APIs and R

Erika Rasnick and Andrew Vancil

DBE Lunch and Learn

May 6, 2021

API Overview

- Application Programming Interface
- defines interactions between different software applications
- simplifies programming by abstracting the underlying implementation and only exposing objects or actions the developer needs

APIs and R

- APIs are often used to request datasets directly from websites instead of downloading them.
- APIs will have unique URLs and access keys that you will need to access the data.
- We can think of an API as a computer waiting for data requests. We need to write R code that tells the API what we need.
- The API will read our code, process the request, and return nicely-formatted data that can be easily parsed by existing R libraries

APIs and R - {`httr`}

- We can make a `GET` request using the R package `httr`
- The `httr::GET()` function requires a URL - the address of the server that the request needs to be sent to

```
library(httr)
library(jsonlite)
response = httr::GET("http://api.open-notify.org/astros.json")
```

- We get a response back from the API

```
response
```

```
## Response [http://api.open-notify.org/astros.json]
##   Date: 2021-05-05 20:03
##   Status: 200
##   Content-Type: application/json
##   Size: 355 B
```

APIs and R - {httr}

- The actual data is not in a human-usable format.

```
response$content
```

```
##   [1] 7b 22 6e 75 6d 62 65 72 22 3a 20 37 2c 20 22 6d 65 73 73 61 67 65 22
##  [26] 22 73 75 63 63 65 73 73 22 2c 20 22 70 65 6f 70 6c 65 22 3a 20 5b 7b
##  [51] 61 6d 65 22 3a 20 22 4d 61 72 6b 20 56 61 6e 64 65 20 48 65 69 22 2c
##  [76] 63 72 61 66 74 22 3a 20 22 49 53 53 22 7d 2c 20 7b 22 6e 61 6d 65 22
## [101] 22 4f 6c 65 67 20 4e 6f 76 69 74 73 6b 69 79 22 2c 20 22 63 72 61 66
## [126] 3a 20 22 49 53 53 22 7d 2c 20 7b 22 6e 61 6d 65 22 3a 20 22 50 79 6f
## [151] 20 44 75 62 72 6f 76 22 2c 20 22 63 72 61 66 74 22 3a 20 22 49 53 53
## [176] 2c 20 7b 22 6e 61 6d 65 22 3a 20 22 54 68 6f 6d 61 73 20 50 65 73 71
## [201] 74 22 2c 20 22 63 72 61 66 74 22 3a 20 22 49 53 53 22 7d 2c 20 7b 22
## [226] 6d 65 22 3a 20 22 4d 65 67 61 6e 20 4d 63 41 72 74 68 75 72 22 2c 20
## [251] 72 61 66 74 22 3a 20 22 49 53 53 22 7d 2c 20 7b 22 6e 61 6d 65 22 3a
## [276] 53 68 61 6e 65 20 4b 69 6d 62 72 6f 75 67 68 22 2c 20 22 63 72 61 66
## [301] 3a 20 22 49 53 53 22 7d 2c 20 7b 22 6e 61 6d 65 22 3a 20 22 41 6b 69
## [326] 6b 6f 20 48 6f 73 68 69 64 65 22 2c 20 22 63 72 61 66 74 22 3a 20 22
## [351] 53 22 7d 5d 7d
```

```
fromJSON(rawToChar(response$content))
```

