Report for Homework 4:

Web Page Scraping for CyberCoders

In this report I would introduce how I scrape job postings related to Data Analyst, Data Engineer and data science. From each job postings, I extracted information of required skills, preferred skills, salary, educational level, location, working hours information and position descriptions.

Get data

Get HTML form of the web page

The way I start web scraping is to read the first page of Data Analyst job postings. I used function "getForm()" to identify sections in the body of the HTTP request and get an HTML form of the web page. When searching for Data Analyst job postings in CyberCoders, the link I got is the following:

https://www.cybercoders.com/search/?searchterms=Data+Analyst&searchlocation=&newsearch=true&originalsearch=true&sorttype=

Therefore, with the function, I separated the HTTP request as the following:

```
getForm("https://www.cybercoders.com/search/?",page = 1, searchterms   =
        "Data Analyst", searchlocation = "", newsearch = "true",
        originalsearch = "true", sorttype = "")
```

You can notice that I have set "page" section equals to 1, which is not shown in the original link, but I have tried to add section "page = 1" in the link and it doesn't change the web page. The reason that I would like to add this section is because with the page number, I can set an index of page number to get information from any page. I will show the details of this step after I thoroughly scraped information in the 1st page.

Read HTML of job list pages

The next step is to read the HTML form with function "htmlParse()" and extract some useful information from the 1st page of the website. Without looking at the details of each job, only by reading the job list in the first page, we can find four useful information: job title, company location, wage and working hours, which are under the nodes of , , , respectively. Therefore, we can extract these information by using the function "getNodeSet()", which extract matched XML nodes from HTML, together with the function "xpathSApply" to simplify our matched result which are pure texts. For example, if we want to get the location information:

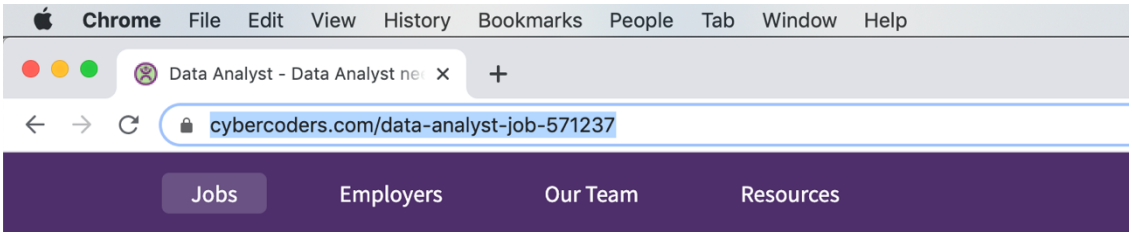
```
location = getNodeSet(doc, "//div[@class = 'location']")
Loc = xpathSApply(location[[1]], ".", xmlValue, trim = TRUE)
```

Where variable "doc" is the HTML form got by function "htmlParse()", and "location[[1]]" represent the first result of our matched node. Our result for the first company's location at the first page of the website would be represented by the function "Loc".

Read HTML of job detail pages

We can use the same method to extract the information mentioned above, but we will find there are some other information which can only be captured from the page of details of each job post, that is, the page showing more information as we click into a link of a job post. Therefore, our way to extract more information like as degree requirement and relevant skills should begin with finding the link of the detail page for each post. It is not hard to find, especially when we already known what the links of detail pages look like:

Graph 1: The link of detail pages example



Graph 2: What I found in the HTML file

```
314     <div class="job-details-container">
315         <div class="job-title">
316             <a href="/data-analyst-job-571237">Data Analyst - SQL, R, Python</a>
317         </div>
318         <div class="details">
319             <div class="location">Seattle, WA </div>
```

As we can see from Graph 2, to get the detail link of each post, I need to extract the highlighted URL from the "href" attribute, which is the child node of . Therefore, we use an Attribute Selector to select the attribute of "href":

```
    href = xpathSApply(doc, "//div[@class = 'job-title']//a/@href")
```

Note that what we got is not the entire hyperlink as what we see in Graph 1, but only the second half of it. We still need to paste it with "https://www.cybercoders.com":

```
    detaillink = paste("https://www.cybercoders.com",href[[1]], sep = "")
```

Then, we can read the link again and get the HTML form by using the function "readLines()" and "htmlParse()". After we get the HTML, we can repeat the steps mentioned above to get nodes that contains information of required skills, preferred skills, position descprition and degree. In the following content, I will briefly explain about my methods to find the information and mainly talk about the challenges I met when extracting them.

