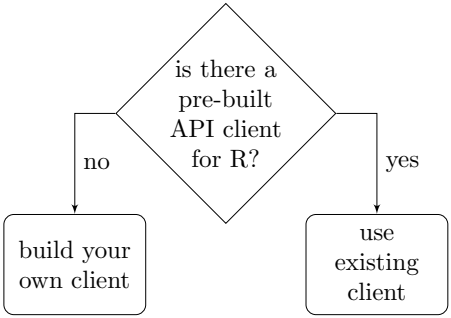
Introduction to Web Scraping with R

Accessing APIs from Scratch

Simon Munzert | IPSDS

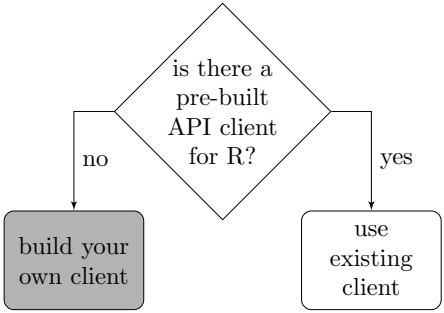
API access with R

There are basically **two scenarios:**



API access with R

There are basically **two scenarios:**



Here, we talk about the case where you have to build your own API binding.

Accessing APIs from Scratch

Accessing APIs from Scratch

Steps to be taken

Accessing APIs from Scratch

Steps to be taken

1. figure out how the API works: every API has a human-readable documentation for developers

Accessing APIs from Scratch

Steps to be taken

1. figure out how the API works: every API has a human-readable documentation for developers
2. build access to the API via R

Accessing APIs from Scratch

Steps to be taken

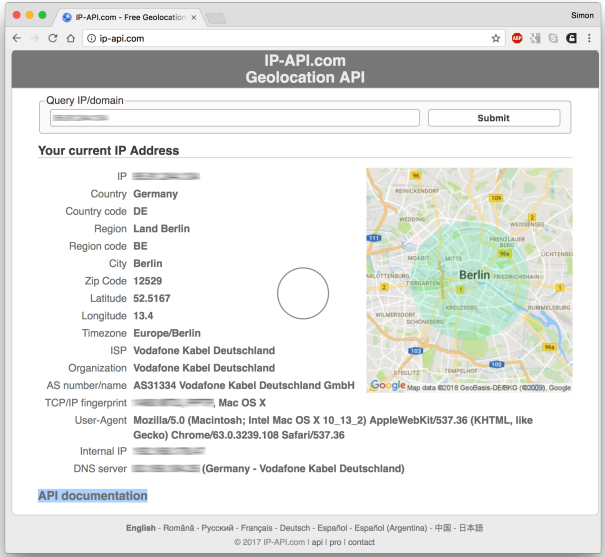
1. figure out how the API works: every API has a human-readable documentation for developers
2. build access to the API via R
3. build functionality that processes the data output (e.g., turns it into R objects)

Example

Example

The IP API

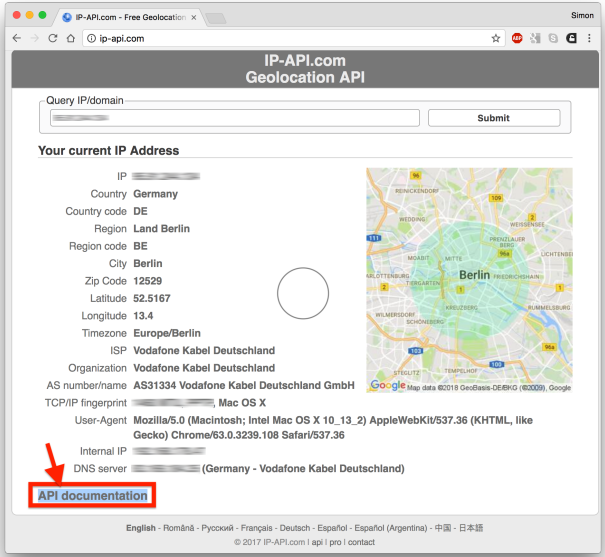
- available at http://ip-api.com/
- its purpose: parses your IP (or a bunch of given IPs) and returns some values, including guessed location, service provider, and organization behind the IP



