# API

02-20-2020

```
library(tidyverse)
```

```
## -- Attaching packages ------------

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- 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 imitative. 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] "2020-02-22 00:57:26 UTC"
```

```
stations <- citibike$data$stations
stations %>%
  filter(num_bikes_available > 0)
```

```
##   station_id num_bikes_available num_ebikes_available num_bikes_disabled
## 1         72                 49                    0                  0
## 2         79                 30                    0                  0
## 3         82                 19                    0                  0
## 4         83                 55                    1                  1
## 5        116                 22                    1                  0
## 6        119                 15                    0                  0
## 7        120                 10                    0                  0
## 8        127                 16                    0                  1
## 9        128                 25                    0                  0
##   num_docks_available num_docks_disabled is_installed is_renting is_returning
## 1                   6                  0            1          1            1
## 2                   3                  0            1          1            1
## 3                   8                  0            1          1            1
## 4                   6                  0            1          1            1
## 5                  28                  0            1          1            1
## 6                   4                  0            1          1            1
## 7                   9                  0            1          1            1
## 8                  14                  0            1          1            1
## 9                   4                  1            1          1            1
##   last_reported eightd_has_available_keys
## 1    1582332160                    FALSE
## 2    1582332497                    FALSE
## 3    1582331941                    FALSE
## 4    1582331768                    FALSE
## 5    1582332941                    FALSE
## 6    1582332682                    FALSE
## 7    1582332389                    FALSE
## 8    1582332874                    FALSE
## 9    1582333024                    FALSE
##  [ reached 'max' / getOption("max.print") -- omitted 866 rows ]
```

```
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"
```

```r
nrow(stations)
```

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

### OnWater https://onwater.io/

```r
# 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] "20aabaa6-6abc-4ec2-a430-48990e2ff35c"
##
## $lat
## [1] 38.54418
##
## $lon
## [1] -121.7398
##
## $water
## [1] FALSE
```

```r
# 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 suffles cards.

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

```r
library(httr)
```

```r
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.
```

```r
cards <- fromJSON(json, flatten = TRUE)
cards
```

```
## $deck_id
## [1] "4mxfg2e42dau"
##
## $remaining
## [1] 47
##
## $cards
##       suit value code                                          image
## 1 DIAMONDS  KING   KD https://deckofcardsapi.com/static/img/KD.png
## 2   SPADES     5   5S https://deckofcardsapi.com/static/img/5S.png
## 3   SPADES     4   4S https://deckofcardsapi.com/static/img/4S.png
##                                     images.svg
## 1 https://deckofcardsapi.com/static/img/KD.svg
## 2 https://deckofcardsapi.com/static/img/5S.svg
## 3 https://deckofcardsapi.com/static/img/4S.svg
##                                     images.png
## 1 https://deckofcardsapi.com/static/img/KD.png
## 2 https://deckofcardsapi.com/static/img/5S.png
## 3 https://deckofcardsapi.com/static/img/4S.png
##
## $success
## [1] TRUE
```

### GeoDataSource https://www.geodatasource.com/

In this secton, 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 = "YOUR PRIVATE API KEY",
    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. We might want to use `usethis::edit_r_environ("project")` to create and edit the file directly.

  ```
  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 appoarch 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] 71
```

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

```
##                                                    webTitle
## 1                           Coronavirus: the huge unknowns
## 2                            Where has coronavirus spread?
## 3      The Observer view on coronavirus | Observer editorial
## 4                       Coronavirus: what is self-isolation?
```

