APIs for Social Science

Exercise 2

```r
library("usethis")
library("httr2")
library("jsonlite")
library("stringr")
library("devtools")
library("gtrendsR")
library("ggplot2")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library("RedditExtractoR")
library("osmdata")
```

```
## Data (c) OpenStreetMap contributors, ODbL 1.0. https://www.openstreetmap.org/copyright
```

```r
library("spotifyr")
library("censusapi")
```

```
##
## Attaching package: 'censusapi'

## The following object is masked from 'package:methods':
##
##     getFunction
```

Introduction

In this exercise, we will explore different APIs useful for social science research. A comprehensive overview of different APIs can be found here. If you encounter difficulties, feel free to refer to this resource. Please note that we won't provide all the details required to make these APIs work here; part of the challenge is to navigate through documentation and tutorials if you encounter issues during your exploration. When you work on your research project, we encourage you to proactively seek out new data sources and APIs.

Remember that the availability of specific APIs may change, such as Twitter's discontinuation of free access for academic research. Always review the developer agreement from the respective API provider to ensure compliance.

Authentication

API providers typically require user identification for API calls, commonly done through the use of API keys. However, it's considered poor practice to hardcode your API key directly into your code, especially if you intend to share your code with others. To safeguard your authentication information, it's recommended to store it in environment variables. This can be achieved by adding them to your `.Renviron` file (for added convenience, you can use the `usethis` package). If you're unsure about the process, refer to this for guidance.

```
usethis::edit_r_environ(scope = "user")
```

```
## * Edit '/Users/unaifischerabaigar/.Renviron'
## * Restart R for changes to take effect
```

Intro to APIs

The `httr2` package allows us to easily send API requests from within R using the `request()` and `req_perform()` function. As a basic example, retrieve the current location of the International Space Station via this url http://api.open-notify.org/iss-now.json. Examine the resulting object. Where can you check if the API call was successful?

... *# your work here*

```
url <- "http://api.open-notify.org/iss-now.json"
request <- request(url)
res <- request %>% req_perform()
res
```

```
## <httr2_response>

## GET http://api.open-notify.org/iss-now.json

## Status: 200 OK

## Content-Type: application/json

## Body: In memory (113 bytes)
```

The data in this case was returned as a json file. Use the `resp_body_json()` function to extract the data we're interested in.

... *# your work here*

```
resp_body_json(res)
```

```
## $message
## [1] "success"
##
## $iss_position
## $iss_position$latitude
## [1] "19.8390"
##
## $iss_position$longitude
## [1] "-10.2169"
##
##
## $timestamp
## [1] 1699221235
```

You can also provide additional parameters with your API request in form of a query. For example, lets use the open source weather API Bright Sky to retrieve the weather data from the DWD – Germany's meteorological service. Retrieve data on the current weather at specific weather station near Munich. Use the `req_url_query()` function to pass the wmo station id parameter (here: “10865”).

... # your work here

```r
url <- "https://api.brightsky.dev/current_weather"
request <- request(url) %>% req_url_query(wmo_station_id = "10865")
weather_data <- request %>% req_perform()
weather_data
```

```
## <httr2_response>

## GET https://api.brightsky.dev/current_weather?wmo_station_id=10865

## Status: 200 OK

## Content-Type: application/json

## Body: In memory (940 bytes)
```

Extract the data on the current temperature.

... # your work here

```r
weather_data_extracted <- weather_data %>% resp_body_json()
weather_data_extracted$weather$temperature
```

```
## [1] 9.3
```

NY Times Books API

Create a developer account at the NY Times (https://developer.nytimes.com/get-started) and create a new app to retrieve an API key and store in your R environment.

The Books API (https://developer.nytimes.com/docs/books-product/1/overview) allows for the retrieval of information on book reviews and best seller lists. The NY Times also offers a selection of other APIs (https://developer.nytimes.com/apis) - feel free to explore!

Make a request to the Books API to retrieve reviews for a specific books (of course the book needs to have been actually reviewed by the NY Times in the last years).

... # your work here

```r
url <- "https://api.nytimes.com/svc/books/v3"
request <- request(url) %>%
  req_url_path_append("/reviews.json") %>%
  req_url_query(`api-key` = Sys.getenv("NYT_KEY"), title = 'Circe')
book_data <- request %>% req_perform() %>% resp_body_json()
str(book_data)
```

```
## List of 4
##  $ status     : chr "OK"
##  $ copyright  : chr "Copyright (c) 2023 The New York Times Company.  All Rights Reserved."
##  $ num_results: int 1
##  $ results    :List of 1
##   ..$ :List of 9
##   .. ..$ url           : chr "https://www.nytimes.com/2018/05/28/books/review/circe-madeline-miller.
##   .. ..$ publication_dt: chr "2018-05-28"
##   .. ..$ byline        : chr "CLAIRE MESSUD"
##   .. ..$ book_title    : chr "Circe"
##   .. ..$ book_author   : chr "Madeline Miller"
##   .. ..$ summary       : chr "In Madeline Miller's latest adaptation of Greek myth, "Circe," we enco
##   .. ..$ uuid          : chr "00000000-0000-0000-0000-000000000000"
##   .. ..$ uri           : chr "nyt://book/00000000-0000-0000-0000-000000000000"
##   .. ..$ isbn13        :List of 1
##   .. .. ..$ : chr "9780316556347"
```

US Census API

The US Census Bureau's data APIs are comprehensive, offering over 1000 available endpoints. To utilize these APIs, you'll need an API key, which you can obtain at this link: https://api.census.gov/data/key_signup.html. Store it as "CENSUS_KEY" in your R enviornment.

Several R packages are available to facilitate interfacing with the Census API, and in this tutorial, we will use the 'censusapi' package to make our initial API calls. As we progress through the lecture, we will conduct more advanced analyses using Census data.

Check out the number of available APIs using the `listCensusApis()` function.

