Assignment 4 - Scraping Job Postings

Yifu Wu (916619585) [*]

12/02/2020

In this assignment, I am going to scrape the job postings related to the search terms ***data scientist, data analyst, statistician, and data engineer*** from the ***CyberCoders***. I will explain the approach I use to find the pattern of each job posts, the way to explore the next page and the method to explore the full posts. Then I will do some text analysis on what I get for each job posts.

1 Approach for Scraping Job Postings

To get job postings from the website, we need to first send queries to the web to tell what we want to search. We can find all useful argument from the "Query String Parameters" from the developer tools. For the CyberCoders, we need 5 different arguments in this query, which is shown in the code below. We need to use **getForm()** function to get all result forms from the website and use **htmlParse()** function to read the html document into an R object. For each queries, we only need to change the "searchterms" argument, we will use "data scientist" as an example.

```
CC_DS <- htmlParse(getForm("https://www.cybercoders.com/search/",
                           searchterms = "data scientist",
                           searchlocation = "",
                           newsearch = "true",
                           originalsearch = "true",
                           sorttype = ""))
```

Then we can deal with this web page in R. To get each job posts separately, we need to find the boundary of each jobs. We find a $\langle div \rangle$ element with a class attribute with the value of "job-listing-item". However, when we use

```
length(getNodeSet(CC_DS, "//div[@class = 'job-listing-item']"))
```

to explore the number of nodes we have in this page, we find that we get one more node compared with the result of an another $\langle div \rangle$ element with a class attribute with the value of "job-status", which is the child of the previous node.

```
length(getNodeSet(CC_DS, "//div[@class = 'job-status']"))
```

That is because there exists one advertisement which also starts with $\langle div\ class = "job-listing-item" \rangle$. So we need to find nodes for the $\langle div \rangle$ element with the class attribute with value of "job-listing-item" and with a child which the value of class attribute is "job-status".

[*]Department of Statistics, UC Davis and ifuwu@ucdavis.edu

```
getNodeSet(job, "//div[@class = 'job-listing-item'][.//div[@class = 'job-status']]")
```

After we get each job postings, we find that there are several useful information in the result page including job title, job location, job duration and salary. Although there also lists several preferred skills, we do not use those since we can get more skills in the full posting page. Since some jobs do not provide information of salary and duration, we will regard those as "Not Provided", and will deal with that later. The key point to get the correct information is to find the correct value of class attribute. The following codes shows that how to get those information.

```
title = xpathSApply(job, ".//div[@class = 'job-title']/a", xmlValue)
date = xpathSApply(job, ".//div[@class = 'posted']/text()", xmlValue)
location = xpathSApply(job, ".//div[@class = 'location']", xmlValue)
salary = xpathSApply(job, ".//div[@class = 'wage']/text()", xmlValue)
duration = xpathSApply(job, ".//div[@class = 'wage']/span", xmlValue)
```

The next step is to find the link of the full posting. We can find the attribute of the anchor tag (href) under the "job-title" class, which contains the URL of the full posting. However, we cannot directly use that since the URL is incomplete. So we need to use **getRelativeURL()** function to get the full URL.

```
link = as.character(getNodeSet(job, ".//div[@class = 'job-title']/a/@href"))
postURL = getRelativeURL(link, "https://www.cybercoders.com/search/")
```

The next step is to deal with the full posting page. In this page, we can get the header and body for each section, the full list of preferred skills, and also some free form texts describing the position. For very few jobs, there is no full job posting, so we regard those as "No Full Post". Since there are several sections in each post, so the body will be a list contains several elements. The following codes shows that how to get those information. We do not show the for loop to get each element in the body list, we just show how to get all body texts from the full post.

```
header = xpathSApply(post, "//h4[@class = 'section-title']", xmlValue)
preferred_skills = xpathSApply(post, "//span[@class = 'skill-name']", xmlValue)
othertext = xpathSApply(post, "//div[@class = 'section-data']", xmlValue)
body = getNodeSet(post, "//div[@class = 'section-data section-data-title']")
```

Then after finishing dealing with all jobs in the first page, we need to get the link to the next page of the results. We find an $\langle a \rangle$ tag which defines the hyperlink with the class attribute with value "get-search-results next". The attribute of the anchor tag (href) under this class shows the part of the URL of the next page. We also need to use the **getRelativeURL()** function to get the full URL.

```
nextpage = getNodeSet(page, "//a[@class = 'get-search-results next']/@href")
nextURL = getRelativeURL(as.character(nextpage), "https://www.cybercoders.com/search/")
```

Then I use a function to gather all jobs from all pages together into a list using while loop. Please see the **getAllPost()** function in the R script.

The next part is to modify the information we got in order that we can analysis them much easier.