```
## 5                Taiwan reports first death from coronavirus
## 6  Worthing hospital healthcare worker contracts coronavirus
## 7               Coronavirus: more than 3,000 Britons tested
## 8    Coronavirus is ruining my happy memories | Stewart Lee
## 9      Thursday briefing: London coronavirus case confirmed
## 10            What coronavirus precautions are you taking?
##      webPublicationDate
## 1  2020-02-16T07:22:00Z
## 2  2020-01-26T17:15:01Z
## 3  2020-02-16T07:00:23Z
## 4  2020-02-05T11:37:47Z
## 5  2020-02-16T16:09:58Z
## 6  2020-02-11T18:55:20Z
## 7  2020-02-16T16:33:53Z
## 8  2020-02-16T10:00:26Z
## 9  2020-02-13T06:30:51Z
## 10 2020-02-11T11:13:23Z
```

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

```
##                                                        webTitle
## 1                      Taiwan reports first death from coronavirus
## 2           Businesses worldwide count cost of coronavirus outbreak
## 3                    Stormzy postpones Asian tour due to coronavirus
## 4          Worthing hospital healthcare worker contracts coronavirus
## 5                  China coronavirus: mayor of Wuhan admits mistakes
## 6  The Observer view on the coronavirus outbreak | Observer editorial
## 7              Coronavirus: Brazil evacuates 34 nationals from Wuhan
## 8           Coronavirus shakes citizens' faith in Chinese government
## 9                    How coronavirus is affecting the global economy
## 10                   Who is most at risk of contracting coronavirus?
##      webPublicationDate
## 1  2020-02-16T16:09:58Z
## 2  2020-02-13T18:49:34Z
## 3  2020-02-13T13:39:36Z
## 4  2020-02-11T18:55:20Z
## 5  2020-01-27T14:29:34Z
## 6  2020-01-26T06:00:15Z
## 7  2020-02-08T17:53:54Z
## 8  2020-01-24T18:03:16Z
## 9  2020-02-05T13:49:00Z
## 10 2020-02-21T13:47:11Z
```

## Yelp

Some APIs such as yelp provides 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                  Blaze Fast-Fire'd Pizza
## 13                               Crepeville
## 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
```

## Noun Project https://thenounproject.com/

The Noun Project uses one-legged OAuth 1.0 protocol to authenticate users. In OAuth protocal, there are two important pieces of strings

- Client key
- Client key secret

```
nouns_app <- oauth_app(
  "nounproject",
  key = "ed652bdcd50a4496bbc2253a603b9e9b",
  secret = Sys.getenv("NOUN_SECRET")
)

get_nouns_api <- function(endpoint) {
  signature <- oauth_signature(endpoint, app = nouns_app)
  GET(endpoint, oauth_header(signature))
```

```
}

r <- get_nouns_api(
  str_glue("https://api.thenounproject.com/icons/{term}", term = "statistics"))

stop_for_status(r)
json <- content(r, as = "text", encoding = "UTF-8")

icons <- fromJSON(json)$icons %>% pull(preview_url)
if (!identical(knitr:::pandoc_to(), "latex")) {
  # don't display the cards in pdf
  knitr::include_graphics(icons[1:10])
}
```

## Twitter

First, create an app at https://developer.twitter.com/. You will need to register a twitter developer account first.

There are two authentication methods for Twitter.

- OAuth 1.0

  Twitter's Oauth 1.0 allows an app to access private account information or perform a Twitter action on behalf of a Twitter account.

- OAuth 2.0

  Twitter's OAuth 2.0 only allows an app to access information publicly available on Twitter.

PS: These are Twitter's specific differences between Oauth 1.0 and 2.0. In general, both OAuth 1.0 and 2.0 can perform either two-legged and three-legged authentication.