... # your work here

```r
apis <- listCensusApis()
head(apis)
```

```
##                                                              title          name
## 1 Current Population Survey Annual Social and Economic Supplement  cps/asec/mar
## 2                        Current Population Survey: Basic Monthly cps/basic/apr
## 3                        Current Population Survey: Basic Monthly cps/basic/aug
## 4                        Current Population Survey: Basic Monthly cps/basic/feb
## 5                        Current Population Survey: Basic Monthly cps/basic/jan
## 6                        Current Population Survey: Basic Monthly cps/basic/jul
##   vintage      type        temporal
## 1    2023 Microdata 2023-03/2023-03
## 2    2023 Microdata 2023-04/2023-04
## 3    2023 Microdata 2023-08/2023-08
## 4    2023 Microdata 2023-02/2023-02
## 5    2023 Microdata 2023-01/2023-01
## 6    2023 Microdata 2023-07/2023-07
##                                              url              modified
## 1  http://api.census.gov/data/2023/cps/asec/mar 2023-08-14 09:09:01.0
## 2 http://api.census.gov/data/2023/cps/basic/apr 2023-01-10 15:11:40.0
## 3 http://api.census.gov/data/2023/cps/basic/aug 2023-01-10 15:11:40.0
## 4 http://api.census.gov/data/2023/cps/basic/feb 2023-01-10 15:11:40.0
## 5 http://api.census.gov/data/2023/cps/basic/jan 2023-01-10 15:11:39.0
## 6 http://api.census.gov/data/2023/cps/basic/jul 2023-01-10 15:11:40.0
##
## 1 The Annual Social and Economic Supplement or March CPS supplement is the primary source of detaile
## 2
## 3
## 4
## 5
## 6
##              contact
## 1 dsd.cps@census.gov
## 2 dsd.cps@census.gov
## 3 dsd.cps@census.gov
## 4 dsd.cps@census.gov
## 5 dsd.cps@census.gov
## 6 dsd.cps@census.gov
```

