

API

02-20-2020

```
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.0 --

## v ggplot2 3.2.1      v purrr  0.3.3
## v tibble  2.1.3      v dplyr  0.8.4
## v tidyr   1.0.2      v stringr 1.4.0
## v readr   1.3.1      v forcats 0.4.0

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

```
library(jsonlite)

##
## Attaching package: 'jsonlite'

## The following object is masked from 'package:purrr':
##
##     flatten
```

API

This section lists some examples of public HTTP APIs that publish data in JSON format. These are great to get a sense of the complex structures that are encountered in real world JSON data.

See also <https://github.com/public-apis/public-apis> for a list of public APIs.

CitiBike NYC

A single public API that shows location, status and current availability for all stations in the New York City bike sharing initiative. <https://www.citibikenyc.com/system-data>

```
citibike <- fromJSON("https://gbfs.citibikenyc.com/gbfs/en/station_status.json")
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##      date
```

```
as_datetime(citibike$last_updated)
```

```
## [1] "2020-02-20 08:36:48 UTC"
```

```
stations <- citibike$data$stations
colnames(stations)
```

```
## [1] "station_id"          "num_bikes_available"
## [3] "num_ebikes_available" "num_bikes_disabled"
## [5] "num_docks_available"  "num_docks_disabled"
## [7] "is_installed"         "is_renting"
## [9] "is_returning"         "last_reported"
## [11] "eightd_has_available_keys"
```

```
nrow(stations)
```

```
## [1] 935
```

OnWater <https://onwater.io/>

```
# davis
url <- str_glue("https://api.onwater.io/api/v1/results/{lat},{long}", lat = 38.54491, long = -121.74052)
fromJSON(url)
```

```
## $query
## [1] "38.54491,-121.74052"
##
## $request_id
## [1] "20aaba6-6abc-4ec2-a430-48990e2ff35c"
##
## $lat
## [1] 38.54418
##
## $lon
## [1] -121.7398
##
## $water
## [1] FALSE
```

```
# lake tahoe
url <- str_glue("https://api.onwater.io/api/v1/results/{lat},{long}", lat = 39.0968, long = -120.0324)
fromJSON(url)
```

```
## $query
## [1] "39.0968,-120.0324"
```

```
##
## $request_id
## [1] "c01e0ed5-f9b5-4dbe-ade3-a621f8f71a27"
##
## $lat
## [1] 39.0968
##
## $lon
## [1] -120.0324
##
## $water
## [1] TRUE
```

Deck of Cards <http://deckofcardsapi.com/>

It is a very simple API which shuffles cards.

```
# get a deck
deck <- fromJSON("https://deckofcardsapi.com/api/deck/new/shuffle/?deck_count=1")
deck_id <- deck$deck_id

# draw two cards
cards <- fromJSON(
  str_glue("https://deckofcardsapi.com/api/deck/{deck_id}/draw/?count={count}",
    deck_id = deck$deck_id, count = 2),
  flatten = TRUE)

if (!identical(knitr::pandoc_to(), "latex")) {
  # don't display the cards in pdf
  knitr::include_graphics(cards$cards$images.svg)
}
```

The parameters after ? are called GET parameters. A more formal way to handle GET parameters is to use the `httr` package.

```
library(httr)
```

```
endpoint <- str_glue("https://deckofcardsapi.com/api/deck/{deck_id}/draw/", deck_id = deck$deck_id)
r <- GET(endpoint, query = list(count = 3))
json <- content(r, as = "text")
```

```
## No encoding supplied: defaulting to UTF-8.
```

```
cards <- fromJSON(json)
cards
```

```
## $deck_id
## [1] "tsdt381pthdn"
##
## $remaining
## [1] 47
```

```
##
## $cards
##      suit value code                images.svg
## 1 DIAMONDS      3   3D https://deckofcardsapi.com/static/img/3D.svg
## 2   CLUBS      7   7C https://deckofcardsapi.com/static/img/7C.svg
## 3   CLUBS KING   KC https://deckofcardsapi.com/static/img/KC.svg
##                        images.png
## 1 https://deckofcardsapi.com/static/img/3D.png
## 2 https://deckofcardsapi.com/static/img/7C.png
## 3 https://deckofcardsapi.com/static/img/KC.png
##                        image
## 1 https://deckofcardsapi.com/static/img/3D.png
## 2 https://deckofcardsapi.com/static/img/7C.png
## 3 https://deckofcardsapi.com/static/img/KC.png
##
## $success
## [1] TRUE
```

GeoDataSource <https://www.geodatasource.com/>

In this section, we are going to show you how we use an API which requires an API key. API key allows you to use the services the API provides on behalf of yourself.