The first one is the job duration. Since we know that in the result pages, some posts do not provide the information of the duration. But we find that the title or the text in full posting clearly indicate that it is a contract job. Thus, we use regular expression to find the word "Contract", if we find it, we regard this job as a contract job. There also exists very few jobs that can find word "Contract" in the posting, but the website shows that it is a full-time job. After exploring these jobs, we find that the website may make some mistakes, and we also regard those as contract jobs. Other jobs are all regard as full-time job, since in the website, we can only find two types of job which are full-time, and contract. The code to find word "Contract" is shown below.

```
grep("\\bContract\\b", job[[i]])
```

The second one is the salary of the job. Same as the previous one, some jobs do not provide salary in result pages. So we also need the regular expression to identify salaries from the text of full postings. Since most companies provide the salary in similar format which is "$000k-$000k", so we can search such pattern in the text of full post. If we find the same pattern we can extract them as salary. It is possible that we still cannot find salary in the full posting page, we regard those as "Not Provided". The code to find the specific pattern of salary is shown below.

```
grep("\\$[0-9]+[kK]* *- *\\$*[0-9]+[kK]*", job[[i]]$fullpost)
```

The third one is the education level needed or preferred for each job. We can find those information from the "What You Need for this Position" section from the full posting page. Since there exists several different education levels such as BS, MS, PhD, etc., we need to find if those can be found in the text. If any of those is found, we extract this element. We also need to pay attention to those phases that contain the same word as education level, for example, MS Windows and MS Office. We need to remove those from the results we get. The education level of other jobs will be regard as "Not Provided". The code is shown below.

```
regex1 = "[Bb]achelor[']*s|\\bB[.]*S[.]*\\b|\\bB[.]*S[.]*c[.]*\\b
          |[Uu]ndergraduate|[Mm]aster[']*s|\\bM[.]*S[.]*\\b
          |M[.]*S[.]*c[.]*|[Pp]h[.]*[Dd][.]*|[Gg]raduate"

grep("MS [Ww]indows|MS [Oo]ffice|MS Exchange|MS Work|MS Stack", edu)
```

The last one is to find the required skills in the full posting page. We can also find those in the "What You Need for this Position" section. Since it is easy to find some key phases for preferred skills, we find those phases from the text and get the index of those lines that contain the phases. We can say that all texts before that line will be required skills. Another important point is to remove lines with education level which we get in previous part. The regular expression of phases of preferred skills is shown below

```
regex2 = "\\bPreferred\\b|\\bPLUS\\b|\\bBONUS\\b|Bonus[-]* *[Ss]kill
          |Bonus[-]* *[Pp]oint|Nice[-]* *to[-]* *[Hh]ave|NICE[-]* *TO[-]* *HAVE
          |Plus[-]* *[Ss]kill|Extra[-]* *[Cc]redit"
```

Then we will apply all functions to each jobs, and we will get a list of job postings for each different job.

2 Exploring the Resulting Data - Text Analysis

In this section, I will use different approaches including regular expression(text mining) and making plots to explore the data I got from the previous section. From the **length()** function we know that there are 16 postings for data scientist, 26 postings for data analyst and 138 posting for data engineer. In this case, we do not find any postings for statistician. The number of posting will be changed since there will be some new posts and outdated posts. The function used in this project will not be influenced by any changes.

Based on the data we get, we need to do some modifications in order that we can do analysis much easier. The first part is to get the average salary of different jobs. Since we know that all salaries we get have lower bound and upper bound and we need to split those.

```
lower = unlist(strsplit(gsub(" ", "", salary), " *- *"))[1]
upper = unlist(strsplit(gsub(" ", "", salary), " *- *"))[2]
```

After doing this, we will calculate the average salary by using the formula: 0.5 * (upper + lower)

```
avg = (as.numeric(gsub("\\$|[Kk]", "", upper))
       + as.numeric(gsub("\\$|[Kk]", "", lower))) * 0.5
```

This works for the salaries of both full-time and contract job.

The next part is to summarize the education level based on the current data. The regular expression we used in this part is rather similar with that in the previous section, I just split it into several parts. In this case, I set 5 different factor levels which are "Bachelors", "Bachelors or higher", "Masters or higher", "PhD" and "Not Provided". All terms are easy to understand.

The last part is to extract some useful information from the required skills. From the experience we know that most works related data science require Python, R and SQL. In this case, we want figure out the number of jobs that require either programming skills. So we will search phases from the required skills section we get.

```
grep("\\b[Pp]ython\\b|\\bR\\b|\\bSQL\\b", skills[[i]])
```

Then we apply those functions to the raw data, and we will get one dataframe for each job that contains variables we are going to explore.