You can use the `listCensusMetadata()` function to get more information about the variables of a specific API. Lets take a closer look at the variables of the "timeseries/poverty/saipe" API that provides small area estimates on poverty and income in the US (https://www.census.gov/data/developers/data-sets/Poverty-Statistics.html).

```
... # your work here
```

```
listCensusMetadata(name = "timeseries/poverty/saipe", type ="variables")
```

```
##                    name
## 1                   for
## 2                    in
## 3                 ucgid
## 4                  time
## 5  SAEPOVRT5_17R_LB90
## 6         SAEMHI_UB90
## 7    SAEPOVRT0_4_UB90
## 8     SAEPOVRT0_4_MOE
## 9    SAEPOVRTALL_LB90
## 10     SAEPOVALL_UB90
## 11              GEOID
## 12       SAEPOVALL_PT
## 13              STATE
## 14      SAEPOV0_17_PT
## 15      SAEPOVU_5_17R
## 16   SAEPOVRT0_17_MOE
## 17               YEAR
## 18   SAEPOVRT0_4_LB90
## 19  SAEPOVRT0_17_UB90
## 20     SAEPOVRT0_4_PT
## 21    SAEPOVRTALL_MOE
## 22       SAEPOVU_0_17
## 23    SAEPOV0_17_UB90
## 24       SAEPOV0_4_PT
## 25   SAEPOVRT5_17R_PT
## 26        SAEPOVU_0_4
## 27   SAEPOV5_17R_UB90
## 28             COUNTY
## 29  SAEPOVRT0_17_LB90
## 30    SAEPOV0_17_LB90
## 31          SAEMHI_PT
## 32    SAEPOVRT0_17_PT
## 33     SAEPOV5_17R_PT
## 34      SAEPOV0_4_MOE
## 35     SAEPOVALL_LB90
## 36     SAEPOV0_4_UB90
## 37            STABREV
## 38               NAME
## 39 SAEPOVRT5_17R_UB90
## 40     SAEPOVRTALL_PT
## 41      SAEPOVALL_MOE
## 42         SAEMHI_MOE
## 43        SAEPOVU_ALL
## 44     SAEPOV0_4_LB90
## 45   SAEPOVRTALL_UB90
## 46  SAEPOVRT5_17R_MOE
## 47    SAEPOV5_17R_MOE
## 48             GEOCAT
## 49        SAEMHI_LB90
## 50   SAEPOV5_17R_LB90
## 51     SAEPOV0_17_MOE
##                                                                               label
## 1                                                      Census API FIPS 'for' clause
## 2                                                       Census API FIPS 'in' clause
## 3                                        Uniform Census Geography Identifier clause
## 4                                                          ISO-8601 Date/Time value
## 5   Ages 5-17 in Families in Poverty, Rate Lower Bound for 90% Confidence Interval
## 6                   Median Household Income Upper Bound for 90% Confidence Interval
## 7                 Ages 0-4 in Poverty, Rate Upper Bound for 90% Confidence Interval
## 8                                          Ages 0-4 in Poverty, Rate Margin of Error
## 9                 All ages in Poverty, Rate Lower Bound for 90% Confidence Interval
## 10              All ages in Poverty, Count Upper Bound for 90% Confidence Interval
## 11                                                           State+County FIPS Code
## 12                                              All ages in Poverty, Count Estimate
## 13                                                                  FIPS State Code
## 14                                             Ages 0-17 in Poverty, Count Estimate
## 15                                                   Ages 5-17r in Poverty Universe
## 16                                       Ages 0-17 in Poverty, Rate Margin of Error
## 17                                                                    Estimate Year
## 18               Ages 0-4 in Poverty, Rate Lower Bound for 90% Confidence Interval
## 19              Ages 0-17 in Poverty, Rate Upper Bound for 90% Confidence Interval
## 20                                               Ages 0-4 in Poverty, Rate Estimate
## 21                                        All ages in Poverty, Rate Margin of Error
## 22                                                    Ages 0-17 in Poverty Universe
## 23             Ages 0-17 in Poverty, Count Upper Bound for 90% Confidence Interval
## 24                                              Ages 0-4 in Poverty, Count Estimate
## 25                                  Ages 5-17 in Families in Poverty, Rate Estimate
## 26                                                     Ages 0-4 in Poverty Universe
## 27 Ages 5-17 in Families in Poverty, Count Upper Bound for 90% Confidence Interval
## 28                                                                 County FIPS Code
## 29              Ages 0-17 in Poverty, Rate Lower Bound for 90% Confidence Interval
## 30                 Ages 0-17 in Poverty, Count Lower Bound 90% Confidence Interval
## 31                                                 Median Household Income Estimate
## 32                                              Ages 0-17 in Poverty, Rate Estimate
## 33                                 Ages 5-17 in Families in Poverty, Count Estimate
## 34                                       Ages 0-4 in Poverty, Count Margin of Error
## 35              All ages in Poverty, Count Lower Bound for 90% Confidence Interval
## 36              Ages 0-4 in Poverty, Count Upper Bound for 90% Confidence Interval
## 37                                             Two-letter State Postal abbreviation
## 38                                                             State or County Name
## 39  Ages 5-17 in Families in Poverty, Rate Upper Bound for 90% Confidence Interval
## 40                                               All ages in Poverty, Rate Estimate
## 41                                       All ages in Poverty, Count Margin of Error
## 42                                          Median Household Income Margin of Error
## 43                                                     All Ages in Poverty Universe
## 44              Ages 0-4 in Poverty, Count Lower Bound for 90% Confidence Interval
## 45               All ages in Poverty, Rate Upper Bound for 90% Confidence Interval
## 46                          Ages 5-17 in Families in Poverty, Rate Margin of Error
## 47                         Ages 5-17 in Families in Poverty, Count Margin of Error
## 48                                                                   Summary Level
## 49                 Median Household Income Lower Bound for 90% Confidence Interval
## 50 Ages 5-17 in Families in Poverty, Count Lower Bound for 90% Confidence Interval
## 51                                     Ages 0-17 in Poverty, Count Margin of Error
##                               concept predicateType group limit predicateOnly
## 1  Census API Geography Specification      fips-for   N/A     0          TRUE
## 2  Census API Geography Specification       fips-in   N/A     0          TRUE
## 3  Census API Geography Specification         ucgid   N/A     0          TRUE
## 4  Census API Date/Time Specification      datetime   N/A     0          TRUE
## 5                                <NA>         float   N/A     0          <NA>
## 6                                <NA>           int   N/A     0          <NA>
## 7                                <NA>         float   N/A     0          <NA>
## 8                                <NA>         float   N/A     0          <NA>
## 9                                <NA>         float   N/A     0          <NA>
## 10                               <NA>           int   N/A     0          <NA>
## 11                               <NA>        string   N/A     0          <NA>
## 12                               <NA>           int   N/A     0          <NA>
## 13                               <NA>        string   N/A     0          <NA>
## 14                               <NA>           int   N/A     0          <NA>
## 15                               <NA>           int   N/A     0          <NA>
## 16                               <NA>         float   N/A     0          <NA>
## 17                               <NA>           int   N/A     0          <NA>
## 18                               <NA>         float   N/A     0          <NA>
## 19                               <NA>         float   N/A     0          <NA>
## 20                               <NA>         float   N/A     0          <NA>
## 21                               <NA>         float   N/A     0          <NA>
## 22                               <NA>           int   N/A     0          <NA>
## 23                               <NA>           int   N/A     0          <NA>
## 24                               <NA>           int   N/A     0          <NA>
## 25                               <NA>         float   N/A     0          <NA>
## 26                               <NA>           int   N/A     0          <NA>
## 27                               <NA>           int   N/A     0          <NA>
## 28                               <NA>        string   N/A     0          <NA>
## 29                               <NA>         float   N/A     0          <NA>
## 30                               <NA>           int   N/A     0          <NA>
## 31                               <NA>           int   N/A     0          <NA>
## 32                               <NA>         float   N/A     0          <NA>
## 33                               <NA>           int   N/A     0          <NA>
## 34                               <NA>           int   N/A     0          <NA>
## 35                               <NA>           int   N/A     0          <NA>
## 36                               <NA>           int   N/A     0          <NA>
## 37                               <NA>        string   N/A     0          <NA>
## 38                               <NA>        string   N/A     0          <NA>
## 39                               <NA>         float   N/A     0          <NA>
## 40                               <NA>         float   N/A     0          <NA>
## 41                               <NA>           int   N/A     0          <NA>
## 42                               <NA>           int   N/A     0          <NA>
## 43                               <NA>           int   N/A     0          <NA>
## 44                               <NA>           int   N/A     0          <NA>
## 45                               <NA>         float   N/A     0          <NA>
## 46                               <NA>         float   N/A     0          <NA>
## 47                               <NA>           int   N/A     0          <NA>
## 48                               <NA>        string   N/A      0          <NA>
## 49                               <NA>           int   N/A      0          <NA>
## 50                               <NA>           int   N/A      0          <NA>
## 51                               <NA>           int   N/A      0          <NA>
##    hasGeoCollectionSupport required
## 1                     <NA>     <NA>
## 2                     <NA>     <NA>
## 3                     TRUE     <NA>
## 4                     <NA>     true
## 5                     <NA>     <NA>
## 6                     <NA>     <NA>
## 7                     <NA>     <NA>
## 8                     <NA>     <NA>
## 9                     <NA>     <NA>
## 10                    <NA>     <NA>
## 11                    <NA>     <NA>
## 12                    <NA>     <NA>
## 13                    <NA>     <NA>
## 14                    <NA>     <NA>
## 15                    <NA>     <NA>
## 16                    <NA>     <NA>
## 17                    <NA>     <NA>
## 18                    <NA>     <NA>
## 19                    <NA>     <NA>
## 20                    <NA>     <NA>
## 21                    <NA>     <NA>
## 22                    <NA>     <NA>
## 23                    <NA>     <NA>
## 24                    <NA>     <NA>
## 25                    <NA>     <NA>
## 26                    <NA>     <NA>
## 27                    <NA>     <NA>
## 28                    <NA>     <NA>
## 29                    <NA>     <NA>
## 30                    <NA>     <NA>
## 31                    <NA>     <NA>
## 32                    <NA>     <NA>
## 33                    <NA>     <NA>
## 34                    <NA>     <NA>
## 35                    <NA>     <NA>
## 36                    <NA>     <NA>
## 37                    <NA>     <NA>
## 38                    <NA>     <NA>
## 39                    <NA>     <NA>
## 40                    <NA>     <NA>
## 41                    <NA>     <NA>
## 42                    <NA>     <NA>
## 43                    <NA>     <NA>
## 44                    <NA>     <NA>
## 45                    <NA>     <NA>
## 46                    <NA>     <NA>
## 47                    <NA>     <NA>
## 48                    <NA>     <NA>
## 49                    <NA>     <NA>
## 50                    <NA>     <NA>
## 51                    <NA>     <NA>
```

Use the same function to retrieve information on the available regions for this API.

```
listCensusMetadata(name = "timeseries/poverty/saipe", type ="geography")
```

```
##     name geoLevelDisplay referenceDate requires wildcard optionalWithWCFor
## 1     us             010    2018-01-01     NULL     NULL              <NA>
## 2  state             040    2018-01-01     NULL     NULL              <NA>
## 3 county             050    2018-01-01    state    state             state
```

We can retrieve data using the `getCensus()` function. In this case, this requires the name of the API, the relevant variables, region and time.