Extracting position description, required and preferred skills are relatively easy, because we can find nodes which exactly contain this information. Below is the function and Xpath I used in order to get nodes with information of description and skills:

```
<div class="section-data section-data-title" data-section="5"><!--START_SECTION_5-->You will develop
```

```
discrip = getNodeSet(detaildoc,
 "//div[@class = 'section-data section-  data-title'][@data-section = 5]")
```

```
<div class="section-data section-data-title" data-section="7"><!--START_SECTION_7-->You need at leas
```

```
skills = getNodeSet(detaildoc,
"//div[@class = 'section-data section-data-title'][@data-section = 7]")
```

```
<h4 class="section-title">Preferred Skills</h4>
<div class="skills">
    <ul class="skill-list">
            <li class="skill-item">
                <a href="/jobs/sql-skills/">
                    <span class="left-off"></span>
                    <span class="skill-name">SQL</span>
                    <span class="right"></span>
                </a>
            </li>
            <li class="skill-item">
                <a href="/jobs/r-skills/">
                    <span class="left-off"></span>
                    <span class="skill-name">R</span>
                    <span class="right"></span>
                </a>
            </li>
```

```
prefer = getNodeSet(detaildoc, "//span[@class = 'skill-name']")
```

Extracting degree information is a little bit tough, because it usually appears in the section of required skills instead of a unique section. Therefore, the idea is that we can extract the information of description as well as required and preferred skills first, and then use regular expression to extract degree information from them. I used "xpathSApply()" function to get the string content of skills and degrees, and then I split the string of skills information because I only want the piece of string with degree information. To get the piece, I used function "grep()" to extract any string that contains "degree" without case sensitivity. The function of "grep()" is shown below, where variable "frag" is the list of string pieces split by a Carriage return "\n".

```
degree = grep("degree", frag, value = TRUE, ignore.case = TRUE)
```

Read each job list in each page

Up to now we have extracted all the information we want from the first post in the first page. Now we need to create a loop in which we can extract all the information of Data Analyst from the web. What we need to do is just creating a nested loop, where we read each page in the outer loop and read each job post in the inner loop. I will not show details of this step since it is very basic.

However, as we try to store the information into a list while looping, we may get an error "subscript out of bounds" – note the result that function "getNodeSet()" return would be a node set which is a list. It will return an empty list if there is no matched node, and hence we cannot directly extract the result by indexing the list. To figure this out, I simply added