```
r <- GET(
  "https://api.geodatasource.com/cities",
  query=list(
    key = "THE API KEY COPIED FROM THE WEBSITE",
    lat = 38.5449,
    lng = -121.741))

stop_for_status(r)
json <- content(r, as = "text")
fromJSON(json)
```

There are multiple ways to protect your API key.

- Create a file called `.Renviron` and put your API key into it.

```
GEODATA_KEY="YOUR API KEY"
```

```
# you might need to change your working directory and restart R session to make it work
r <- GET(
  "https://api.geodatasource.com/cities",
  query=list(
    key = Sys.getenv("GEODATA_KEY"),
    lat = 38.5449,
    lng = -121.741))
stop_for_status(r)
json <- content(r, as = "text")
fromJSON(json)
```

```
##      country      region                city latitude longitude
```

## 1	US California	Davis Mobile Estates	38.5422	-121.738
## 2	US California	Davis	38.5449	-121.741
## 3	US California	Dixon	38.4455	-121.823
## 4	US California	El Macero	38.5468	-121.694
## 5	US California	Merritt	38.6141	-121.761
## 6	US California	Plainfield	38.5907	-121.797
## 7	US California	Rancho Yolo Mobile Home Park	38.5522	-121.724
## 8	US California	Royal Oak Manufactured Home Community	38.5447	-121.73
## 9	US California	Saxon	38.4666	-121.656
## 10	US California	Sucro	38.4696	-121.805
## 11	US California	Swingle	38.5582	-121.676
## 12	US California	Webster	38.5621	-121.655
## 13	US California	Briggston	38.5313	-121.749

- The second approach is to make use of the package **keyring**. (PS: this method doesn't work for shiny app)

```
# use keyring::key_set to set a password
# only need to do it once, you will be prompted for the API key
keyring::key_set("GEODATA_KEY")
```

```
r <- GET(
  "https://api.geodatasource.com/cities",
  query=list(
    key = keyring::key_get("GEODATA_KEY"),
    lat = 38.5449,
    lng = -121.741))
stop_for_status(r)
json <- content(r, as = "text")
fromJSON(json)
```

The Guardian News <https://open-platform.theguardian.com/>

```
search_guardian <- function(text, page = 1) {
  r <- GET(
    "https://content.guardianapis.com/search",
    query=list(
      `api-key` = Sys.getenv("GUARDIAN_KEY"),
      q = text,
      page = page))
  stop_for_status(r)
  json <- content(r, as = "text", encoding = "UTF-8")
  fromJSON(json)$response
}

response <- search_guardian("coronavirus")
```

```
# number of pages
response$pages
```

```
## [1] 68
```

```
response$results %>% select(webTitle, webPublicationDate)
```

```
##               webTitle      webPublicationDate
## 1      Coronavirus: the huge unknowns 2020-02-16T07:22:00Z
## 2              Where has coronavirus spread? 2020-01-26T17:15:01Z
## 3 The Observer view on coronavirus | Observer editorial 2020-02-16T07:00:23Z
## 4      Coronavirus: more than 3,000 Britons tested 2020-02-16T16:33:53Z
## 5 Coronavirus is ruining my happy memories | Stewart Lee 2020-02-16T10:00:26Z
## 6      What coronavirus precautions are you taking? 2020-02-11T11:13:23Z
## 7      Coronavirus: what is self-isolation? 2020-02-05T11:37:47Z
## 8      Coronavirus quarantine precautions around the world 2020-02-04T13:37:42Z
## 9      Thursday briefing: London coronavirus case confirmed 2020-02-13T06:30:51Z
## 10     Taiwan reports first death from coronavirus 2020-02-16T16:09:58Z
```

```
search_guardian("coronavirus", 2)$results %>% select(webTitle, webPublicationDate)
```