Retrieve the percentage of people in poverty (“SAEPOVRTALL_PT”) and the median household income estimate (“SAEMHI_PT”) in 2020 per state. *Hint*: Specify the region as “state:*“.

```
getCensus(name="timeseries/poverty/saipe",
    vars=c("NAME", "SAEPOVRTALL_PT", "SAEMHI_PT"),
    region="state:*", time=2020)
```

```
##    time state                 NAME SAEPOVRTALL_PT SAEMHI_PT
## 1  2020    01              Alabama           14.9     53958
## 2  2020    02               Alaska            9.6     79961
## 3  2020    04              Arizona           12.8     64652
## 4  2020    05             Arkansas           15.2     51146
## 5  2020    06           California           11.5     83001
## 6  2020    08             Colorado            9.0     77688
## 7  2020    09          Connecticut            9.7     79723
## 8  2020    10             Delaware           10.9     71335
## 9  2020    11 District of Columbia           15.0     91957
## 10 2020    12              Florida           12.4     61724
## 11 2020    13              Georgia           14.0     62800
## 12 2020    15               Hawaii            8.9     86878
## 13 2020    16                Idaho           10.1     62603
## 14 2020    17             Illinois           11.0     71243
## 15 2020    18              Indiana           11.6     60794
## 16 2020    19                 Iowa           10.2     62362
## 17 2020    20               Kansas           10.6     63214
## 18 2020    21             Kentucky           14.9     54074
## 19 2020    22            Louisiana           17.8     51730
## 20 2020    23                Maine           10.6     59145
## 21 2020    24             Maryland            9.0     88589
## 22 2020    25        Massachusetts            9.4     87288
## 23 2020    26             Michigan           12.6     61352
## 24 2020    27            Minnesota            8.3     75489
## 25 2020    28          Mississippi           18.7     47368
## 26 2020    29             Missouri           12.1     58812
## 27 2020    30              Montana           12.4     57730
## 28 2020    31             Nebraska            9.2     64735
## 29 2020    32               Nevada           12.5     64608
## 30 2020    33        New Hampshire            7.0     81415
## 31 2020    34           New Jersey            9.4     87095
## 32 2020    35           New Mexico           16.8     52285
## 33 2020    36             New York           12.7     73354
## 34 2020    37       North Carolina           12.9     59616
## 35 2020    38         North Dakota           10.2     64289
## 36 2020    39                 Ohio           12.6     60360
## 37 2020    40             Oklahoma           14.3     54512
## 38 2020    41               Oregon           11.0     67832
## 39 2020    42         Pennsylvania           10.9     64898
## 40 2020    44         Rhode Island           10.6     73919
## 41 2020    45       South Carolina           13.8     57216
## 42 2020    46         South Dakota           11.6     61149
## 43 2020    47            Tennessee           13.6     56962
## 44 2020    48                Texas           13.4     66048
## 45 2020    49                 Utah            7.3     77785
## 46 2020    50              Vermont            9.4     67717
## 47 2020    51             Virginia            9.2     79154
## 48 2020    53           Washington            9.5     80319
## 49 2020    54        West Virginia           15.8     49202
## 50 2020    55            Wisconsin           10.0     64901
## 51 2020    56              Wyoming            9.2     67284
```

Reddit API

Reddit is a popular and influential social media platform that allows users to engage in a variety of activities, such as posting content, participating in discussions, and evaluating the submissions of other users within dedicated sections known as subreddits.

Reddit typically requires authentication via OAuth2. However, in practice, it's often not strictly necessary to authenticate yourself. In R, the RedditExtractoR package serves as a convenient wrapper for accessing the Reddit API.

Look into the documentation of the package, and extract the top post urls from the r/statistics subreddit.

... *# your work here*

```
top_stats_urls <- find_thread_urls(subreddit = "statistics", sort_by='top')
head(top_stats_urls)
```

Query the r/aww subreddit for the urls of discussions that included the "cat" keyword during last week.

... *# your work here*

```
top_aww_cats <- find_thread_urls(subreddit = "aww", keywords = 'cat', sort_by='top')
head(top_aww_cats)
```

Extract the content of a specified thread (e.g. one of the urls your collected).

... *# your work here*

```
top_stats_post <- get_thread_content(top_stats_urls$url[1])
head(top_stats_post)
```

Find subreddits that include have something to do with machine learning. Display the top ten choices sorted by subscriber count.

... *# your work here*

```
subreddits_machinelearning <- find_subreddits(keywords = "machine learning")

top_ten_ml_subreddits <- subreddits_machinelearning %>% select(subreddit, title, subscribers) %>% arrang
rownames(top_ten_ml_subreddits) <- NULL
```

```
head(top_ten_ml_subreddits, n = 10)
```

Retrieve information about a particular user (e.g. “GovSchwarzenegger”).

. . . # your work here

```
arnold_reddit <- get_user_content('GovSchwarzenegger')
```

Open Street Map API

OpenStreetMap is a international open access mapping project. You can access OSM using the osmdata package.

The OPM API is fairly complicated, so feel free to consult relevant documentation if you are unsure how to proceed. Use the `available_features()` function to get a list of physical features recorded in OSM. You can then use the `available_tags()` function to explore the associated tags for each feature.