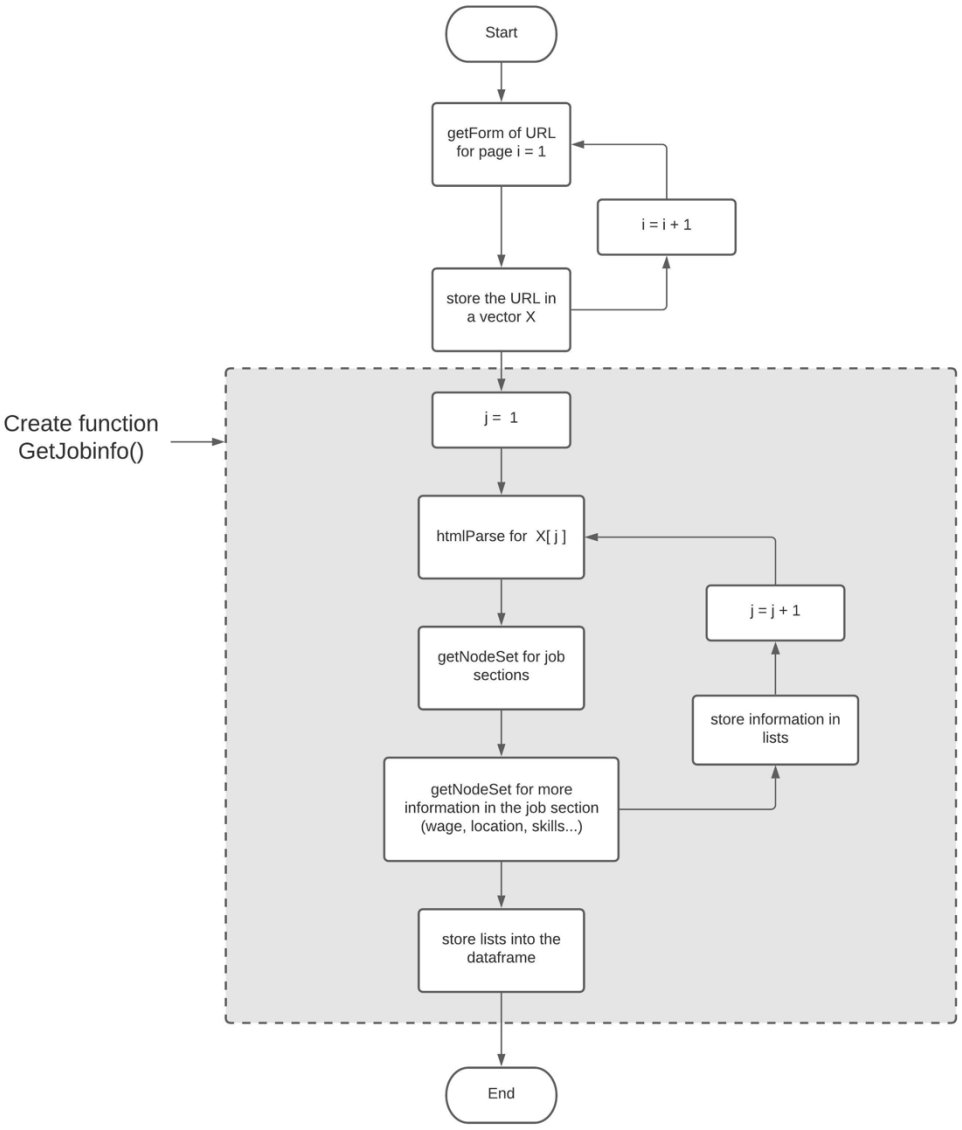
a string such as “no matched result” in empty lists when I find their length is zero:

```
   if (length(discrip) != 0){
           Discription = xpathSApply(discrip[[1]], ".", xmlValue,
             trim = TRUE)} else Discription = "No specific discription"
```

Create Function: GetJobinfo()

As we are trying to extract data of several different jobs, I would like to convert my work to a function and use this function to extract different job information of all pages.

My idea is to have a vector of URLs, and all its elements would be a link with different page numbers. My function is able to read through each URL so that it will get information from all pages. I created a flow chart to show the entire process of my codes, and the shaded area is the process of my function: GetJobinfo().