*Oauth 1.0* (one-legged, though some people called it two-legged)

```
twitter_app <- oauth_app("twitter",
  key = "1vqbnsftUcNLucoVxQiWYnD2d",
  secret = Sys.getenv("TWITTER_SECRET")
)

get_twitter_api <- function(endpoint, query = NULL) {
  signature <- oauth_signature(
    endpoint,
    app = twitter_app,
    token = "131203353-732JhLQdj519ILTQrraFXpU6bR7cMvb8LZzLxNSC",
    token_secret = Sys.getenv("TWITTER_TOKEN_SECRET")
  )
  GET(endpoint, oauth_header(signature), query = query)
}

# read my timeline
r <- get_twitter_api("https://api.twitter.com/1.1/statuses/home_timeline.json")
stop_for_status(r)
json <- content(r, as = "text")
fromJSON(json)
```

```
##                       created_at           id             id_str
## 1 Sat Feb 22 22:48:08 +0000 2014 4.373582e+17 437358172983279616
##
## 1                                                    WhatsApp service has been restored. We are so
##   truncated entities.hashtags entities.symbols entities.user_mentions
## 1     FALSE              NULL             NULL                   NULL
##                                                                      entities.urls
## 1                                                                             NULL
##                                                                  source
## 1 <a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
##   in_reply_to_status_id in_reply_to_status_id_str in_reply_to_user_id
## 1                    NA                        NA                  NA
##   in_reply_to_user_id_str in_reply_to_screen_name   user.id user.id_str
## 1                      NA                      NA 114894699   114894699
##         user.name user.screen_name user.location
## 1 WhatsApp Status        wa_status    California
##                                                 user.description
## 1 We are working very hard to make this twitter account irrelevant.
##                 user.url
## 1 http://t.co/9UmbbIowwT
##                                                  user.entities.urls
## 1 http://t.co/9UmbbIowwT, http://www.whatsapp.com, whatsapp.com, 0, 22
##   user.entities.urls user.protected user.followers_count user.friends_count
## 1               NULL          FALSE              2013307                 1
##   user.listed_count                user.created_at user.favourites_count
## 1              3401 Tue Feb 16 23:35:05 +0000 2010                     0
##   user.utc_offset user.time_zone user.geo_enabled user.verified
## 1              NA             NA            FALSE          TRUE
##   user.statuses_count user.lang user.contributors_enabled user.is_translator
## 1                  60        NA                     FALSE              FALSE
##   user.is_translation_enabled user.profile_background_color
## 1                       FALSE                         C0DEED
##                  user.profile_background_image_url
## 1 http://abs.twimg.com/images/themes/theme1/bg.png
##            user.profile_background_image_url_https
## 1 https://abs.twimg.com/images/themes/theme1/bg.png
##   user.profile_background_tile
## 1                        FALSE
##                                                    user.profile_image_url
## 1 http://pbs.twimg.com/profile_images/2489640725/iyn161c0ptsxt7indbrf_normal.png
##                                                  user.profile_image_url_https
## 1 https://pbs.twimg.com/profile_images/2489640725/iyn161c0ptsxt7indbrf_normal.png
##   user.profile_link_color user.profile_sidebar_border_color
## 1                  1DA1F2                         C0DEED
##   user.profile_sidebar_fill_color user.profile_text_color
## 1                          DDEEF6                 333333
##   user.profile_use_background_image user.has_extended_profile
## 1                              TRUE                     FALSE
##   user.default_profile user.default_profile_image user.following
## 1                 TRUE                      FALSE          TRUE
##   user.follow_request_sent user.notifications user.translator_type geo
## 1                    FALSE             FALSE                 none  NA
##   coordinates place contributors is_quote_status retweet_count favorite_count
## 1          NA    NA           NA           FALSE         18752          30614
```

```
##   favorited retweeted lang possibly_sensitive possibly_sensitive_appealable
## 1     FALSE     FALSE   en                 NA                              NA
##  [ reached getOption("max.print") -- omitted 3 rows ]
##  [ reached 'max' / getOption("max.print") -- omitted 16 rows ]
```

*Oauth 1.0* (three-legged)

```
twitter_app <- oauth_app("twitter",
  key = "1vqbnsftUcNLucoVxQiWYnD2d",
  secret = Sys.getenv("TWITTER_SECRET")
)

twitter_token <- oauth1.0_token(
  oauth_endpoints("twitter"),
  twitter_app
)

# read my timeline
r <- GET(
  "https://api.twitter.com/1.1/statuses/home_timeline.json",
  config(token = twitter_token)
)

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

