Development Notes

2023-07-18

## Introduction

These are the development notes for scraping data from [Top Universities](www.topuniversities.com) and [Times Higher Education](www.timeshighereducation.com). Since, both websites present data as dynamic content they require the use of a headless browser to simulate the user clicking through the sites to generate the content.

The best tool for that kind of scraping in the R ecosystem is [RSelenium](https://docs.ropensci.org/RSelenium/).

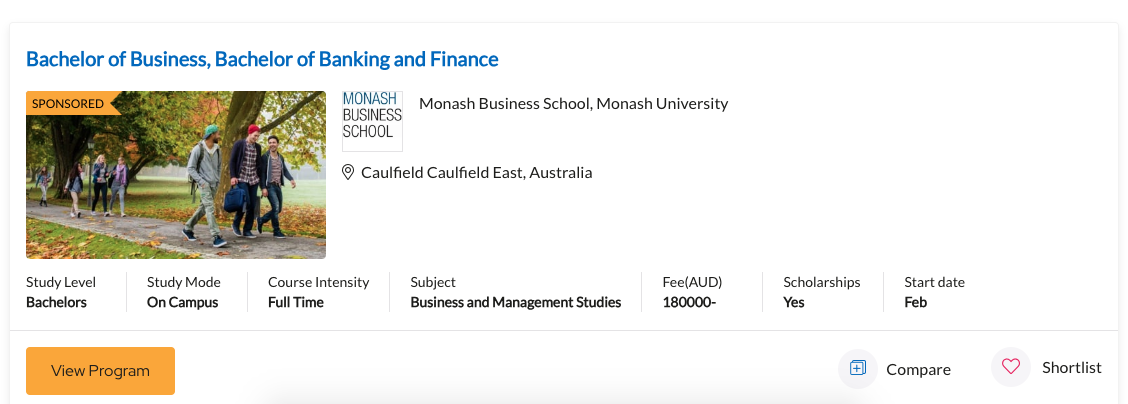
## Installing RSelenium

RSelenium requires the Rjava package. Rjava, in turn, requires access to the Java runtime. I recommend installing [Azul Java](https://www.azul.com/downloads/#zulu). Instructions vary with OS.

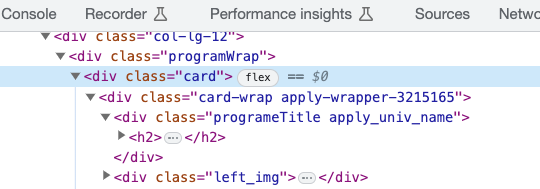
We also need a browser that works with RSelenium. The provided scripts work with Chrome.

## Scraping Top Universities

The topuniversities.com websites provides a [programs finder](https://www.topuniversities.com/programs?qs_qp=topnav). We used this tool to extract information for programs related to Bioinformatics, Machine Learning and Data Science. This is how the website presents the data:



To scrap the data, it is necessary to understand the structure of the generated markup in each page. In this case, the data is presented inside a div element with a card class:



The findElements method finds all elements with a class of card. The helper function extract\_row is applied to all found cards to extract the required data using css selectors.

To start scraping the website, we use RSelenium’s rsDriver to connect to a browser:

Once the connection with the browser is established, we navigate to the starting url.

This url encodes a starting query of the programs we are interested in, filtering programs with the subjects of Biological Sciences, Computer Science and Information Systems, Data Science, Genetics, Mathematics, Medicine Related Studies, Pharmacology, Pharmacy and Pharmacology, Physics and Astronomy, Statistics and Operational Research in Bahrain, Cyprus, Egypt, Iran, Iraq, Israel, Jordan, Kuwait, Lebanon, Oman, Palestine, Qatar, Saudi Arabia, Syria, Turkey, United Arab Emirates (UAE) and Yemen. The website presents the data in pages of 25 elements:



The scripts queries one page at a time and downloads the data for each. The function scrap\_page(i) scraps page i from this query. The 25 results from each page are saved to the temp folder.

Since the requirements information for each program are not available in this page, we save a link to the program’s page to scrap the data from requirements and tuition in a second pass.

