Assignment V: GitHub and the ticketmaster.com API

Data Science Project Management (DS400) | Winter Term 2022/23

In this assignment, you will apply what you have learned about APIs and about version control with Git(Hub). First, you will acquire data about event venues using the API provided by ticketmaster.com. You will then use the geospatial data to visualize the extracted data on a map. Finally, you will repeat the same steps for a different country. It is further required that the entire project and its version history is documented in your personal GitHub repository.

Formal requirements

Submit your assignment earlier than **January 31, 12:00 noon**. The submission must include your code, results, and explanations for code and results, for both **Python** and **R**.

Submit your assignment via ILIAS. Upload the **two** `*.html`-files that were rendered from your *.Rmd and *.ipynb solutions. You can, optionally, upload the *.Rmd and *.ipynb files as well. You must use the following naming convention: `<Lastname>_<Firstname>_Assignment##_<Language>.<extension>` (e.g., `Lovelace_Ada_Assignment05_R.html`).

You will receive up to **10 points** upon passing this assignment.

Code of conduct

By submitting the assignment, you are acknowledging that the submitted assignment is your own work. This implies that ...

- the work you submit is your own
- the code you wrote is your own
- any comment or interpretation is in your own words
- you made clear whom you worked with

Disregarding this code of conduct will result in failure of the assignment!

Setting up a new GitHub repository

(1) Register on github.com in case you have not done this already.
(2) Initialize a new public repository for this assignment on GitHub.
(3) For the following exercises of this assignment, follow the standard Git workflow (i.e., pull the latest version of the project to your local computer, then stage, commit, and push all the modifications that you make throughout the project). Every logical programming step should be well documented on GitHub with a meaningful commit message, so that other people (e.g., your course instructor) can follow and understand the development history. You can do this either using Shell commands or a Git GUI of your choice.
(4) In the HTML file that you submit, include the hyperlink to the project repository (e.g., *https://github.com/yourUserName/yourProjectName*)

Getting to know the API

(5) Visit the documentation website for the API provided by ticketmaster.com (see here). Familiarize yourself with the features and functionalities of the Ticketmaster *Discovery API*. Have a particular look at rate limits.

(6) Whithin the scope of this assignment, you do **not** have to request your own API key. Instead retrieve a valid key from the API Explorer. This API key enables you to perform the `GET` requests needed throughout this assignment. Even though this API key is not secret *per se* (it is publicly visible on the API Explorer website), please comply to the common secrecy practices discussed in the lecture and the tutorial: Treat the API key as a **secret** token. Your API key should neither appear in the code that you are submitting nor in your public GitHub repository.

Interacting with the API - the basics

(7) Perform a first `GET` request, that searches for event venues in Germany (`countryCode = "DE"`). Extract the `content` from the `response` object and inspect the resulting list. Describe what you can see.

(8) Extract the `name`, the `city`, the `postalCode` and `address`, as well as the `url` and the `longitude` and `latitude` of the venues to a data frame. This data frame should have the following structure:

```
## Rows: 20
## Columns: 7
## $ name       <chr> "Gruenspan", "Huxleys Neue Welt", "Kleine Olympiahalle", "Z~
## $ city       <chr> "Hamburg", "Berlin", "Munich", "Emmelshausen", "Mülheim", "~
## $ postalCode <dbl> 22767, 10967, 80809, 56281, 45479, 76646, 68766, 44263, 542~
## $ address    <chr> "Grosse Freiheit 58", "Hasenheide 107 – 113", "Spiridon-Lou~
## $ url        <chr> "http://www.ticketmaster.de/venue/287155", "http://www.tick~
## $ longitude  <dbl> 9.958075, 13.421380, 11.550920, 7.556560, 6.874710, 8.59908~
## $ latitude   <dbl> 53.55188, 52.48639, 48.17543, 50.15544, 51.42778, 49.12692,~
```

Interacting with the API - advanced

(9) Have a closer look at the list element named `page`. Did your `GET` request from exercise (7) return *all* event locations in Germany? Obviously not - there are of course much more venues in Germany than those contained in this list. Your `GET` request only yielded the first results page containing the first 20 out of several thousands of venues. Check the API documentation under the section *Venue Search*. How can you request the venues from the remaining results pages? Iterate over the results pages and perform `GET` requests for *all* venues in Germany. After each iteration, extract the seven variables `name`, `city`, `postalCode`, `address`, `url`, `longitude`, and `latitude`. Join the information in one large data frame. Print the first 10 rows and the shape of the resulting data frame. The resulting data frame should look something like this (note that the exact number of search results may have changed since this document has been last modified):

```
## Rows: 12,671
## Columns: 7
## $ name       <chr> "Gruenspan", "Huxleys Neue Welt", "Kleine Olympiahalle", "Z~
## $ city       <chr> "Hamburg", "Berlin", "Munich", "Emmelshausen", "Mülheim", "~
## $ postalCode <dbl> 22767, 10967, 80809, 56281, 45479, 76646, 68766, 44263, 542~
## $ address    <chr> "Grosse Freiheit 58", "Hasenheide 107 – 113", "Spiridon-Lou~
## $ url        <chr> "http://www.ticketmaster.de/venue/287155", "http://www.tick~
## $ longitude  <dbl> 9.958075, 13.421380, 11.550920, 7.556560, 6.874710, 8.59908~
## $ latitude   <dbl> 53.55188, 52.48639, 48.17543, 50.15544, 51.42778, 49.12692,~
```

Visualizing the extracted data

(10) Below, you can find code that produces a map of Germany. Add points to the map indicating the locations of the event venues across Germany.

(11) You will find that some coordinates lie way beyond the German borders and can be assumed to be faulty. Set coordinate values to `NA` where the value of `longitude` is outside the range (`5.866, 15.042`) or where the value of `latitude` is outside the range (`47.270, 55.059`) (these coordinate ranges have been derived from the extreme points of Germany as listed on Wikipedia (see here). For extreme points of other countries, see here).

```r
# R
ggplot() +
  borders("world", "Germany", colour = "black", fill = "grey90") +
  theme_void() +
  coord_quickmap() +
  labs(title = "Event locations across Germany",
       caption = "Source: ticketmaster.com") +
  theme(title = element_text(size=8, face='bold'),
        plot.caption = element_text(face = "italic"))
```

```python
# Python
import geopandas as gpd

# get a base map of Germany
map = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))
map = map[map.name == "Germany"]

# plot the map
map.plot(ax = ax)
```

Event locations in other countries

(12) Repeat exercises (9)–(11) for another European country of your choice. (Hint: Clean code pays off! If you have coded the exercises efficiently, only very few adaptions need to be made.)