```
##               webTitle
## 1      China coronavirus: mayor of Wuhan admits mistakes
## 2 The Observer view on the coronavirus outbreak | Observer editorial
## 3      Businesses worldwide count cost of coronavirus outbreak
## 4      Stormzy postpones Asian tour due to coronavirus
## 5      Coronavirus shakes citizens' faith in Chinese government
## 6      Coronavirus: Brazil evacuates 34 nationals from Wuhan
## 7      Taiwan reports first death from coronavirus
## 8      Expert questions effectiveness of coronavirus airport screening
## 9      How coronavirus is affecting the global economy
## 10     Geneva motor show organisers brace for coronavirus disruption
##      webPublicationDate
## 1  2020-01-27T14:29:34Z
## 2  2020-01-26T06:00:15Z
## 3  2020-02-13T18:49:34Z
## 4  2020-02-13T13:39:36Z
## 5  2020-01-24T18:03:16Z
## 6  2020-02-08T17:53:54Z
## 7  2020-02-16T16:09:58Z
## 8  2020-01-18T19:20:36Z
## 9  2020-02-05T13:49:00Z
## 10 2020-02-13T19:07:01Z
```

Bearer token

Some APIs such as yelp uses Bearer token instead of query string.

First, you will need to register an app on yelp: <https://www.yelp.com/developers>

```
r <- GET(
  "https://api.yelp.com/v3/businesses/search",
  add_headers(Authorization = paste("Bearer", Sys.getenv("YELP_TOKEN"))),
  query = list(
```

```

    location = "Davis"
  )
)
stop_for_status(r)
json <- content(r, as = "text")

```

No encoding supplied: defaulting to UTF-8.

```
fromJSON(json)$businesses %>% select(name)
```

```
##                               name
## 1          Sam's Mediterranean Cuisine
## 2                Burgers and Brew
## 3                Dutch Bros Coffee
## 4  Four Seasons Gourmet Chinese Restaurant
## 5                Taqueria Davis
## 6                Nugget Markets
## 7                Zumapoke & Lush Ice
## 8  Mikuni Japanese Restaurant and Sushi Bar
## 9                Sweet and Shavery
## 10               Taqueria Guadalajara
## 11               Woodstock's Pizza Davis
## 12               Crepeville
## 13               Blaze Fast-Fire'd Pizza
## 14               Temple Coffee Roasters
## 15               Thai Canteen
## 16               De Vere's Irish Pub
## 17               Tommy J's Grill & Catering
## 18               Raja's Tandoor
## 19               Tea List
## 20               In-N-Out Burger

```

OAUTH

Some APIs such as twitter, facebook, google, require using OAUTH to authenticate.

OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords

In our case, the “application” would be the R code.

I use google as an demo. First, you need to setup an app at <https://console.developers.google.com/> For gmail services, you also need to enable the GMail api for your app.

```

myapp <- oauth_app(
  "google",
  key = "929233483196-o0ge3pc7q3ec4gbe51ph21rg5tuucbbh.apps.googleusercontent.com",
  secret = Sys.getenv("GOOGLE_SECRET")
)

```

```

google_token <- oauth2.0_token(
  oauth_endpoints("google"),
  myapp,
  scope = c("profile", "email",
    "https://www.googleapis.com/auth/gmail.readonly"))

google_request <- function(endpoint, query = NULL) {
  r <- GET(endpoint, config(token = google_token), query = query)
  stop_for_status(r)
  json <- content(r, as = "text")
  fromJSON(json)
}

```

```

# search mailbox for GeoDataSource
google_request("https://www.googleapis.com/gmail/v1/users/me/messages",
  query = list(q = "GeoDataSource"))

```

```

## $messages
##           id           threadId
## 1 17060c703d2c617b 17060c703d2c617b
## 2 17060c703052bd61 17060c703052bd61
##
## $resultSizeEstimate
## [1] 2

```

```

# Get the title of a specific mail
email <- google_request(
  str_glue("https://www.googleapis.com/gmail/v1/users/me/messages/{thread}", thread = "17060c703052bd61")
email$payload$headers %>% filter(name == "Subject") %>% select(value)

```

```

##                               value
## 1 GeoDataSource(TM) License Information

```

Remark: if you just want to do google search, use this API <https://serpapi.com/>