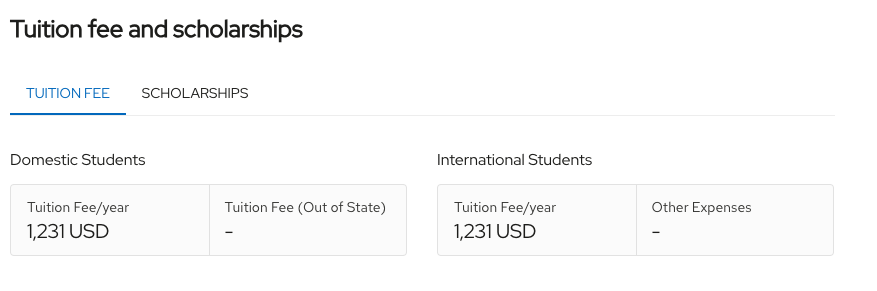
After scraping all the pages, we merge all the programs into a single data frame and save it:

### Scraping the requirements and tuitions data

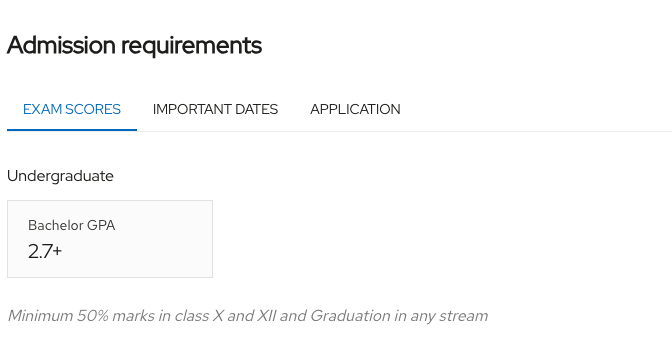
The script scrap\_programs\_tuition\_fees.R takes the list of urls scraped previously and downloads the tuition fee and requirements data from the program pages.

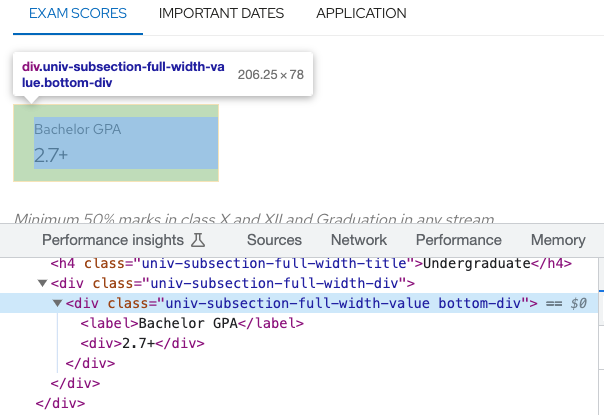
Since these are static html pages, it is not necessary to use a browser to access this data. The html pages are downloaded with rvest::read\_html.

This is how the webpages presents the tuition fee data:



And the requirements:

 The associated markup:



To extract the necessary information, we download the html page using read\_html, and then we specify css selectors that have the information we need to the html\_elements.

### Cleaning up the data

After scraping the data from all the programs, it is necessary to clean it by removing duplicates, parsing the requirements information and converting tuition fees to the same currency (British Pounds), removing programs that are not related to Machine Learning and Data Science.

Parsing each programs requirements involves taking the string from the html webpage and converting it to columns in our dataset. Each column corresponds to a requirement (TOEFL, IELTS, etc.). The value for each program in the column represents the minimum requirement of the program.

Currencies reported for the tuition fees are reportes in Euro, US Dollars, British Pounds and Singapore Dollars. I used the following exchange rates to convert currencies to British Pounds:

| Currency | Exchange |
| --- | --- |
| USD | 1.30 |
| EUR | 1.16 |
| SGD | 1.72 |

The final dataset is located in '../data\_output/programs\_geolocated.rds' and has 5356 programs. Here are the basic stats from the dataset:

Data summary

|  |  |
| --- | --- |
| Name | df |
| Number of rows | 5356 |
| Number of columns | 37 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 13 |
| numeric | 24 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