. . . # your work here

```
available_features()
```

```
##   [1] "4wd_only"                    "abandoned"
##   [3] "abutters"                    "access"
##   [5] "addr"                        "addr:city"
##   [7] "addr:conscriptionnumber"     "addr:country"
##   [9] "addr:county"                 "addr:district"
##  [11] "addr:flats"                  "addr:full"
##  [13] "addr:hamlet"                 "addr:housename"
##  [15] "addr:housenumber"            "addr:inclusion"
##  [17] "addr:interpolation"          "addr:place"
##  [19] "addr:postbox"                "addr:postcode"
##  [21] "addr:province"               "addr:state"
##  [23] "addr:street"                 "addr:subdistrict"
##  [25] "addr:suburb"                 "addr:unit"
##  [27] "admin_level"                 "aeroway"
##  [29] "agricultural"                "alt_name"
##  [31] "amenity"                     "area"
##  [33] "atv"                         "backward"
##  [35] "barrier"                     "basin"
##  [37] "bdouble"                     "bicycle"
##  [39] "bicycle_road"                "biergarten"
##  [41] "boat"                        "border_type"
##  [43] "boundary"                    "brand"
##  [45] "bridge"                      "building"
##  [47] "building:colour"             "building:fireproof"
##  [49] "building:levels"             "building:material"
##  [51] "building:min_level"          "building:part"
##  [53] "building:soft_storey"        "bus_bay"
##  [55] "busway"                      "capacity"
##  [57] "castle_type"                 "change"
##  [59] "charge"                      "clothes"
##  [61] "construction"                "construction_date"
##  [63] "construction#Railways"       "covered"
##  [65] "craft"                       "crossing"
##  [67] "crossing:island"             "cuisine"
##  [69] "cutting"                     "cycleway"
##  [71] "denomination"                "destination"
##  [73] "diet:*"                      "direction"
##  [75] "dispensing"                  "disused"
##  [77] "drinking_water"              "drive_in"
##  [79] "drive_through"               "ele"
##  [81] "electric_bicycle"            "electrified"
##  [83] "embankment"                  "embedded_rails"
##  [85] "emergency"                   "end_date"
##  [87] "entrance"                    "est_width"
##  [89] "fee"                         "female"
##  [91] "fire_object:type"            "fire_operator"
##  [93] "fire_rank"                   "food"
##  [95] "foot"                        "footway"
##  [97] "ford"                        "forestry"
##  [99] "forward"                     "frequency"
## [101] "fuel"                        "gauge"
## [103] "golf_cart"                   "goods"
## [105] "hazard"                      "hazmat"
## [107] "healthcare"                  "healthcare:counselling"
## [109] "healthcare:speciality"       "height"
## [111] "hgv"                         "highway"
## [113] "historic"                    "horse"
## [115] "hot_water"                   "ice_road"
## [117] "incline"                     "industrial"
## [119] "inline_skates"               "inscription"
## [121] "int_name"                    "internet_access"
## [123] "junction"                    "kerb"
## [125] "landuse"                     "lanes"
## [127] "lanes:bus"                   "lanes:psv"
## [129] "layer"                       "leaf_cycle"
## [131] "leaf_type"                   "leisure"
## [133] "lhv"                         "lit"
## [135] "loc_name"                    "location"
## [137] "male"                        "man_made"
## [139] "max_age"                     "max_level"
## [141] "maxaxleload"                 "maxheight"
## [143] "maxlength"                   "maxspeed"
## [145] "maxstay"                     "maxweight"
## [147] "maxwidth"                    "military"
## [149] "min_age"                     "min_level"
## [151] "minspeed"                    "mofa"
## [153] "moped"                       "motor_vehicle"
## [155] "motorboat"                   "motorcar"
## [157] "motorcycle"                  "motorroad"
## [159] "mountain_pass"               "mtb:description"
## [161] "mtb:scale"                   "name"
## [163] "name_1"                      "name_2"
## [165] "name:left"                   "name:right"
## [167] "narrow"                      "nat_name"
## [169] "natural"                     "nickname"
## [171] "noexit"                      "non_existent_levels"
## [173] "nudism"                      "office"
## [175] "official_name"               "old_name"
## [177] "oneway"                      "oneway:bicycle"
## [179] "opening_hours"               "opening_hours:drive_through"
## [181] "operator"                    "operator:type"
## [183] "orientation"                 "oven"
## [185] "overtaking"                  "parking"
## [187] "parking:condition"           "parking:lane"
## [189] "passing_places"              "place"
## [191] "power"                       "power_supply"
## [193] "priority"                    "priority_road"
## [195] "produce"                     "proposed"
## [197] "protected_area"              "psv"
## [199] "public_transport"            "railway"
## [201] "railway:preserved"           "railway:track_ref"
## [203] "recycling_type"              "ref"
## [205] "reg_name"                    "religion"
## [207] "rental"                      "residential"
## [209] "roadtrain"                   "route"
## [211] "sac_scale"                   "sauna"
## [213] "service"                     "service_times"
## [215] "shelter_type"                "shop"
## [217] "short_name"                  "shower"
## [219] "sidewalk"                    "site"
## [221] "ski"                         "smoothness"
## [223] "social_facility"             "sorting_name"
## [225] "speed_pedelec"               "start_date"
## [227] "step_count"                  "substation"
## [229] "surface"                     "tactile_paving"
## [231] "tank"                        "tidal"
## [233] "toilets"                     "toilets:wheelchair"
## [235] "toll"                        "topless"
## [237] "tourism"                     "tracks"
## [239] "tracktype"                   "traffic_calming"
## [241] "traffic_sign"                "trail_visibility"
## [243] "trailblazed"                 "trailblazed:visibility"
## [245] "tunnel"                      "turn"
## [247] "type"                        "unisex"
## [249] "usage"                       "vehicle"
## [251] "vending"                     "voltage"
## [253] "water"                       "wheelchair"
## [255] "wholesale"                   "width"
## [257] "winter_road"                 "wood"
```

```
available_tags("amenity")
```

```
## # A tibble: 129 x 2
##    Key     Value
##    <chr>   <chr>
##  1 amenity animal_boarding
##  2 amenity animal_breeding
##  3 amenity animal_shelter
##  4 amenity animal_training
##  5 amenity arts_centre
##  6 amenity atm
##  7 amenity baby_hatch
##  8 amenity baking_oven
##  9 amenity bank
## 10 amenity bar
## # i 119 more rows
```

You have to use a so called bounding box to define geographical area you want to include in your query. Use the `getbb()` function to define a bounding box for Munich.