Example

The IP API

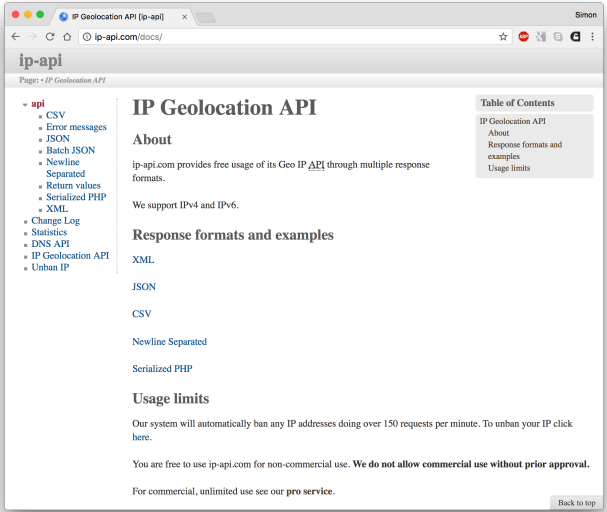
- available at http://ip-api.com/
- its purpose: parses your IP (or a bunch of given IPs) and returns some values, including guessed location service provider, and organization behind the IP
- check out API documentation



Example

The IP API

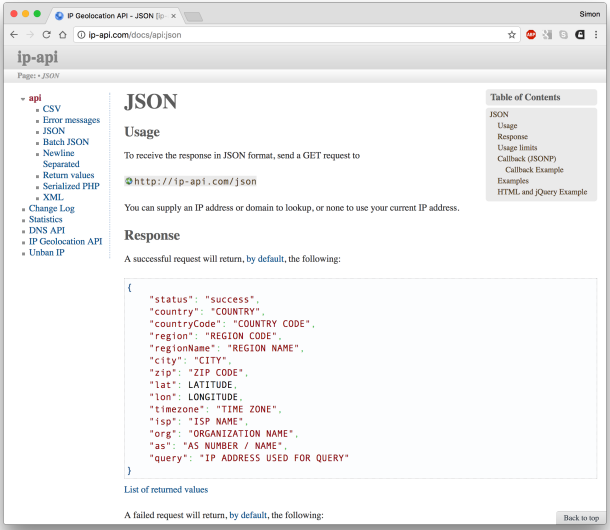
- access is free for non-commercial use (up to 150 requests per minute)
- various response formats available
- paid pro service available



Example

The IP API

- to access the API, we have to send a `GET` request to http://ip-api.com/json
- we can provide any IP we want
- the response is raw JSON code



Example

IP API access in R

A `GET` request essentially means that we assemble a URL using

- the API **endpoint** (a basic URL) and
- **parameters-value pairs** that are added to the URL
- here, we only add the literal IP address
- we use `fromJSON` to pose the request and directly parse the data from the API

R code

```
1 url <- "http://ip-api.com/json"
2 url_ip <- paste0(url, "/208.80.152.201")
3 ip_parsed <- jsonlite::fromJSON(url_ip)
```

end

Example

Investigating the output

R code

```
4  names(ip_parsed)
 [1] "as"          "city"        "country"     "countryCode" "isp"
 [6] "lat"         "lon"         "org"         "query"       "region"
[11] "regionName"  "status"      "timezone"    "zip"
5  ip_parsed
$as
[1] "AS14907 Wikimedia Foundation, Inc."

$city
[1] "Cleveland"

$country
[1] "United States"

$countryCode
[1] "US"

$isp
```

Example

Bringing it into shape

- in our case, `fromJSON()` has created an R list object
- we turn it into a data frame

R code

```
6  ip_parsed %>% as.data.frame(stringsAsFactors = FALSE)
                                  as      city       country countryCode
1 AS14907 Wikimedia Foundation, Inc. Cleveland United States          US
                         isp     lat      lon                       org
1 Wikimedia Foundation, Inc. 41.4995 -81.6954 Wikimedia Foundation, Inc.
           query region regionName  status         timezone   zip
1 208.80.152.201     OH       Ohio success America/New_York 44192
```

end

Summary

Summary

- accessing APIs from scratch can be almost as easy as using a readily available R client
- often however, APIs are more complex (provide much more parameters and varations in data output)
- ideally, you build functions around your API calls to make it even more accessible