Explore Data

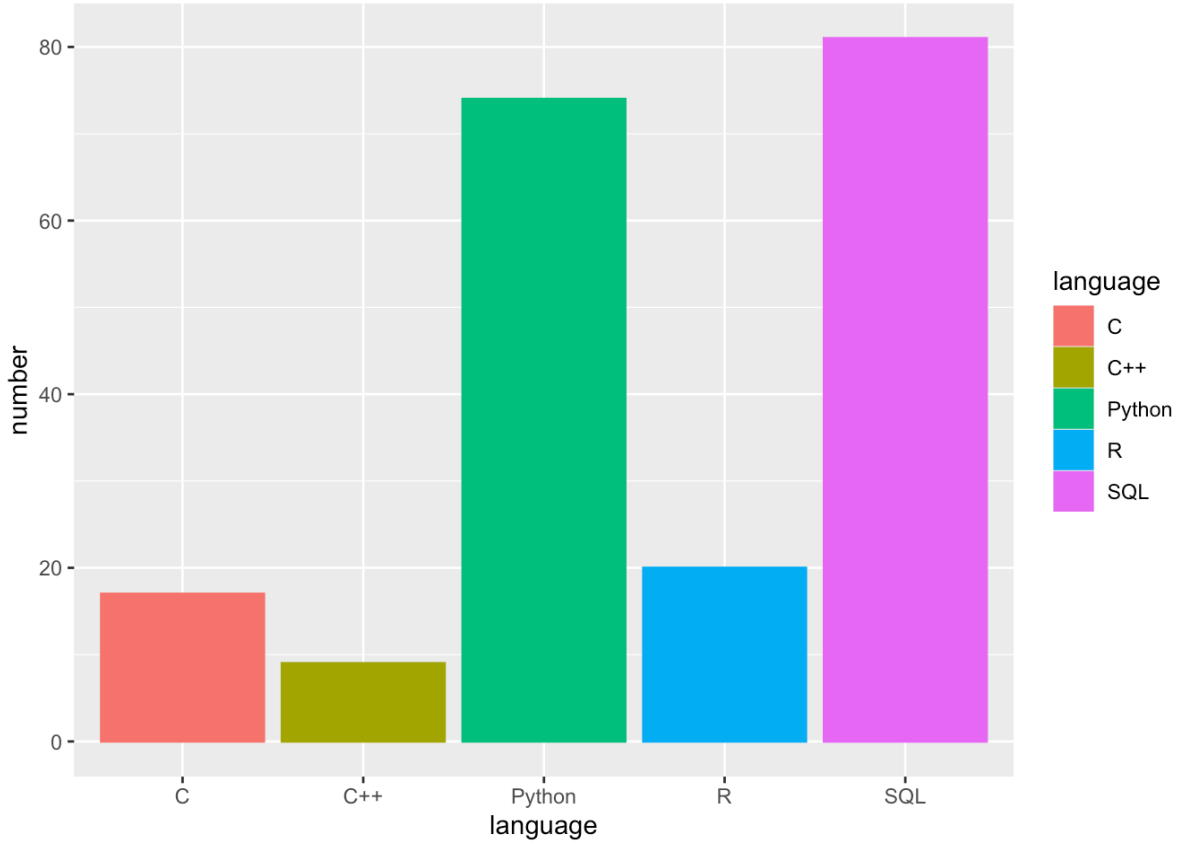
In this section, I would like to focus on two questions:

1. **What are the required skills in the field of statistics?**
2. **What are the salaries for each three type of job?**

We used regular expression and function "grepl()" to extract the logical result (TRUE or FALSE) of matched strings such as "R", "Python", "C++", etc. To be specific, the regular expression I used is shown below, which are used to extract information of R, Python, C++, C and SQL respectively.

```
num_r = grepl("\\bR\\b", Result$Skills)
num_python = grepl("python", Result$Skills, ignore.case = TRUE)
num_Css = grepl("C\\+\\+", Result$Skills, ignore.case = TRUE)
num_C = grepl("\\bC\\b", Result$Skills)
num_sql = grepl("sql", Result$Skills, ignore.case = TRUE
```

The I count the number of each existence of the computer language, we can see that Python and SQL are language that are frequently being asked, while C++ is the least frequently being asked.



Then we can focus on the second question by looking at the salaries list that we have extracted. Since the salaries information is not plain numbers but usually with dollar sign and other information, we need to extract the numbers by using "grep()" and "gusb()" functions. Also, since the salaries information are usually a range instead of a value, we extracted the highest and lowest possible salaries for all the jobs and stored them as two separate lists.