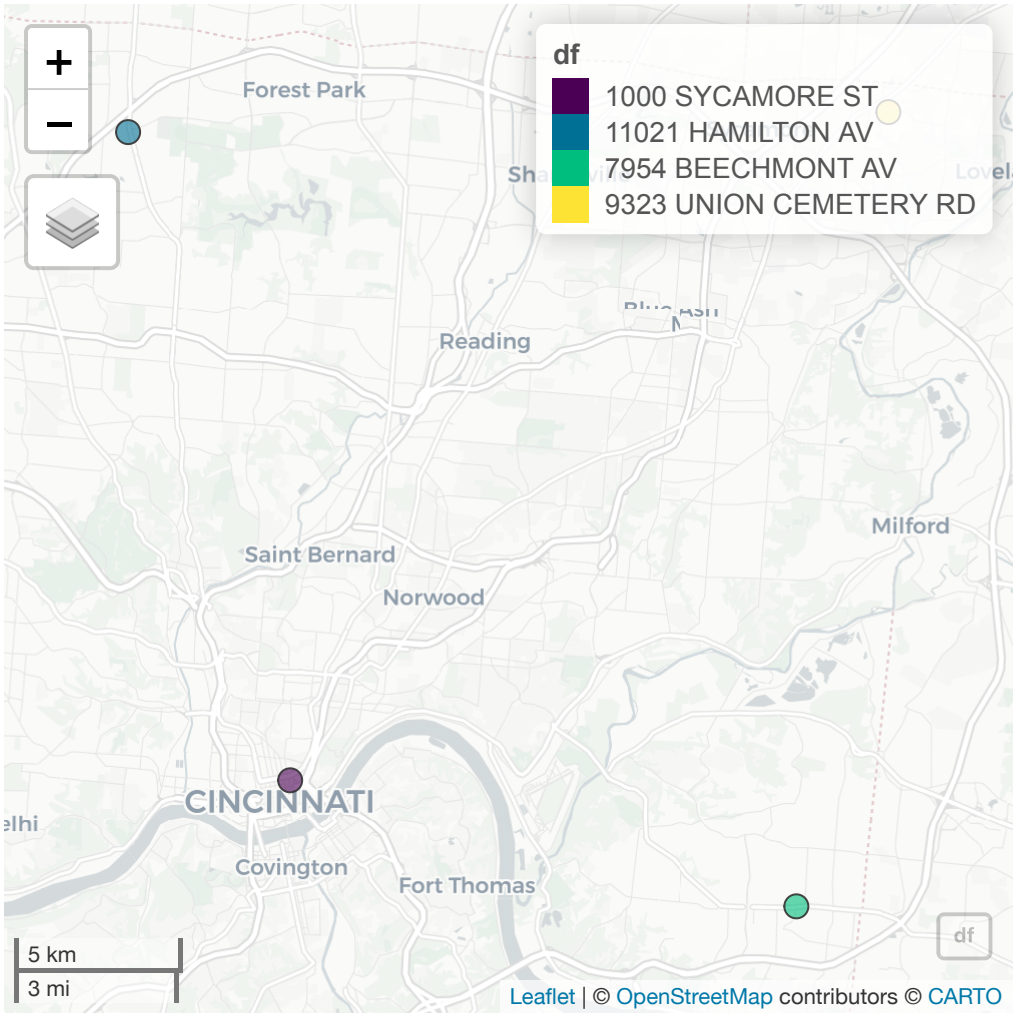
APIs and R - {`httr`}

```
##              name craft
## 1  Mark Vande Hei   ISS
## 2  Oleg Novitskiy   ISS
## 3    Pyotr Dubrov   ISS
## 4  Thomas Pesquet   ISS
## 5  Megan McArthur   ISS
## 6 Shane Kimbrough   ISS
## 7 Akihiko Hoshide   ISS
```

APIs and R - {httr}

```
library(httr); library(jsonlite); library(tidyverse)

res <- GET("http://cagisonline.hamilton-co.org/arcgis/rest/services/C

dat <- content(res, as = "text", encoding = "UTF-8")

df <- fromJSON(dat, flatten = TRUE) %>%
  data.frame() %>%
  dplyr::filter(!is.na(features.attributes.NEWADDR)) %>%
  dplyr::select(address = features.attributes.NEWADDR, lon = features
  sf::st_as_sf(coords = c('lon', 'lat'), crs = 4326)
```

```
mapview::mapview(df)
```



R API Wrapper Packages

- Many R wrapper packages have been created to make accessing APIs using R even easier.
- These packages consist of code similar the previous example that have been wrapped into user-friendly R functions, meaning the user doesn't need to find the URL for the API, know how to use the `httr` package, or understand and deal with the format of the response.

`tidycensus`

- "`tidycensus` is an R package that allows users to interface with the US Census Bureau's decennial Census and five-year American Community APIs and return tidyverse-ready data frames."
- After setting up a Census API Key (essentially a username/password for accessing Census data), querying Census data in R is relatively straightforward.

This example illustrates how to get the median age by state from the 2010 decennial census.

```r
library(tidycensus)
age10 <- get_decennial(geography = "state",
                       variables = "P013001",
                       year = 2010)
```

```
## Getting data from the 2010 decennial Census

## Using Census Summary File 1
```

```r
head(age10)
```

```
## # A tibble: 6 x 4
##   GEOID NAME       variable value
##   <chr> <chr>      <chr>    <dbl>
## 1 01    Alabama    P013001   37.9
## 2 02    Alaska     P013001   33.8
## 3 04    Arizona    P013001   35.9
## 4 05    Arkansas   P013001   37.4
## 5 06    California P013001   35.2
## 6 22    Louisiana  P013001   35.8
```

`tigris`

- `tigris` is an R package that allows users to directly download and use TIGER/Line shapefiles from the US Census Bureau.
- Available data include boundaries for states, counties, tracts, block groups, blocks, ZCTAs, school districts, voting districts, roads, and more.
- `tigris` returns data as simple features objects, which interface with the `sf` and `ggplot2` packages, as well as others.

