API

02-20-2020

```r
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   0.8.3
## v tidyr   1.0.0     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()
```

```r
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

```r
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
```

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

```
## [1] "2020-02-22 09:00:34 UTC"
```

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

```
##   station_id num_bikes_available num_ebikes_available num_bikes_disabled
## 1         72                  52                    0                  0
## 2         79                  30                    0                  0
## 3         82                  22                    0                  0
## 4         83                  54                    0                  0
## 5        116                  26                    0                  0
## 6        119                  13                    0                  0
## 7        120                   8                    0                  0
## 8        127                  21                    0                  0
## 9        128                  24                    0                  0
##   num_docks_available num_docks_disabled is_installed is_renting is_returning
## 1                   3                  0            1          1            1
## 2                   3                  0            1          1            1
## 3                   5                  0            1          1            1
## 4                   8                  0            1          1            1
## 5                  24                  0            1          1            1
## 6                   6                  0            1          1            1
## 7                  11                  0            1          1            1
## 8                  10                  0            1          1            1
## 9                   5                  1            1          1            1
##   last_reported eightd_has_available_keys
## 1    1582361107                     FALSE
## 2    1582356297                     FALSE
## 3    1582353858                     FALSE
## 4    1582356644                     FALSE
## 5    1582361304                     FALSE
## 6    1582353879                     FALSE
## 7    1582352556                     FALSE
## 8    1582357196                     FALSE
## 9    1582358154                     FALSE
##  [ reached 'max' / getOption("max.print") -- omitted 855 rows ]
```

```r
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] "20aabaa6-6abc-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 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] "uwupmzu0tgdl"
##
## $remaining
## [1] 47
##
## $cards
##       suit value code                                                image
## 1 DIAMONDS   ACE   AD https://deckofcardsapi.com/static/img/aceDiamonds.png
## 2   HEARTS     2   2H          https://deckofcardsapi.com/static/img/2H.png
## 3 DIAMONDS     5   5D          https://deckofcardsapi.com/static/img/5D.png
##                                   images.svg
## 1 https://deckofcardsapi.com/static/img/AD.svg
## 2 https://deckofcardsapi.com/static/img/2H.svg
## 3 https://deckofcardsapi.com/static/img/5D.svg
##                                   images.png
## 1 https://deckofcardsapi.com/static/img/AD.png
## 2 https://deckofcardsapi.com/static/img/2H.png
## 3 https://deckofcardsapi.com/static/img/5D.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
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"
  ```

```r
# 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)

```r
# 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
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/

```r
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")
```

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

```
## [1] 71
```

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

```
##                                                   webTitle   webPublicationDate
## 1                           Coronavirus: the huge unknowns 2020-02-16T07:22:00Z
## 2    Thursday briefing: London coronavirus case confirmed 2020-02-13T06:30:51Z
## 3                            Where has coronavirus spread? 2020-01-26T17:15:01Z
## 4                   How to protect yourself from coronavirus 2020-02-03T11:06:28Z
## 5             Coronavirus: more than 3,000 Britons tested 2020-02-16T16:33:53Z
## 6  Coronavirus is ruining my happy memories | Stewart Lee 2020-02-16T10:00:26Z
## 7            What coronavirus precautions are you taking? 2020-02-11T11:13:23Z
## 8   The Observer view on coronavirus | Observer editorial 2020-02-16T07:00:23Z
## 9     Coronavirus quarantine precautions around the world 2020-02-04T13:37:42Z
## 10                  Coronavirus: what is self-isolation? 2020-02-05T11:37:47Z
```

```
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
```

Google map

You will need to register a free (one-year) google clould platofmr account first. THen following the instruction here to generate an api key. https://developers.google.com/places/web-service/get-api-key

```
r <- GET(
  "https://maps.googleapis.com/maps/api/place/nearbysearch/json",
  query = list(
    key = Sys.getenv("GOOGLE_API_KEY"),
    location = "38.5449,-121.741",
    radius = 500,
    types = "food",
    name = "in-n-out"
  )
)
stop_for_status(r)
json <- content(r, as = "text", encoding = "UTF-8")
fromJSON(json, flatten = TRUE)$results %>% pull(vicinity)
```

```
## [1] "1020 Olive Dr, Davis"
```

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
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.
```

```r
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.

Twitter's OAuth 2.0 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 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",
```

```r
    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                          Girl
## 2                        Bernie
## 3           #AfterTheSexIsOver
## 4              #TurnThePartyON
## 5             #FILTERCHALLENGE
## 6             #TheMauldalorian
## 7            #SaturdayThoughts
## 8                  Josh Jackson
## 9                    Costa Mesa
## 10                    Josh Hart
## 11                Avery Bradley
## 12     Democratic Establishment
## 13                       Bellas
## 14                  Lake Lanier
## 15                 The Pelicans
## 16                   CLASSIFIED
## 17              Johnny Football
## 18                     Carushow
## 19                       AJ Lee
## 20                 Daniel Theis
## 21                     Hamsters
## 22                  Molly Holly
## 23                    Zulu Ball
## 24                        Xolos
## 25                  Congrats Jim
## 26             Hassan Whiteside
## 27                Malik Beasley
## 28                         Melli
## 29     #AvatarTheLastAirbender
## 30               #clubtwitter
## 31 #INYOURAREA_WORLDTOURFinale
## 32            #loveafterlockup
## 33           #SofaKingAnything
## 34            #MonbebeGotWonho
## 35                 #ismchubble
## 36                   #memvslal
## 37              #GoAllOutForX1
## 38                    #WWEHOF
## 39  #MyInnerBeastWillSlaughter
## 40               #90sBookTitle
## 41                 #labynight
## 42                   #RIZIN21
## 43               #DickVanDyke
## 44                  #NOPvsPOR
## 45                      #WAAF
## 46             #BDMeritWithMEW
## 47               #ZuluBall2020
## 48                #RemoveSSBM
## 49               #VAVinDallas
## 50               #BroeWedding
```

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