprintf('Bearer %s',
                              Sys.getenv("BEARER_TOKEN")))

params <-
  list(query = "from:TwitterDev",
       granularity = "day")
```

What are we doing here?

1. With `url` we specify the endpoint we want to use for this API request. The Twitter API has several endpoints. Note that sometimes we need to include parameters here instead of passing them through the HTTP query.

2. `headers` is the first layer of information that we send over to the server. In this case it contains our token. If this is accepted - the status of the request is `200 OK` - then the API is ready to process our request. If the token is not accepted we get as status `401 Unauthorized`. Note that these error codes and messages define the status of the HTTP request. The Twitter API has a different set of error codes. In this sense, we can get a `200 OK` from the HTTP layer and still get an error (e.g. `429 Too Many Requests`) from the API layer (think in stacks!).

3. With `params` we define the queries with want to append to the URL. Functionally, you can imagine that the list of key-value pairs what we define in list object `params` are appended after the string we set with `url` and a `?` (for example, in `https://example.com/over/there?name=ferret` the query is defined by the key-value `name=ferret`).

Now we can add these as attributes to the function `GET` and collect the response in `res`.

```
res <-
  httr::GET(url,
            httr::add_headers(.headers = headers),
            query = params)
```

By printing `res` we see details about the HTTP response (but not yet the API response or the content returned from the API).

```
print(res)
```

```
## Response [https://api.twitter.com/2/tweets/counts/recent?query=from%3ATwitterDev&granularity=day]
##   Date: 2022-03-03 03:12
##   Status: 200
##   Content-Type: application/json; charset=utf-8
##   Size: 729 B
```

If our request was authorised we should get

> Status: 200

if our request was not authorised (likely because your token was not correctly specified) we should instead get

> Status: 401

Assuming, that we got an OK from the HTTP layer, then we can access the content we receive as a response from the API layer. We access it with the function `httr::content()`.

```
obj.json <-
  httr::content(res, as = "text")
```

Now by default the Twitter API responses are in JSON format. We can use the jsonlite package to translate the JSON-formatted string into an R object with

```
obj.r <-
  jsonlite::fromJSON(obj.json)
```

```
print(obj.r)
```

```
## $data
##                        end                    start tweet_count
## 1 2022-02-25T00:00:00.000Z 2022-02-24T03:11:54.000Z           0
## 2 2022-02-26T00:00:00.000Z 2022-02-25T00:00:00.000Z           0
## 3 2022-02-27T00:00:00.000Z 2022-02-26T00:00:00.000Z           0
## 4 2022-02-28T00:00:00.000Z 2022-02-27T00:00:00.000Z           0
## 5 2022-03-01T00:00:00.000Z 2022-02-28T00:00:00.000Z           1
## 6 2022-03-02T00:00:00.000Z 2022-03-01T00:00:00.000Z           0
## 7 2022-03-03T00:00:00.000Z 2022-03-02T00:00:00.000Z           0
## 8 2022-03-03T03:11:54.000Z 2022-03-03T00:00:00.000Z           0
##
## $meta
## $meta$total_tweet_count
## [1] 1
```

And this is information on the number of tweets posted by **?** in the days before our request.