Using Web APIs in R

Amanda Gadrow

July 12, 2017

Overview

- Quick review of web API basics
- Tools for accessing APIs
- Lots of examples
- Practical application
- What next?

API Basics

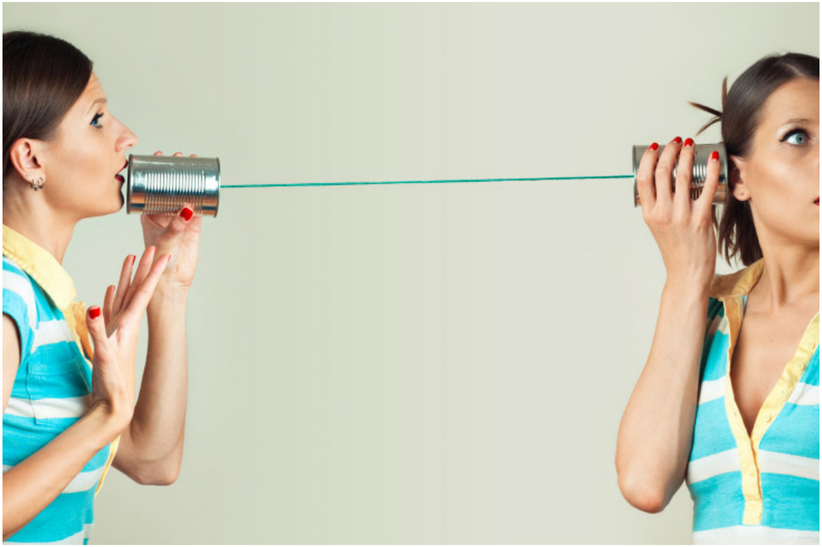
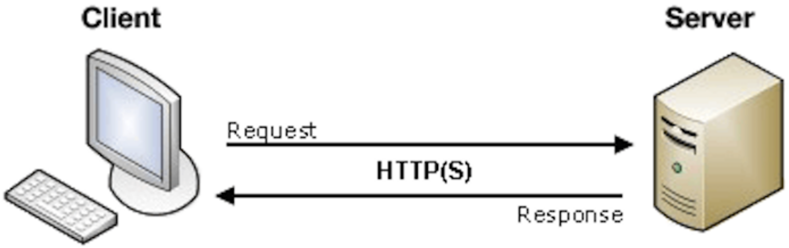


Figure 1:

Client-Server Communication



Request:

```
curl "http://www.omdbapi.com/?t=clue&r=json"
```

Response:

```
{"Title":"Clue","Year":"1985","Rated":"PG", ...}
```

API Resource Map

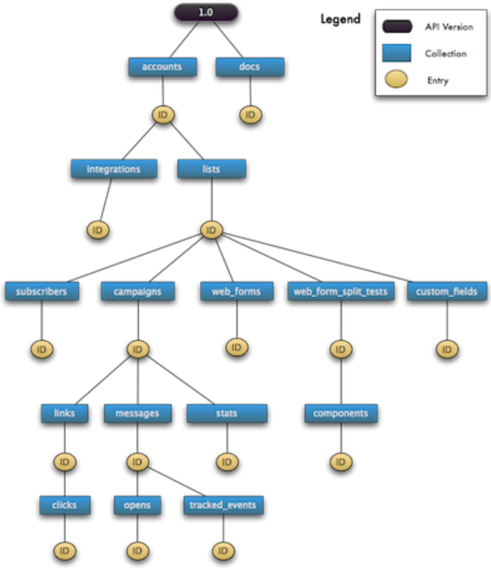
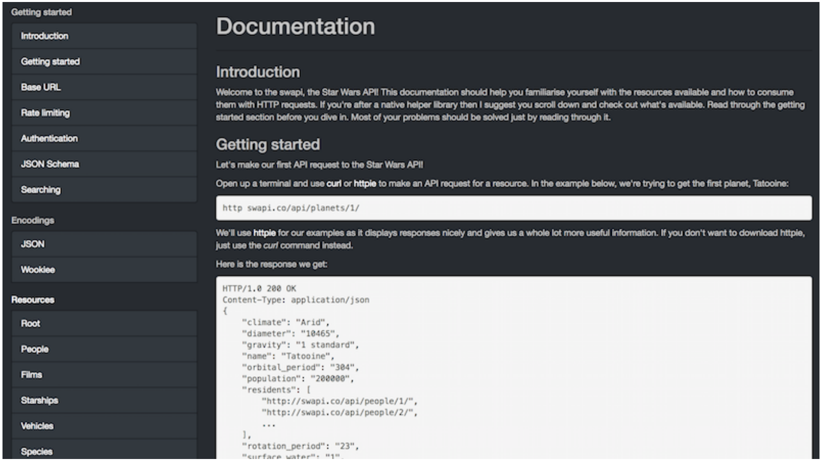


Figure 2:

API Documentation



Documentation

Introduction

Welcome to the swapi, the Star Wars API! This documentation should help you familiarise yourself with the resources available and how to consume them with HTTP requests. If you're after a native helper library then I suggest you scroll down and check out what's available. Read through the getting started section before you dive in. Most of your problems should be solved just by reading through it.