We first explore the relationship between the average salary and education level. Here we only select job postings that provide salary and requirement of education level. Also we only explore full-time job since very few of job postings are contract job.

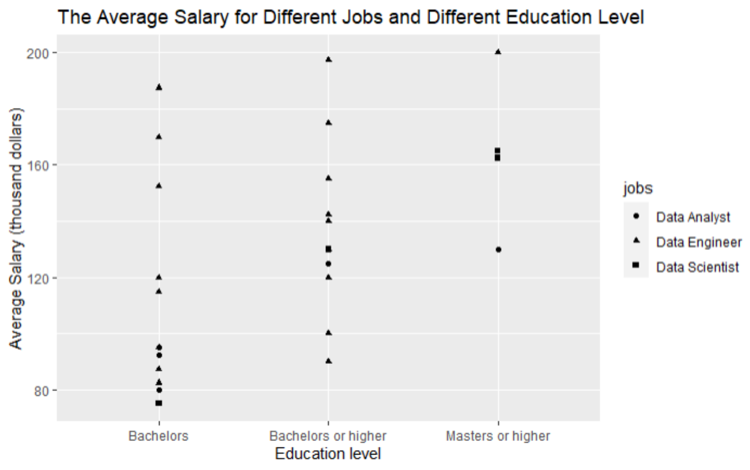

Figure 1: The Average Salary for Different Jobs and Different Education Level

From the first plot above we know that for all education level, data engineer always gets the highest average salary, and data analyst and data scientist usually get lower average salary. For bachelors and/or higher education level, the range of average salary is wider that those jobs that require masters or higher. The average starting salary of higher education level is much higher than those people only with bachelor degree.

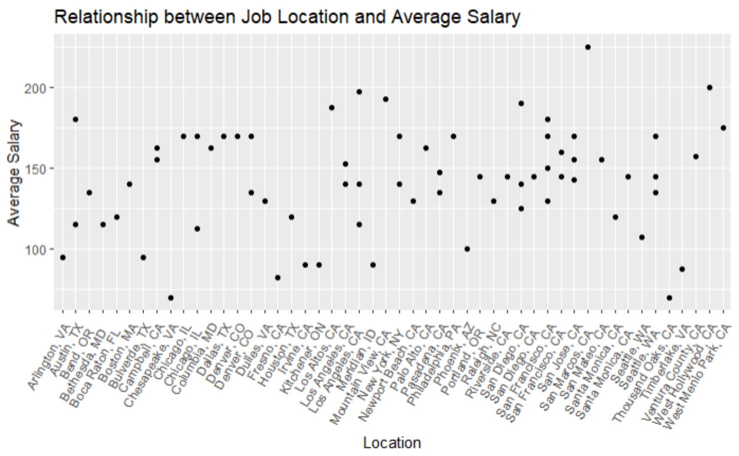

Figure 2: Relationship between Job Location and Average Salary

The second plot shows the relationship between average salary and job location. In this case, we still only explore those full-time jobs which provide salaries. We use data engineer as an example to show the relationship. From the plot we know that most jobs in California have higher average salary than other states. States like Virginia usually have much lower average salary compared with other states. We can also know that for jobs in the same state, the average salary has great difference.

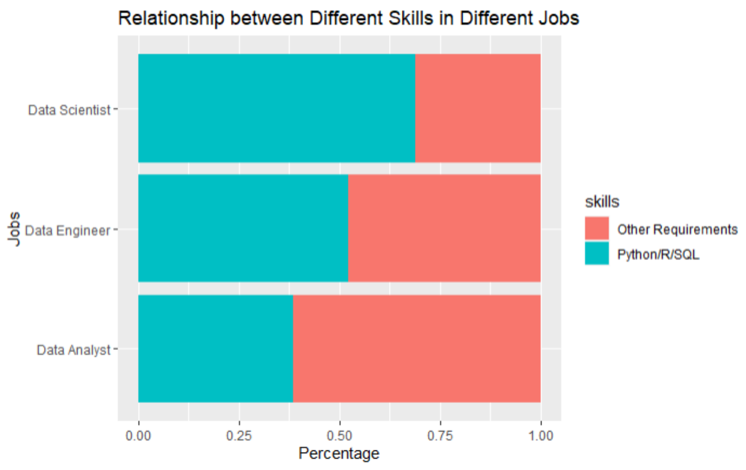

Figure 3: Relationship between Different Skills in Different Jobs

The third plot shows the relationship between different skills and different jobs. Since the number of postings for each job are different, we choose to use percentage plot to show the ratio between skills. From the plot we know that for most data scientist jobs, we are required to have experience on one or more programming language including Python, R and SQL. While for data engineer, about half of the postings require those skills. For data analyst, the percentage of jobs that require Python, R or SQL are much less than the data scientist jobs.