. . . *# your work here*

```r
location <- getbb(place_name = "Munich")
```

Use the `opq()`, `add_osm_feature()` and `osmdata_sf()` function to query for all arts centres in Munich. *Hint:* Use the “amenity” feature.

. . . *# your work here*

```r
arts_center_munich <- opq(bbox = location) %>%
  add_osm_feature(key = 'amenity', value = c('arts_centre')) %>%
  osmdata_sf()
```

Extract the highest-performance roads and natural water sources in Munich.

. . . *# your work here*

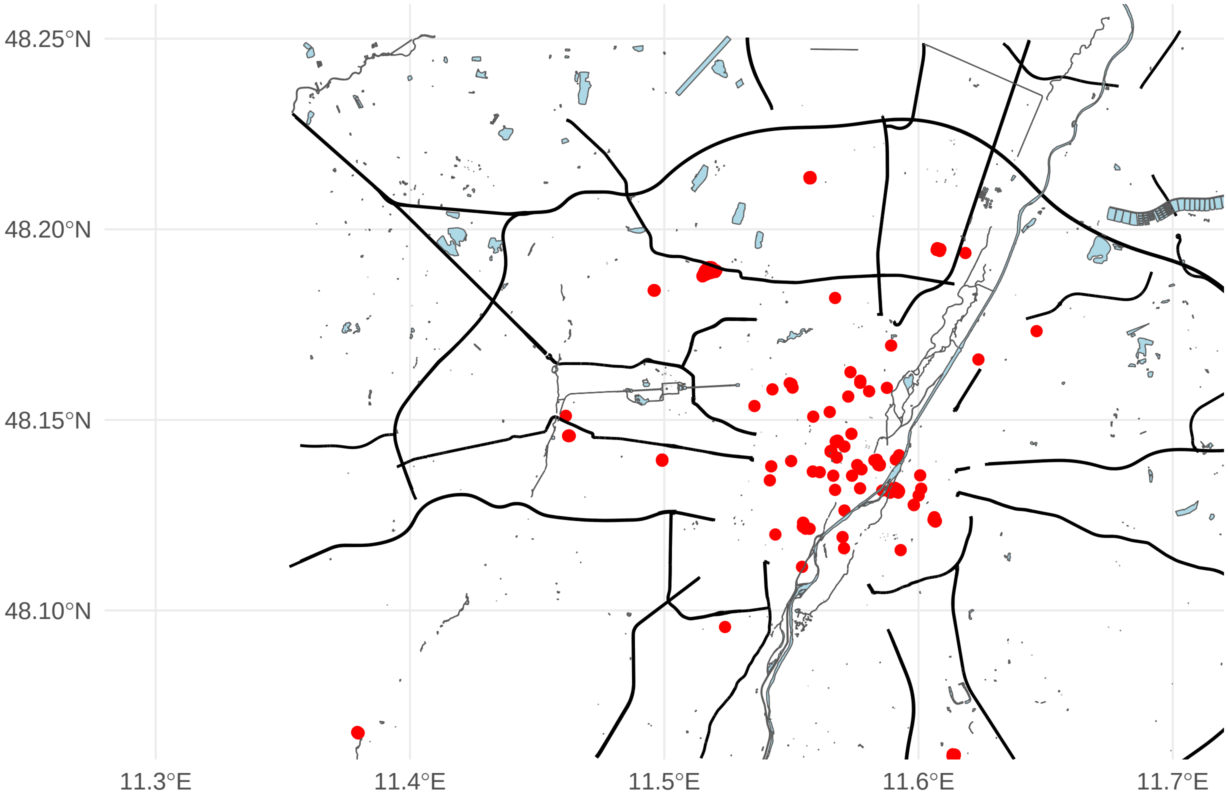
```r
streets <- opq(bbox = location) %>%
    add_osm_feature(key = 'highway', value = c('motorway', 'primary', 'highway')) %>%
   osmdata_sf()

water <- opq(bbox = location) %>%
  add_osm_feature(key = 'natural', value = 'water') %>%
  osmdata_sf()
```

Plot the results of all of your queries into one map using the `geom_sf` function.

. . . *# your work here*

```r
ggplot() +
  geom_sf(data = arts_center_munich$osm_points, color = 'red') +
  geom_sf(data = streets$osm_lines) +
  geom_sf(data = water$osm_polygons, fill = 'light blue') +
  theme_minimal() + coord_sf(xlim = c(11.3, 11.7), ylim = c(48.07, 48.25))
```