Getting started

Let's make our first API request to the Star Wars API!

Open up a terminal and use **curl** or **httpie** to make an API request for a resource. In the example below, we're trying to get the first planet, Tatooine:

```
http swapi.co/api/planets/1/
```

We'll use **httpie** for our examples as it displays responses nicely and gives us a whole lot more useful information. If you don't want to download httpie, just use the *curl* command instead.

Here is the response we get:

```
HTTP/1.0 200 OK
Content-Type: application/json
{
    "climate": "Arid",
    "diameter": "10465",
    "gravity": "1 standard",
    "name": "Tatooine",
    "orbital_period": "304",
    "population": "200000",
    "residents": [
        "http://swapi.co/api/people/1/",
        "http://swapi.co/api/people/2/",
        ...
    ],
    "rotation_period": "23",
```

Figure 3:

Documentation example: `https://swapi.co/documentation`

Tools for accessing web APIs from R

- packages that wrap API calls for a given service
  - aws.s3, RGoogleAnalytics, acs, etc.
- **httr** for making requests
- **jsonlite, xml2** for parsing the response

And once you have the data...

- **dplyr, tidyr, lubridate, stringr**, etc. for data wrangling

httr request functions

Wrap (and very conveniently match) HTTP verbs:

`HEAD()`

`GET()`

`POST()`

`PATCH()`

`PUT()`

`DELETE()`

httr request

To make a request, first load httr, then call GET() with a url:

```
library(httr)
r <- GET("http://swapi.co/api/planets/1")
```

This gives you a response object:

```
r
```

```
## Response [http://swapi.co/api/planets/1/]
##   Date: 2017-07-12 18:48
##   Status: 200
##   Content-Type: application/json
##   Size: 805 B
```

httr response

Use various helpers to dig into the response object:

```r
status_code(r)
```

```
## [1] 200
```

```r
headers(r)
```

```
## $date
## [1] "Wed, 12 Jul 2017 18:48:37 GMT"
##
## $`content-type`
## [1] "application/json"
##
## $`transfer-encoding`
## [1] "chunked"
##
## $connection
## [1] "keep-alive"
##
```

httr response body

And understand the content of the response:

```
str(content(r))
```

```
## List of 14
##  $ name           : chr "Tatooine"
##  $ rotation_period: chr "23"
##  $ orbital_period : chr "304"
##  $ diameter       : chr "10465"
##  $ climate        : chr "arid"
##  $ gravity        : chr "1 standard"
##  $ terrain        : chr "desert"
##  $ surface_water  : chr "1"
##  $ population     : chr "200000"
##  $ residents      :List of 10
##   ..$ : chr "http://swapi.co/api/people/1/"
##   ..$ : chr "http://swapi.co/api/people/2/"
##   ..$ : chr "http://swapi.co/api/people/4/"
##   ..$ : chr "http://swapi.co/api/people/6/"
```

http response components: status

```
http_status(r)
```

```
## $category
## [1] "Success"
##
## $reason
## [1] "OK"
##
## $message
## [1] "Success: (200) OK"
```

```
r$status_code
```

```
## [1] 200
```

You can automatically throw a warning or raise an error if a request did not succeed with `warn_for_status(r)` or `stop_for_status(r)`

http response components: headers

```
headers(r)

## $date
## [1] "Wed, 12 Jul 2017 18:48:37 GMT"
##
## $`content-type`
## [1] "application/json"
##
## $`transfer-encoding`
## [1] "chunked"
##
## $connection
## [1] "keep-alive"
##
## $vary
## [1] "Accept, Cookie"
##
## $allow
```

http response components: cookies

```
cookies(r)
```

```
##                 domain flag path secure          expirati
## 1 #HttpOnly_.swapi.co TRUE    /  FALSE 2018-07-12 14:48:
##                                                value
## 1 d9e82b9c9f9ea632561d6607b88dc08411499885317
```