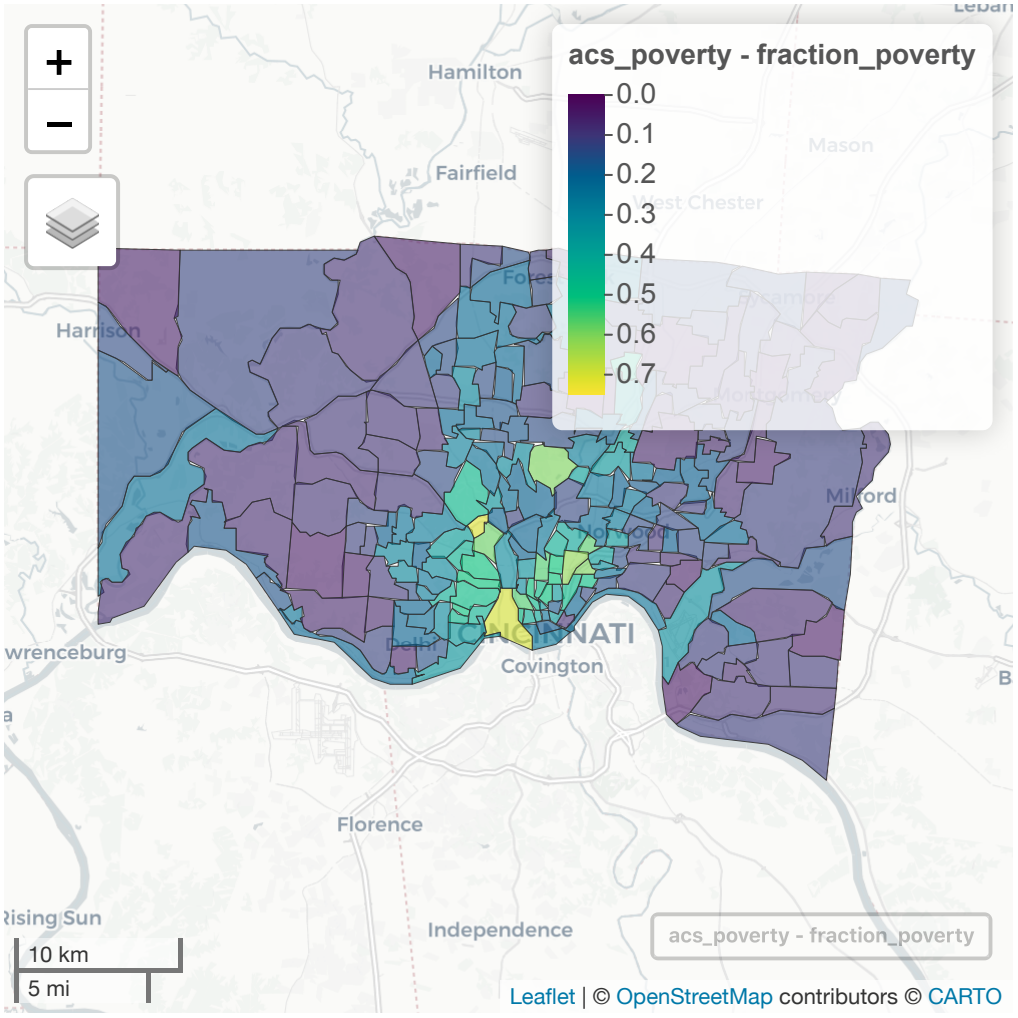
```r
library(tigris)
library(ggplot2)

manhattan_roads <- roads("NY", "New York")
ggplot(manhattan_roads) +
  geom_sf() +
  theme_void()
```

