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   0.8.4
## 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()
```

```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
```

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

```
## [1] "2020-02-27 06:07:56 UTC"
```

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

```
##   station_id num_bikes_available num_ebikes_available num_bikes_disabled
## 1        304                   5                    0                  4
## 2        359                  36                    1                  6
## 3       3255                  10                    0                  0
## 4         72                  53                    0                  0
## 5         79                  23                    0                  1
## 6         82                  25                    0                  2
## 7         83                  52                    0                  1
## 8        116                  28                    1                  3
##   num_docks_available num_docks_disabled is_installed is_renting is_returning
## 1                  24                  0            1          1            0
## 2                  22                  0            1          1            0
## 3                   9                  0            1          1            0
## 4                   2                  0            1          1            1
## 5                   9                  0            1          1            1
## 6                   0                  0            1          1            1
## 7                   9                  0            1          1            1
## 8                  19                  0            1          1            1
##   last_reported eightd_has_available_keys       eightd_active_station_services
## 1    1582779956                      TRUE a58d9e34-2f28-40eb-b4a6-c8c01375657a
## 2    1582782335                     FALSE 2e104e31-606a-44af-8b25-ceaffc338489
## 3    1582779938                     FALSE 9fb74cf0-b08b-4983-ae0e-be909fc28bc3
## 4    1582774496                     FALSE                                 NULL
## 5    1582777731                     FALSE                                 NULL
## 6    1582778056                     FALSE                                 NULL
## 7    1582779068                     FALSE                                 NULL
## 8    1582782094                     FALSE                                 NULL
##  [ reached 'max' / getOption("max.print") -- omitted 836 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"      "eightd_active_station_services"
```

```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] "zxm2b7ikh16z"
##
## $remaining
## [1] 47
##
## $cards
##       suit value code                                        image
## 1   SPADES     8   8S https://deckofcardsapi.com/static/img/8S.png
## 2 DIAMONDS QUEEN   QD https://deckofcardsapi.com/static/img/QD.png
## 3    CLUBS  KING   KC https://deckofcardsapi.com/static/img/KC.png
##                                    images.svg
## 1 https://deckofcardsapi.com/static/img/8S.svg
## 2 https://deckofcardsapi.com/static/img/QD.svg
## 3 https://deckofcardsapi.com/static/img/KC.svg
##                                    images.png
## 1 https://deckofcardsapi.com/static/img/8S.png
## 2 https://deckofcardsapi.com/static/img/QD.png
## 3 https://deckofcardsapi.com/static/img/KC.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)
```

How to store your secrets

There are multiple ways to protect your API key.

- Make use of environment variables. Environment variables are stored in `.Renviron`. You could put this file in various places.
  - HOME directory
    `usethis::edit_r_environ()`
  - Project home directory
    `usethis::edit_r_environ("project")`
  - Under the same directory as the Rscript
    Create a file called `.Renviron` and put your API key into it.

```
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 other appeach 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] 86
```

```r
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: what is self-isolation? 2020-02-05T11:37:47Z
## 5          Economic impact of coronavirus outbreak deepens 2020-02-23T17:57:53Z
## 6  Coronavirus: China postpones National People's Congress 2020-02-24T14:31:47Z
## 7                    How to protect yourself from coronavirus 2020-02-25T16:10:43Z
## 8                 Coronavirus: more than 3,000 Britons tested 2020-02-16T16:33:53Z
## 9   Coronavirus is ruining my happy memories | Stewart Lee 2020-02-16T10:00:26Z
## 10               Taiwan reports first death from coronavirus 2020-02-16T16:09:58Z
```

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

```
##                                                           webTitle
## 1                    Iran's deputy health minister: I have coronavirus
## 2                          Coronavirus: more than 3,000 Britons tested
## 3                   Coronavirus is ruining my happy memories | Stewart Lee
## 4                          What coronavirus precautions are you taking?
## 5                       Coronavirus quarantine precautions around the world
## 6                  Tenerife coronavirus: 1,000 guests at hotel quarantined
## 7                       Apple warns of coronavirus causing iPhone shortages
## 8                  Coronavirus: US evacuates Americans onboard cruise ship
## 9                    China coronavirus: mayor of Wuhan admits mistakes
## 10 The Observer view on the coronavirus outbreak | Observer editorial
##      webPublicationDate
## 1  2020-02-25T13:30:10Z
## 2  2020-02-16T16:33:53Z
## 3  2020-02-16T10:00:26Z
## 4  2020-02-11T11:13:23Z
## 5  2020-02-04T13:37:42Z
## 6  2020-02-25T15:38:07Z
## 7  2020-02-17T22:42:57Z
## 8  2020-02-16T23:57:39Z
## 9  2020-01-27T14:29:34Z
## 10 2020-01-26T06:00:15Z
```

Wolfram alpha

```r
r <- GET(
  "https://api.wolframalpha.com/v2/query",
  query = list(
    appid = Sys.getenv("WOLFRAM_ALPHA_KEY"),
    input = "integrate x^3",
    format = "plaintext",
    output = "json"
  )
)
stop_for_status(r)
json <- content(r, as = "text", encoding = "UTF-8")

if (!identical(knitr:::pandoc_to(), "latex")) {
  fromJSON(json, flatten = TRUE)$queryresult$pods %>%
    hoist(subpods, text = "plaintext") %>%
    select(title, text) %>%
    unnest(text)
}
```

Google map

You will need to register a google clould platfram account with $300 credit first. THen following the instruction here to generate an api key. https://developers.google.com/places/web-service/get-api-key

```r
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.
```

```
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")
```

```r
)

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 allows an app to access information publicly available on Twitter via two legged Oauth.

```r
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
)
```

```r
# 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                         Facts
## 2                          Girl
## 3                       America
## 4                         Pence
## 5             Gabriel Fernandez
## 6  #MyHeroAcademiaHeroesRising
## 7                #AEWDynamite
## 8          #adamkutnerpowerplay
## 9             #TheMaskedSinger
## 10           #HelloHanbinIsFree
## 11             Donovan Mitchell
## 12                       Pitino
## 13                       Kanter
## 14               Darryl Morsell
## 15                Solano County
## 16                  Mike Conley
## 17           tatum and mitchell
## 18               Robert Edwards
## 19                     The Jazz
## 20                 Public Enemy
## 21                       Tigres
## 22                       Mewtwo
## 23               Orange Cassidy
## 24                 Best Concert
## 25                Austin Rivers
## 26                      Alianza
## 27                #CNNTownHall
## 28                   #Survivor
## 29                       #RHONJ
## 30     #TrumpCouldBeAGoodGuyIf
## 31                       #Terps
## 32                  #ChicagoPD
## 33         #MarriedAtFirstSight
## 34                   #BOSvsUTA
## 35 #IWonderWhatItWouldBeLikeIf
## 36         #ItSeemsTheOlderIGet
## 37           #BTSStreamingParty
## 38          #CoronaVirusUpdates
## 39                #ChicagoFire
## 40              #AEWRevolution
## 41               #BlackInkCrew
## 42                 #BernieBruh
## 43         #fancamsareoverparty
## 44                 #my600lblife
## 45                  #Riverdale
## 46            #MilwaukeeStrong
## 47               #TheMagicians
## 48                #SistasOnBET
## 49                   #PITvsLAK
## 50                   #MDvsMINN
```

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