```
# post a twitter
r <- POST(
  "https://api.twitter.com/1.1/statuses/update.json",
  config(token = twitter_token),
  query = list(status = "I posted a tweet from R using httr")
)
stop_for_status(r)
```

*Oauth 2.0* (client credentials, aka, two-legged)

```
twitter_app <- oauth_app("twitter",
  key = "1vqbnsftUcNLucoVxQiWYnD2d",
  secret = Sys.getenv("TWITTER_SECRET")
)

twitter_token <- oauth2.0_token(
  oauth_endpoint(
    authorize = NULL,
    access = "https://api.twitter.com/oauth2/token"
  ),
  twitter_app,
  client_credentials = TRUE
)
```

```
# Where On Earth IDentifier
get_woeid <- function(city, country) {
  r <- GET(
    "https://api.twitter.com/1.1/trends/available.json",
    config(token = twitter_token)
  )

  stop_for_status(r)
  json <- content(r, as = "text")
  fromJSON(json) %>%
    filter(name == {{ city }}, country == {{ country }}) %>%
    pull(woeid)
}

get_trends <- function(woeid) {
  r <- GET(
    "https://api.twitter.com/1.1/trends/place.json",
    config(token = twitter_token),
    query = list(id = woeid)
  )

  stop_for_status(r)
  json <- content(r, as = "text")
  fromJSON(json)$trends[[1]]
}

woeid <- get_woeid("Sacramento", "United States")
get_trends(woeid) %>% select(name)
```

```
##                          name
## 1                   #EDCLV2020
## 2                         Girl
## 3                   California
## 4                    Bloomberg
## 5                       People
## 6                       Boogie
## 7         #RussianInterference
## 8                       Morris
## 9                      Cousins
## 10                Troy Daniels
## 11             #FriendsReunion
## 12                 #removeSBMM
## 13      #FreeCodeFridayContest
## 14                      Dudley
## 15                    Demarcus
## 16                   Dan Didio
## 17                    Markieff
## 18                   Paul Ryan
## 19                 Jake Tapper
## 20            Jeremy Christian
## 21               All The Smoke
## 22               Magnitsky Act
## 23                      He's 9
```

```
## 24                   A.P. Indy
## 25                      3 NDAs
## 26                  Russiagate
## 27                John Tonelli
## 28                     Marlene
## 29                     Pelinka
## 30                      Alshon
## 31              Taylor Gabriel
## 32                  Classified
## 33                        Sady
## 34           #FireJasonJohnson
## 35       #TrumpIsARussianAsset
## 36            #FreeStuffFriday
## 37            #filterchallenge
## 38                   #DoesToMe
## 39           #StreamingPartyON
## 40  #StephenMillerWeddingGifts
## 41            #FightForWynonna
## 42                #mnwildfirst
## 43              #WildervsFury2
## 44                    #teamknj
## 45                   #NYRvsCAR
## 46          #ConspiracyPalette
## 47                #Bellator239
## 48       #MarathonDRWFirstGoal
## 49 #AddAvocadoToAnyMovieorShow
## 50           #BeardedButtigieg
```

PS: There is `rtweet` package, no one, in practice, will directly work with twitter API.

## Google

First, you need to setup an app at https://console.developers.google.com/. Additionally, you also need to enable the gmail api if you want the manage gmail.

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

google_token <- oauth2.0_token(
  oauth_endpoints("google"),
  gooogle_app,
  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 1: if you just want to manage gmail in R, see gmailr https://gmailr.r-lib.org/
- Remark 2: if you just want to do google search, see https://serpapi.com/
- Remark 3: if you want to use google API, see gargle https://gargle.r-lib.org/
- Remark 4: if you want to use google authentication in your shiny app, see googleAuthR https://code.markedmondson.me/googleAuthR/

## Exisiting packages

You might not have to interact with the APIs directly.