The following 3 plot shows the top 10 preferred skills for different jobs.

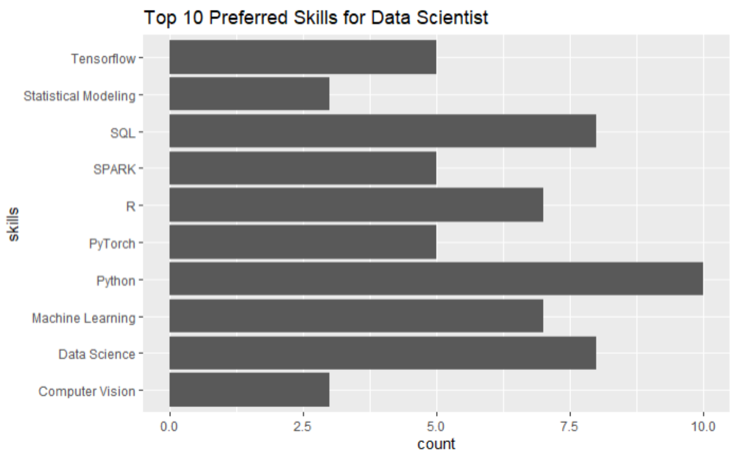

Figure 4: Top 10 Preferred Skills for Data Scientist

From the plot above we know that for data scientist, Python is still the top 1 preferred skill for this position. Other programming language like R and database language like SQL are also good to have. We also know that for this position, it is also good to know some Python libraries like Tensorflow and PyTorch. We can deduct that most work for data scientist are related to machine learning and computer vision.

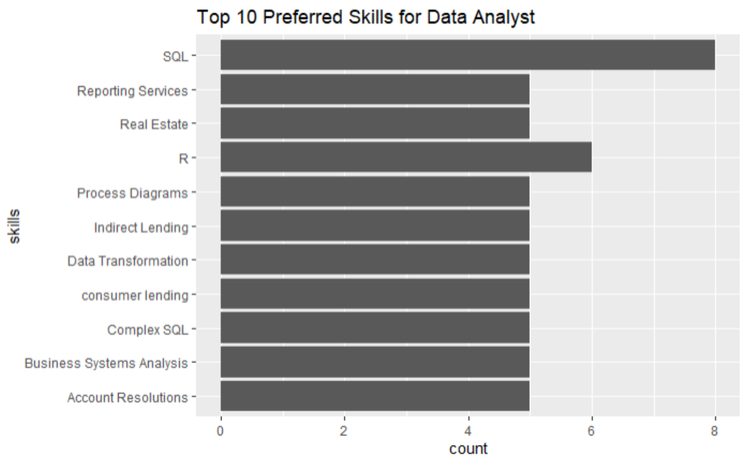

Figure 5: Top 10 Preferred Skills for Data Analyst

From the plot above we know that for data analyst, SQL is the top 1 preferred skill for this position, which means this job is highly related to database. We also know that R is preferred by many data analyst jobs, which indicates that we need to first use SQL to extract data from the database, and then use R to analyse it. Other skills like Business Systems Analysis and Real Estate are associate with financial field which requires other professional knowledge as well.

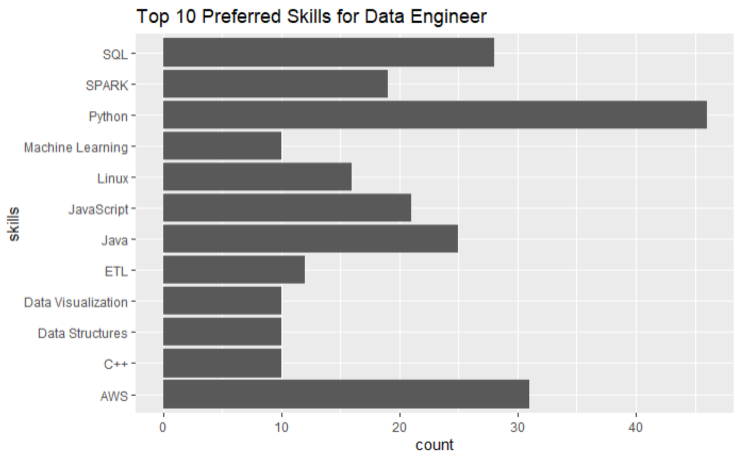

Figure 6: Top 10 Preferred Skills for Data Engineer

From the plot above we know that for data engineer, Python, SQL and AWS are top 3 preferred skills for this position. We can also find some other programming language and some HTML tools. Although these three job positions are all related with data, the preferred skills are not identical except few programming languages and database tools.