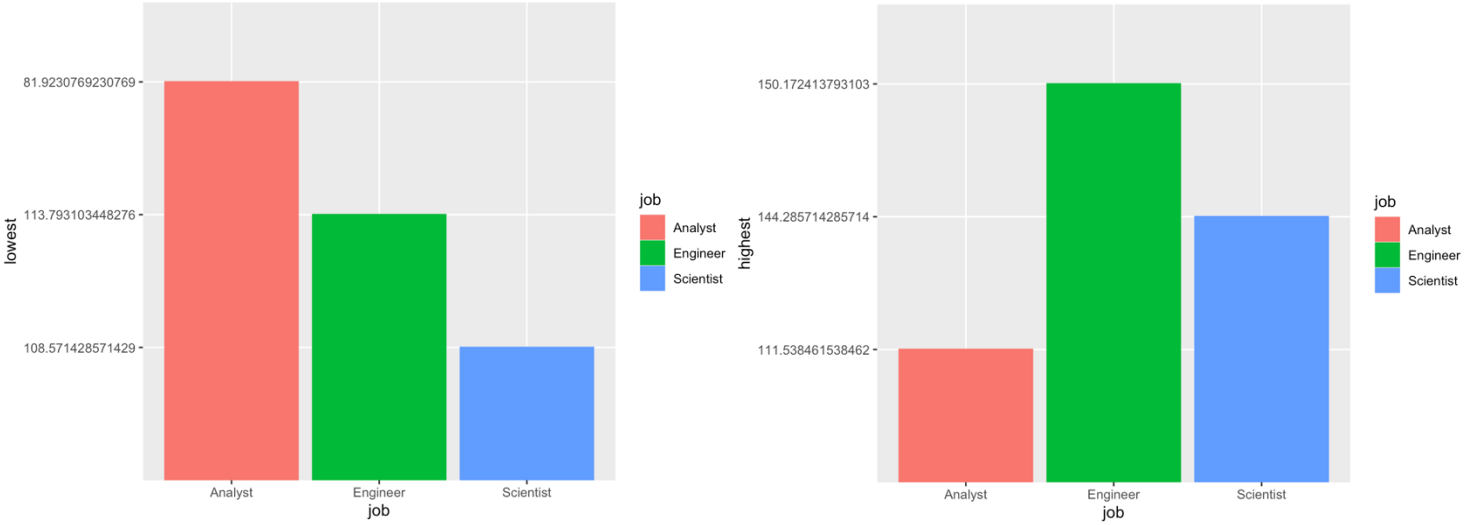
To be specific, the original salaries information is usually in the format of "Full-time $80k - $110k". As we know that the highest salary for this job would be $110k and the lowest would be $80k. What we want to get is the plain number 80 and 110 and store them into the lists of lowest salaries and highest salaries respectively. The way we extract the two numbers is as follows:

```
Sal = strsplit(as.vector(Result$Salary), "[[:punct:]]")[[i]]
k = grep("[0-9]", Sal, value = TRUE)
p = gsub("[a-zA-Z]", "", k)
lowersal[i] = as.numeric(p[1])
uppersal[i] = as.numeric(p[2])
```

The variable "Result" is our dataset with all extracted information. We firstly split the string of salary information, "Full-time $80k - $110k" as our example above, by punctuations. Therefore, we will get a list with separated strings "80k" and "110k". We can get these two strings by using the grep() function to extract strings with numbers. Then we can use function gsub() to substitute letters in strings with an empty string "". Finally, we can convert our string into numeric values and store them into lists. Of course, there are some jobs which haven't mentioned their salaries' information. We will get NA's in our "lowersal" and "uppersal" lists, hence we have to use na.omit() to avoid NA's.

As we have known the number of posts in each type of job, we can figure out the distributions of average lowest wage and average highest wage in different jobs. As we can see, the lower bound of salaries of Data Analyst is the highest among all types of jobs, while the higher bound of salaries of Data Engineer is higher than the other two's higher bounds.

The average lower bound and higher bound of salaries for the three jobs



PS: Since the dataset of job posts shown in R studio is not aesthetic, I have exported my dataset as an Excel. It has been also sent to canvas. Thank you for reading!