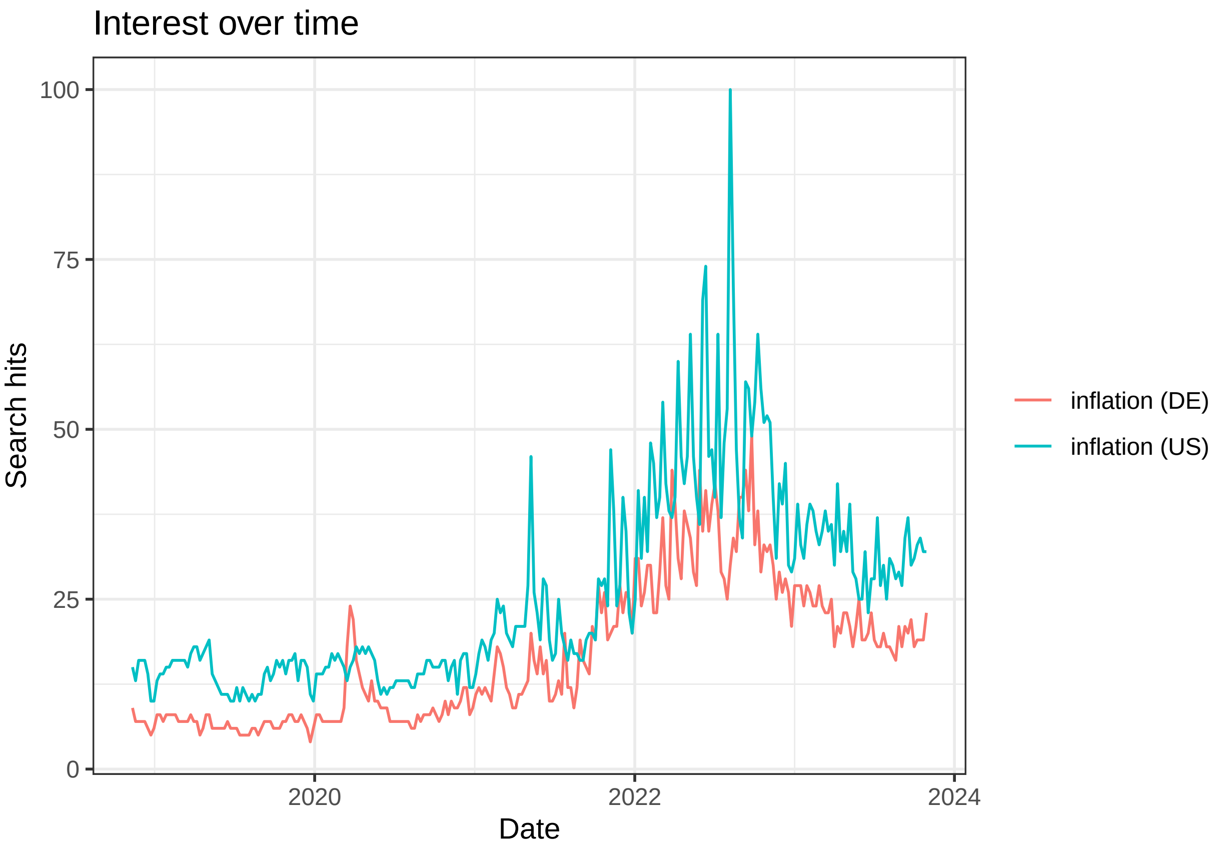


Google Trends API

The Google Trends API can be used without additional authentication. In R, we can access the API using the `httr2` package, but for a more user-friendly approach make use of the dedicated `gtrendsR` package. Get the search interest data for "inflation" over the last five years in Germany and the US and plot the result. *Hint:* If you are unsure you can find the relevant country codes in the `data('countries')` dataset.

. . . # your work here

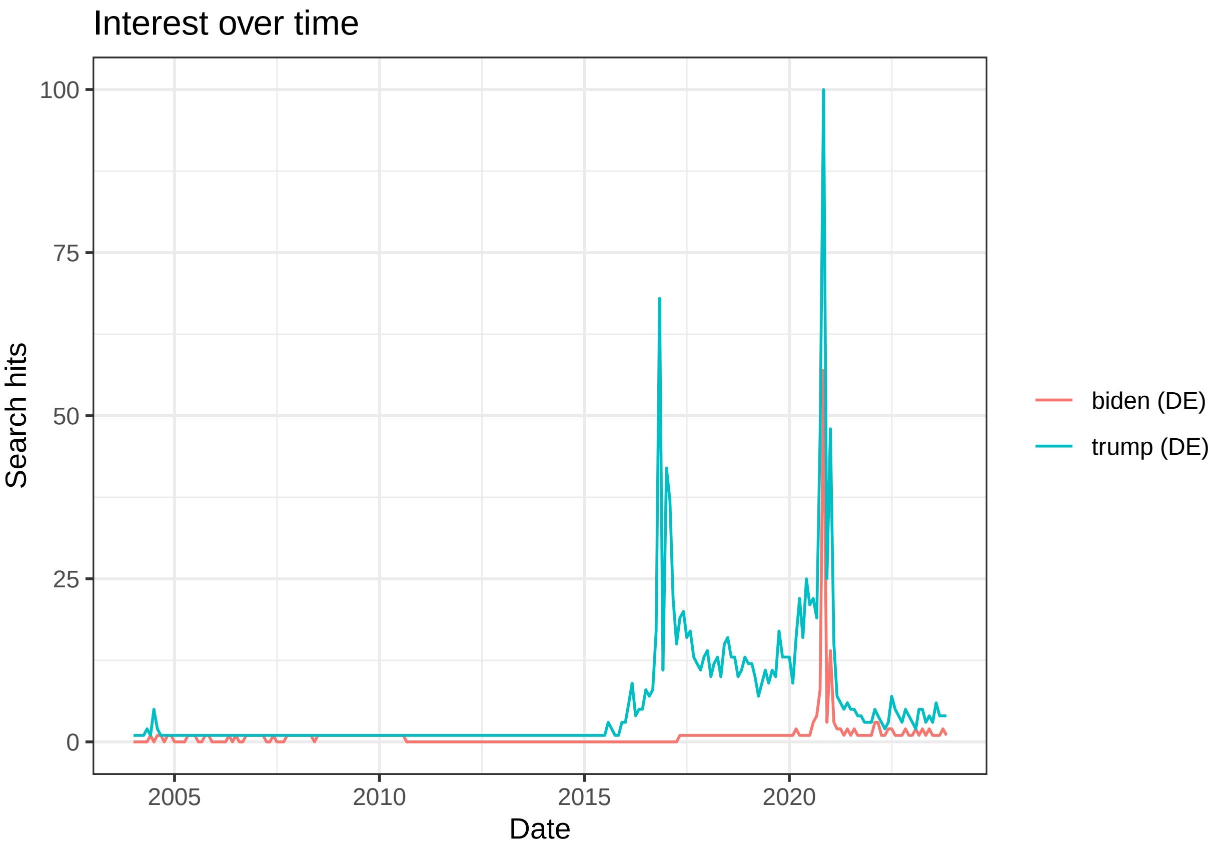
```
res <- gtrends(c('inflation'), geo=c("DE", "US"))
plot(res)
```



Now compare the search interest in “trump” and “biden” in Germany since the beginning of Google Trends.