Using `tidycensus` and `tigris`

```
acs_poverty <- tidycensus::get_acs(geography = 'tract',
                                   variables = 'B17001_002',
                                   summary_var = 'B17001_001',
                                   year = 2018,
                                   state = 'Ohio',
                                   county = 'Hamilton',
                                   geometry = TRUE) %>%
  dplyr::mutate(fraction_poverty = estimate / summary_est) %>%
  dplyr::select(tract_id = GEOID, fraction_poverty)
```

```
## Simple feature collection with 222 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -84.82016 ymin: 39.02208 xmax: -84.25651 ymax: 39.312
## Geodetic CRS:  NAD83
## First 10 features:
##       tract_id fraction_poverty                       geometry
## 1  39061002500       0.4870996 MULTIPOLYGON (((-84.51982 3...
## 2  39061002700       0.2944625 MULTIPOLYGON (((-84.53346 3...
## 3  39061004000       0.1189446 MULTIPOLYGON (((-84.46725 3...
## 4  39061005301       0.1539427 MULTIPOLYGON (((-84.43285 3...
## 5  39061008300       0.2571858 MULTIPOLYGON (((-84.58361 3...
## 6  39061008800       0.4374615 MULTIPOLYGON (((-84.5807 39...
## 7  39061010003       0.2892442 MULTIPOLYGON (((-84.59709 3...
## 8  39061020762       0.1873174 MULTIPOLYGON (((-84.58611 3...
## 9  39061021422       0.1263052 MULTIPOLYGON (((-84.60225 3...
## 10 39061023400       0.2085094 MULTIPOLYGON (((-84.4543 39...
```

```r
mapview::mapview(acs_poverty, zcol = 'fraction_poverty')
```



HealthData.gov

- Using the package `RSocrata`, you can easily access datasets using a single command

```
df <- read.socrata("https://healthdata.gov/resource/g62h-syeh.json")
```

- This command will import the dataset for Covid-19 Patient Impact and Hospital Capacity by State Timeseries

d

Filter | Cols: « < 0 – 50 > »

	state	date	critical_staffing_shortage_today_yes	critical_staffing_shortage_today_no	critical_staffing_shortage_today_not_reported	critical_staffing_shortage_anticipated_within_week_yes	critical_s
1	MO	2020-08-04	20	68	47	29	58
2	MS	2020-08-04	33	59	4	40	48
3	MT	2020-08-04	4	22	40	7	19
4	NC	2020-08-04	4	93	23	11	90
5	ND	2020-08-04	12	30	5	12	29
6	NE	2020-08-04	4	4	89	4	4

Showing 1 to 6 of 100 entries, 62 total columns

REDCap API

- You can quickly access your REDCap project data using an api wrapper package called `redcapAPI`
- Using your project-specific API access token and REDCap url, you are able to access your data

```
rcon <- redcapConnection(url = my_url, token = my_token)
dat <- exportRecords(rcon)
```

- It is also possible to specify specific fields and records in the `exportRecords()` function

REDCap Example

- We currently use the `redcapAPI` package to access user experience survey data for DeGAUSS

```
my_url <- c('https://redcap.research.cchmc.org/api/')
my_token <- c('D7...')

options(redcap_api_url = my_url)
rcon <- redcapConnection(url = my_url, token = my_token)
dat <- exportRecords(rcon, labels=F)
write.csv(dat,"survey_raw_data.csv")
```

	record_id	redcap_survey_identifier	degauss_user_survey_timestamp	institution	terminal_deg	other_deg	user_status
1	1	*NA*	*NA*	Columbia University	BA/BS	*NA*	*NA*
2	2	*NA*	*NA*	CCHMC	MA/MS	*NA*	*NA*
3	3	*NA*	*NA*	Vanderbilt University	PhD.	*NA*	*NA*

Creating Your Own API With `plumber`

- add special comments to code to expose existing R code as a service to others on the Web.

Example Use Case:

- You build a predictive model. You want others to be able to use the model to predict for new data.
- You can write a function with `plumber` comments that turns the predict function into an API.
- Other users can pass new data to the API and get back predictions.

More APIs

- A wrapper packaged for the Google Maps API: `mapsapi` that allows you to pull directions, distances, rasters and geocoded locations (point or polygon)
- Baseball fans can access interesting data using the package `baseballr`
  - `baseballr` uses the official MLB stats API and can pull pitch-by-pitch data, as well as source data from various baseball statistics databases
- RapidAPI is an online repository of thousands of APIs across hundreds of topics
  - Most are free, but generally opensource, so quality and validity can be hard to verify
- List of more useful wrapper packages

APIs and Data Privacy

- Because APIs are hosted via the internet, we would never want to send any PHI through an API request.
- Recent issues with mobile health apps exposing PHI via APIs.
    - Recent study suggested that PHI is vulnerable in most mobile health apps via the APIs that they use

References

https://en.wikipedia.org/wiki/API

https://www.dataquest.io/blog/r-api-tutorial/

https://walker-data.com/tidycensus/index.html

https://github.com/walkerke/tigris

https://www.rplumber.io/

https://www.rstudio.com/resources/webinars/expanding-r-horizons-integrating-r-with-plumber-apis/

https://www.hipaajournal.com/100-of-tested-mhealth-apps-vulnerable-to-api-attacks/