Web Scraping monster.com

I also did some preparation work for web scraping monster.com. I found a useful link for scraping this website from the "network", which is a in a json type. Therefore, I getForm of this link and used the function fromJSON() to get the structure of the web page:

```
tt = getForm("https://www.monster.com/jobs/search/pagination/?", q =
       "data-scientist", isDynamicPage = "true", isMKPagination =
       "true", page = pagenum)
u1 = fromJSON(tt)
```

Then I used the function str(u1[[1]]) to return the entire structure of the first job post in the first page (pagenum = 1) of job list for data scientists. It returns all the unstructured information that I need.

```
## List of 16
##  $ Title                  : chr "Data Scientist"
##  $ TitleLink              : chr "https://job-openings.monster.com/data-scientist-boston-ma-us-fidelity-talent
source/0224b774-292c-49b7-a43a-b66e0ab546cb"
##  $ DatePostedText         : chr "4 days ago"
##  $ DatePosted             : chr "2020-11-30T12:00"
##  $ LocationText           : chr "Boston, MA, 02210"
##  $ JobViewUrl             : chr "https://job-openings.monster.com/data-scientist-boston-ma-us-fidelity-talent
source/0224b774-292c-49b7-a43a-b66e0ab546cb"
##  $ ImpressionTracking     : chr "data-m_impr_uuid=\"4b264895-41fe-4b96-8f47-3124fd23199e\" data-m_impr_a_plac
ement_id=\"JSR2CW\" data-m_impr_s_t"| __truncated__
##  $ Company                :List of 6
##   ..$ Name             : chr "Fidelity TalentSource"
##   ..$ HasCompanyAddress: logi TRUE
##   ..$ LogoLink         : chr "https://media.newjobs.com/clu/xver/xveritudex/branding/18478/Fidelity-TalentSour
ce-LLC-logo-637267776772938686.jpg"
##   ..$ LogoImageUrl     : chr "https://media.newjobs.com/clu/xver/xveritudex/branding/18478/Fidelity-TalentSour
ce-LLC-logo-637267776772938686.jpg"
##   ..$ LogoLinkTitle    : chr "See all jobs openings for Fidelity TalentSource"
##   ..$ LogoTooltip      : chr "Fidelity TalentSource logo"
##  $ Text                   : chr "Data Scientist"
##  $ IsAggregated           : chr "true"
##  $ JobViewUrlMeta         : chr "https://job-openings.monster.com/data-scientist-boston-ma-us-fidelity-talent
source/0224b774-292c-49b7-a43a-b66e0ab546cb"
##  $ MusangKingId           : chr "0224b774-292c-49b7-a43a-b66e0ab546cb"
##  $ CompanyLogoUrl         : chr "https://media.newjobs.com/clu/xver/xveritudex/branding/18478/Fidelity-Talent
Source-LLC-logo-637267776772938686.jpg"
##  $ PrivateBoardIconImageUrl: chr ""
##  $ FitIcon                : chr ""
##  $ FitIconType            : chr ""
```

As we can see, it shows us the job title and location. More importantly, from the section "TitleLink" or "JobViewUrl", it gives us a link for the detail page of this job. We can go further and read this link, and then we can find that this link is a html type. Therefore, I used the function readLines() and htmlParse()to get the structure of the web page.

However, when I looked at the structure of this web, I have noticed that it would be a much more difficult work than web scraping CyberCoders.com, as its html form doesn't have a clear pattern or specific sections for information such as skills, degree requirements, position descriptions, etc. I tried to look for specific words in the document, for example, the string "position", to "grep" some information of position descriptions. However, we cannot guarantee that all the position descriptions are mentioned as "position descriptions" – I have also noticed that some are called "job responsibilities".

In a word, I have already found all information that are required in the homework, but it would be a big challenge to use a systematic method to extract information from all job posts. I may need a lot more time to find all the patterns of html forms and come up with methods to extract information from each of them.