... *# your work here*

```
res <- gtrends(c('biden', 'trump'), geo = c('DE'), time = 'all')
plot(res)
```



Spotify API

You will need a Spotify Account to access the Spotify API. Once you have a Spotify Account you can create a developer account here. Create a new app on the dashboard (you can use http://localhost:3000/ as redirect link). Safely store your credentials inside the R enviornment.

We will use the spotifyr package to access the API. Use the `get_spotify_access_token()` function to create a Spotify access token from your credentials.

... *# your work here*

```
access_token <- get_spotify_access_token()
```

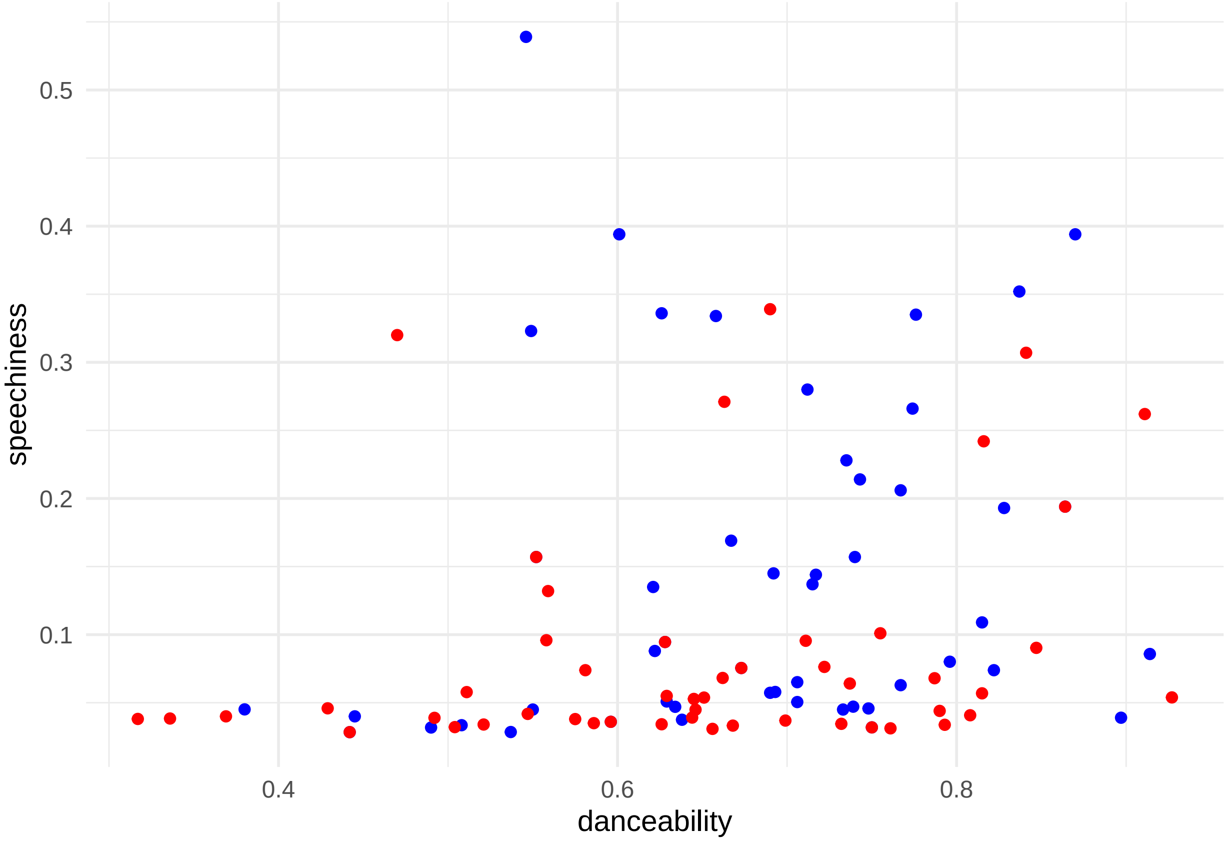
We are going to analyze the features of the Top 50 - Germany playlist and Top 50 - US playlist playlist. You can find the right id in your browser link when opening the playlist. Use the `get_playlist_audio_features()` function.

... *# your work here*

```
top50_germany_features <- get_playlist_audio_features(playlist_uris = "37i9dQZEVXbJiZcmkrIHGU")
top50_us_features <- get_playlist_audio_features(playlist_uris = "37i9dQZEVXbLRQDuF5jeBp")
```

Plot the "danceability" versus the "speechiness" of tracks in the playlist for both countries. ... *# your work here*

```
ggplot() +
  geom_point(data = top50_germany_features, aes(x = danceability, y = speechiness), color = "blue") +
  geom_point(data = top50_us_features, aes(x = danceability, y = speechiness), color = "red") +
  labs(x = "danceability", y = "speechiness") +
  theme_minimal()
```



Bonus: ChatGPT

The ChatGPT API is not available for free. However, OpenAI typically provides users with some complimentary tokens for the initial weeks after sign-up. If you do not have access to free tokens, feel free to skip this exercise.

To get started, you will need your API key found at the following https://platform.openai.com/account/api-keys.

Define a prompt you want to send to ChatGPT.

. . . *# your work here*

```
prompt <- "Explain to me what APIs are."
```

Use the following `request` to send your prompt to the server.

```
response <- request("https://api.openai.com/v1/chat/completions") %>%
  req_headers(Authorization = paste("Bearer", Sys.getenv("OPENAI_KEY"))) %>%
  req_body_json(list(
    model = "gpt-3.5-turbo",
    temperature = 1,
    messages = list(list(
      role = "user",
      content = prompt
    ))
  ))
```

```
response %>% req_dry_run()
```

```
## POST /v1/chat/completions HTTP/1.1
## Host: api.openai.com
## User-Agent: httr2/0.2.3 r-curl/5.1.0 libcurl/7.85.0
## Accept: */*
## Accept-Encoding: deflate, gzip
## Authorization: <REDACTED>
## Content-Type: application/json
## Content-Length: 111
##
## {"model":"gpt-3.5-turbo","temperature":1,"messages":[{"role":"user","content":"Explain to me what AP
```

In both Python and R, various packages have emerged to streamline interactions with ChatGPT. Given the fast changing nature of this landscape, we encourage you to explore and discover your preferred solution that works best for your specific requirements and preferences.