Problem Set 1

Web scraping the Apple App Store

Due 3-10-2025, by midnight

Assignment

The assignment consists in writing a Python code to scrape data, assembling a dataset and perform some basic data manipulations. When you are done, send your Python code and a pdf file describing the table and figure that you will produce (keep this short, 1 page maximum) to Javier Lopez at javier.lopezalfaro@phd.unibocconi.it. One email per group of student is enough, but indicate clearly the names of all group members. Please store the dataset produced: in Problem Set 2, you will have to use the data produced in this first assignment to perform demand estimation.

Web scraping

The goal of the Python code is to create a database containing the

- Rank
- Developer
- Price
- Number of reviews
- Genre
- Average score
- Number of downloads

First step

The list of the top 500 paid apps on Apple App Store in the US is available HERE. From this page, the scraper should obtain: The rank of the app, The name of the app and The ID of the app. The collected data must be organized in a Python data frame using the Pandas package (the standard to deal with datasets in Python).

This is the real scraping part of the exercise. Look at the tips section for some advises.

Second step

To obtain the remaining info, namely the price, the genre, the average score, the number of reviews and the approximate number of downloads, a ready-to-use package, `itunes-app-scraper` can be used. The package needs the ID of the app, obtained in the previous step as input, and returns all the necessary information in the form of a Python dictionary.

Finally, the data obtained has to be merged with the previously created Pandas dataframe.

Data manipulation

Now you have to generate a dataset suitable to perform a logit and nested logit demand estimation. For the demand estimation exercise, the dataset obtained in the first part lack some important characteristics:

- The quantity, as the number of downloads
- An outside good

To obtain the actual number of downloads you can use the (very approximate) assumption that apps have roughly 20 downloads per review. The number of downloads computed in this way has to be checked with the rank of the app, as the rank is a good proxy for the number of downloads.

In order to generate the outside good, you can focus on the genre of the apps. The genres can be aggregated in a smaller number of groups (around 10), and genres that can't be aggregated and include only a limited number of apps (let's say less than 10) can composed the outside good. Add these newly created variables to your dataset containing all the info about the apps, that should now be ready for the demand estimation exercise, and save it as a csv file. Your last task is to produce a nicely looking table of summary statistics, providing the mean, median, SD, min, max and number of observation of each variable in the dataset. Then create a scatterplot showing together the price and quantity variables. Briefly comment on your table and figure (no more than 1 page overall!)

Bouns point

There are other alternatives coming from the literature to estimate the number of downloads. You can try to implement another method. Just remember to explain it in the pdf file you will send and to provide the estimation in the dataset.

Tips

In this section there are some advises and tips on how to use Python to perform the required web scraping. If you are already familiar with Python and want to find your own way of performing the task it's perfectly fine, you don't need to follow them.

Parsing HTML tables: To ensure the correct parsing of the table containing the top 500 paid apps, you will need to process the content of the website's response using the `html.unescape()` method. Once the table is properly parsed, you can utilize the Beautifulsoup package to extract its content by searching for the `tr` (table row) `td` (table cells) tags. By looping over these tags you can extract all column values for each row, recreating the table in a DataFrame.

IDs for Apple App Store: The `itunes-app-scraper` is a very useful package: you feed it with the Apple App Store ID of the app and it automatically returns a number of information. The App IDs are provided in the App id column. Please note that some of the IDs are outdated, and the package will return an error. You can skip these apps and move on to the next one.

Documentation for the `itunes-app-scraper`: Note that there is no Documentation for this package, but the code is relatively simple and you can look in the `scraper.py` file to see what methods are available and what their parameters are. The file can be found at their GitHub repository, available HERE. Also popular LLMs are quite accurate in building documentation if you provide them the code.