http response components: body

```
content(r, "text")  # character vector

## [1]
"{\"name\":\"Tatooine\",\"rotation_period\":\"23\",
\"orbital_period\":\"304\",\"diameter\":\"10465\",\"climate
\"gravity\":\"1 standard\",\"terrain\":\"desert\",
\"surface_water\":\"1\",\"population\":\"200000\",
\"residents\":[\"http://swapi.co/api/people/1/\",
\"http://swapi.co/api/people/2/\",\"http://swapi.co/api/peo
\"http://swapi.co/api/people/6/\",\"http://swapi.co/api/peo
\"http://swapi.co/api/people/8/\",\"http://swapi.co/api/peo
\"http://swapi.co/api/people/11/\",\"http://swapi.co/api/pe
\"http://swapi.co/api/people/62/\"],\"films\":[\"http://swa
\"http://swapi.co/api/films/4/\",\"http://swapi.co/api/film
\"http://swapi.co/api/films/3/\",\"http://swapi.co/api/film
\"created\":\"2014-12-09T13:50:49.641000Z\",
\"edited\":\"2014-12-21T20:48:04.175778Z\",\"url\":\"http:/
```

http response components: body

```
content(r, "raw")  # raw vector, for non-text responses

##   [1] 7b 22 6e 61 6d 65 22 3a 22 54 61 74 6f 6f 69 6e 6
##  [24] 61 74 69 6f 6e 5f 70 65 72 69 6f 64 22 3a 22 32 3
##  [47] 69 74 61 6c 5f 70 65 72 69 6f 64 22 3a 22 33 30 3
##  [70] 6d 65 74 65 72 22 3a 22 31 30 34 36 35 22 2c 22 6
##  [93] 22 3a 22 61 72 69 64 22 2c 22 67 72 61 76 69 74 7
## [116] 74 61 6e 64 61 72 64 22 2c 22 74 65 72 72 61 69 6
## [139] 65 72 74 22 2c 22 73 75 72 66 61 63 65 5f 77 61 7
## [162] 22 2c 22 70 6f 70 75 6c 61 74 69 6f 6e 22 3a 22 3
## [185] 2c 22 72 65 73 69 64 65 6e 74 73 22 3a 5b 22 68 7
## [208] 77 61 70 69 2e 63 6f 2f 61 70 69 2f 70 65 6f 70 6
## [231] 22 68 74 74 70 3a 2f 2f 73 77 61 70 69 2e 63 6f 2
## [254] 6f 70 6c 65 2f 32 2f 22 2c 22 68 74 74 70 3a 2f 2
## [277] 63 6f 2f 61 70 69 2f 70 65 6f 70 6c 65 2f 34 2f 2
## [300] 3a 2f 2f 73 77 61 70 69 2e 63 6f 2f 61 70 69 2f 7
## [323] 36 2f 22 2c 22 68 74 74 70 3a 2f 2f 73 77 61 70 6
## [346] 69 2f 70 65 6f 70 6c 65 2f 37 2f 22 2c 22 68 74 7
```

http response components: body

```
content(r, "parsed")  # default parsers for common file typ

## $name
## [1] "Tatooine"
##
## $rotation_period
## [1] "23"
##
## $orbital_period
## [1] "304"
##
## $diameter
## [1] "10465"
##
## $climate
## [1] "arid"
##
## $gravity
```

Examples

SWAPI - `https://swapi.co/`

Twitter (OAuth1) - `https://api.twitter.com`

GitHub (OAuth2) - `https://api.github.com`

Other considerations

- API documentation, honesty, response format
- Paging
- Rate limiting/throttling
- Time to first byte (response time)
- Data storage
- Additional **httr** functionality
  - Authentication
    - `authenticate()`, `oauth1.0_token()`, `oauth2.0_token()`, `use_proxy()`
  - Request modifiers
    - `set_cookies()`, `add_headers()`, `content_type()`, `accept()`
  - Other
    - `stop_for_status()`, `warn_for_status()`, `timeout()`, `verbose()`, `parse_url()`, `progress()`

Practical application

Support Ticket Reports, in Two Parts

1. R script to get support ticket data via API
   - Pages through data and puts it all in a single dataframe
   - Filters, tidies, and flattens the raw data into a nice, easy-to-use rectangle
   - Saves the tidied data to an S3 bucket
   - Deployed to RStudio Connect; scheduled to poll the API and update the datastore automatically
2. Various applications and Rmd documents pull the data from S3
   - Visualizations for tickets by product, OS, priority, category/feature, etc.
   - We use this data to target product improvements, documentation enhancements, process improvements

Support Ticket Demo

Recap

- APIs are very useful sources of data, especially for datasets that are updated regularly.
- It's easy to pull API data into R – and keep it up-to-date – with the right packages and tools.
- Once you have the data in R, you can do all sorts of wonderful things with it.

What next?

- More on APIs and HTTP:
  - https://zapier.com/learn/apis/chapter-1-introduction-to-apis/
  - https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

- More on the tools:
  - `> help(package=httr)`
  - https://cran.r-project.org/web/packages/httr/vignettes/quickstart.html
  - http://github.com/hadley/httr/tree/master/demo
  - https://cran.r-project.org/web/packages/jsonlite/vignettes/json-apis.html
  - https://cran.r-project.org/web/packages/httr/vignettes/api-packages.html

- amanda@rstudio.com