

POIR 613: Computational Social Science

Pablo Barberá

University of Southern California

`pablobarbera.com`

Course website:

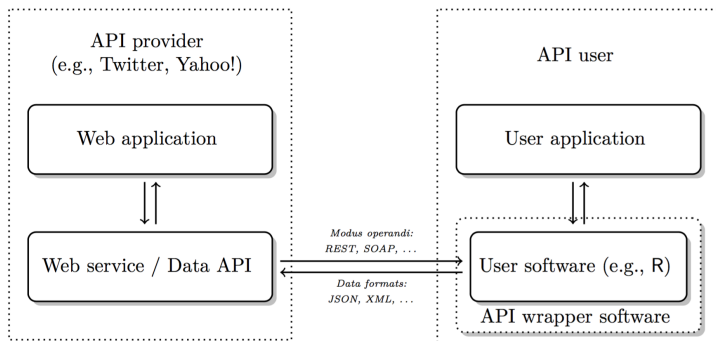
pablobarbera.com/POIR613/

APIs

APIs

API = Application Programming Interface; a set of structured http requests that return data in a lightweight format.

HTTP = Hypertext Transfer Protocol; how browsers and e-mail clients communicate with servers.



Source: Munzert et al, 2014, Figure 9.8

APIs

Types of APIs:

1. **RESTful APIs**: queries for static information at current moment (e.g. user profiles, posts, etc.)
2. **Streaming APIs**: changes in users' data in real time (e.g. new tweets, weather alerts...)

APIs generally have extensive **documentation**:

- ▶ Written for developers, so must be understandable for humans
- ▶ What to look for: **endpoints** and **parameters**.

Most APIs are **rate-limited**:

- ▶ Restrictions on number of API calls by user/IP address and period of time.
- ▶ Commercial APIs may impose a monthly fee

Connecting with an API

Constructing a REST API call:

- ▶ Baseline URL **endpoint**:

```
https://maps.googleapis.com/maps/api/geocode/json
```

- ▶ Parameters: ?address=budapest
- ▶ Authentication token (optional): &key=XXXXX

From R, use `httr` package to make GET request:

```
library(httr)
r <- GET(
  "https://maps.googleapis.com/maps/api/geocode/json",
  query=list(address="budapest"))
```

If request was successful, returned code will be 200, where 4xx indicates client errors and 5xx indicates server errors. If you need to attach data, use POST request.

```
{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "Budapest",
          "short_name" : "Budapest",
          "types" : [ "locality", "political" ]
        },
        {
          "long_name" : "Hungary",
          "short_name" : "HU",
          "types" : [ "country", "political" ]
        }
      ],
      "formatted_address" : "Budapest, Hungary",
      "geometry" : {
        "bounds" : {
          "northeast" : {
            "lat" : 47.6130119,
            "lng" : 19.3345049
          },
          "southwest" : {
            "lat" : 47.349415,
            "lng" : 18.9261011
          }
        },
        "location" : {
          "lat" : 47.497912,
          "lng" : 19.040235
        }
      },
      ...
    }
  ]
}
```

```
{
...
  "location_type" : "APPROXIMATE",
  "viewport" : {
    "northeast" : {
      "lat" : 47.6130119,
      "lng" : 19.3345049
    },
    "southwest" : {
      "lat" : 47.349415,
      "lng" : 18.9261011
    }
  },
  "place_id" : "ChIJyc_U0TTDQucRYBEeDCnEAAQ",
  "types" : [ "locality", "political" ]
},
"status" : "OK"
}
```

JSON

Response is often in **JSON format** (Javascript Object Notation).

- ▶ Type: `content(r, "text")`
- ▶ Data stored in key-value pairs. Why? Lightweight, more flexible than traditional table format.
- ▶ Curly brackets embrace objects; square brackets enclose arrays (vectors)
- ▶ Use `fromJSON` function in `jsonlite` the package or `stream_in` in `ndjson` the package to read JSON data
- ▶ But many packages have their own specific functions to read data in JSON format; `content(r, "parsed")`

Authentication

- ▶ Many APIs require an access key or token
- ▶ An alternative, open standard is called OAuth
- ▶ Connections without sharing username or password, only temporary tokens that can be refreshed
- ▶ `httr` package in R implements most cases ([examples](#))

R packages

Before starting a new project, worth checking if there's already an R package for that API. Where to look?

- ▶ [CRAN Web Technologies Task View](#) (but only packages released in CRAN)
- ▶ GitHub (including unreleased packages and most recent versions of packages)
- ▶ [rOpenSci Consortium](#)

Also see this [great list of APIs](#) in case you need inspiration.

Why APIs?

Advantages:

- ▶ 'Pure' data collection: avoid malformed HTML, no legal issues, clear data structures, more trust in data collection...
- ▶ Standardized data access procedures: transparency, replicability
- ▶ Robustness: benefits from 'wisdom of the crowds'

Disadvantages

- ▶ They're not too common (yet!)
- ▶ Dependency on API providers
- ▶ Lack of natural connection to R