| skim\_variable | n\_missing | complete\_rate | empty | n\_unique |
| --- | --- | --- | --- | --- |
| university | 0 | 1.00 | 0 | 149 |
| university\_link | 0 | 1.00 | 0 | 154 |
| requirements | 0 | 1.00 | 0 | 1837 |
| study\_level | 0 | 1.00 | 0 | 3 |
| location | 0 | 1.00 | 1231 | 187 |
| program\_title | 0 | 1.00 | 0 | 4618 |
| program\_link | 0 | 1.00 | 0 | 5353 |
| subject | 0 | 1.00 | 0 | 39 |
| study\_mode | 3776 | 0.29 | 0 | 3 |
| course\_intensity | 2704 | 0.50 | 0 | 2 |
| duration | 825 | 0.85 | 0 | 31 |
| currency | 2851 | 0.47 | 0 | 4 |
| duration\_units | 825 | 0.85 | 0 | 1 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd |
| --- | --- | --- | --- | --- |
| fee | 2851 | 0.47 | 20964.14 | 14131.95 |
| duration\_length | 825 | 0.85 | 36.63 | 16.64 |
| fee\_gbp | 2851 | 0.47 | 21388.79 | 15638.57 |
| toefl | 2001 | 0.63 | 85.44 | 19.90 |
| ielts | 23 | 1.00 | 6.15 | 2.48 |
| bachelor\_gpa | 4054 | 0.24 | 2.61 | 1.95 |
| cambridge\_cae\_advanced | 3108 | 0.42 | 171.91 | 18.37 |
| pte\_academic | 2570 | 0.52 | 58.87 | 6.79 |
| a\_levels | 3488 | 0.35 | 9.14 | 28.51 |
| international\_baccalaureate | 3010 | 0.44 | 29.91 | 23.37 |
| ucas\_tariff | 4083 | 0.24 | 94.48 | 38.40 |
| atar | 4817 | 0.10 | 85.70 | 7.70 |
| sat | 4175 | 0.22 | 1058.66 | 375.89 |
| act | 5238 | 0.02 | 25.18 | 1.39 |
| gre | 5343 | 0.00 | 234.62 | 78.09 |
| gpa | 5318 | 0.01 | 2.80 | 0.37 |
| btec\_qualifications | 4236 | 0.21 | 4.17 | 19.76 |
| op | 5292 | 0.01 | 18.81 | 6.29 |
| sqa\_highers | 4991 | 0.07 | 17.75 | 38.96 |
| sqa\_advanced | 5271 | 0.02 | 1.41 | 13.02 |
| as\_levels | 5302 | 0.01 | 27.85 | 38.21 |
| year\_12\_scores | 5353 | 0.00 | 60.00 | 0.00 |
| latitude | 1644 | 0.69 | 52.17 | 4.30 |
| longitude | 1644 | 0.69 | -0.45 | 8.00 |

## Scraping timeshighereducation.com

The website also renders data dynamically, so we connect to it through a browser. The initial url is <https://www.timeshighereducation.com/world-university-rankings/2023/world-ranking#!/length/-1/sort_by/rank/sort_order/asc/cols/stats>

The data is presented in a single page, so it is easier to scrap. The script that achieves this is scrap\_rankings.R. It uses the same functions from Rselenium and another custom function to extract the data from the webpage.

The scraped data is located in data\_output/the\_ranking\_data.rds, and it includes 267 universities with its ranking and scores on the following categories: overall score, teaching, research, citations, industry income and international outlook.

Data summary

|  |  |
| --- | --- |
| Name | the\_ranking\_data |
| Number of rows | 267 |
| Number of columns | 12 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 9 |
| numeric | 3 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

| skim\_variable | n\_missing | complete\_rate | empty | n\_unique |
| --- | --- | --- | --- | --- |
| university | 0 | 1.0 | 0 | 163 |
| location | 0 | 1.0 | 49 | 216 |
| overall\_score | 28 | 0.9 | 0 | 34 |
| teaching | 28 | 0.9 | 0 | 86 |
| research | 28 | 0.9 | 0 | 97 |
| citations | 28 | 0.9 | 0 | 101 |
| industry\_income. | 28 | 0.9 | 0 | 101 |
| international\_outlook | 28 | 0.9 | 0 | 59 |
| rank | 28 | 0.9 | 0 | 34 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd |
| --- | --- | --- | --- | --- |
| longitude | 49 | 0.82 | 7.94 | 19.51 |
| latitude | 49 | 0.82 | 47.66 | 10.47 |
| n | 0 | 1.00 | 39.24 | 50.60 |

## Geocoding

To geocode the downloaded data, we used the [Google Maps API](https://developers.google.com/maps/) and the tidygeocoder [package](https://jessecambon.github.io/tidygeocoder/). The script that does this is geocode\_programs.R. The Google maps API requires an API key.

## Scripts

#### scrap\_times\_higher\_education.R

Downloads data from [THE](https://www.timeshighereducation.com/) and outputs ./data/ranks.rds and ./data/scores.rds.

#### scrap\_programs\_tuition\_fee.R

Takes ./data/programs.rds as input, downloads tuition and requirements data and outputs ./data/programs\_requirements\_fee.rds.

#### clean\_top\_universities.R

Takes ./data/programs\_requirements\_fee.rds as input, wrangles requirements data, converts tuition fees, filters programs not related to Bioinformatics, etc, recodes university names to match the names from the Times Higher Education website and merges the program data with the /data/geolocated\_locations.rds dataset.

#### geocode.R

Takes './data/programs\_clean.rds' as input adds coordinates and outputs ./data\_output/programs\_geolocated.rds.

### merge\_the\_topuniversities.R

Takes ./data/geolocated\_locations.rds, ranks.rds and scores.rds as input and outputs a single data frame with all the locations with coordinates and the rankings and scores of the corresponding universities.