STA141 B Web Scraping

Sitong Qian

12/1/2020

For this assignment, I first took a look at each job posting websites, and researched on each by using browser's(CHROME) defauly tool, View -> Developer -> View Sourceand View -> Developer -> Developer tools. After basic scraping, I found out websites cybercoder is the most organized and cleaned one compared with other job posting websites. Thus, I began on working with cybeercoder.

My very first approach is finding the website link and used getForm to looking for specific job titles, and have them returned to a readable HTML document to further dig for pattern.WHen having it returned to the html file, I found out, the job list usually have a begining of 'class = 'job-listing-item". Thus, I applied the getNodeSet on it. Now, I have a list of jobs that related to data industry.

I used keyword 'data' for search since I found cybercoder is a relatively botique websites, the job posts presented in this websites is relatively limited. Thus, to obtain more useful information, I choose to use the keywords 'data'.

```
library(XML)
library(RCurl)
library(stringr)
library(httr)
library(ggplot2)
link = 'https://www.cybercoders.com/search'
search = getForm(link,searchterms = 'Data' )    #using this form because this is how
websites used this 'Data'
document = htmlParse(search)
list = getNodeSet(document,"//div[@class = 'job-listing-item']")
```

The next steps is reading the content. While at first, I found that CyberCoders is a very neat and organized website, with almost all the information available without looking into individual job sub posts. So, based on just the content of page, I extracted job_title, salary, location, postDate, date, post description, preferred_skills,job_status.

When doing the above-mentioned information extraction, I found out salary and job_status are mixed in the wage section, so for specialization purposes, I separated these two, by using string_extract and string_split with different patterns (use what I learned from HW regular expressions).

Then I found out, the required skills are not listed on the page, so I still need to look inside each post, for which, I tried a lot to find the pattern for looking for the exact post, and have the half-completed href links, and pasted it with the initial links from websites.

I also added a section here, to determine if the links to the actual posts do not exist, I will have required_skills returns as NA,later I found out, the fifth posts on each page does not of real information. When extracting the required_skills, I found it's really strange that I can't use it by simply applying the method I used for extracting other information like job_title. Then, after multiple times of experiments, I found a pattern of ''//h4/..//div", and then, I looked into it, the third list returned what I want.

After all, I had all the content I need to accesss from one pages.

```
Post =
  function(ind)
  {
    title = unique(xpathSApply(ind, ".//div[@class = 'job-title']",xmlValue))
    job_title = trimws(gsub("\\r\\n","", title))  #have trimws to get rid of white sp
ace
    salary_jobStatus = trimws(unique(xpathSApply(ind, ".//div[@class = 'wage']",xmlVa
lue)))
    location = trimws(unique(xpathSApply(ind, ".//div[@class = 'location']",xmlValue)
)) #add trimws() to remove empty space in the leading or trailing spaces in a string
    postDate = xpathSApply(ind,".//div[@class = 'posted']",xmlValue)
    date = trimws(gsub("Posted",'',postDate))
    postDescription =  xpathSApply(ind,".//div[@class = 'description']",xmlValue)
    description = trimws(gsub("\\r\\n","", postDescription))
    preferred_skills = trimws(unique(xpathSApply(ind, ".//li[@class = 'skill-item']",
xmlValue)))

    link = 'https://www.cybercoders.com/search'
    title_link = getNodeSet(ind,".//a[contains(.,job_title)]/@href")[[1]] #get the li
nk which direct to the actual page

    if (length(title_link) > 0) {
      postURL = getRelativeURL(title_link,link)
      postcontent = htmlParse(getURLContent(getRelativeURL(title_link,link))) # get t
he link direct to the individual post, have get Relative because the link doesn't com
e with http
      required_skills_unorganized = (xpathSApply(postcontent,'//h4/..//div',xmlValue)
)[3] # noted the structure of the data and have it returned in
      required_skills_organized = str_split(required_skills_unorganized,'\n|\t') # ha
ve\t because on the second page, there is one post information seperated with \t
    } else {
      postURL = 'NA'
      postcontent = 'NA'
      required_skills_unorganized = 'NA'
      required_skills_organized = 'NA'
    }

    if (length(salary_jobStatus) >0) {
      if (salary_jobStatus == 'Compensation Unspecified') {
```

```
        salary = NA
        job_status = NA}
      else{
        salary = str_extract(salary_jobStatus,'\\$.*k')
        job_test = str_split(salary_jobStatus,'\\s')          # noticing the pattern
is a mixed of job-status and salary with blank connecting this two, stingsplit to ext
arct the pattern of job-status
        job_status = job_test[[1]][1]
      }
    } else{
      salary = NA                                             #retuen NA here for eas
iler comparison when graphing
      job_status = NA}

    list(title = job_title, salary = salary, job_status = job_status,date = date,loca
tion = location,job_description = description,preferred_skills = preferred_skills,req
uired_skills = required_skills_organized,url = postURL)
  }
```

Then, I am working on getting the next page. First, I investigated the pattern to read the next page. I found there are two patterns. 1) the first one is denoted as with ‘next’ pages, by clicking the corresponding button in the real websites, I will have the access to the next pages. 2) the second one has accessed all ten pages at the same time since there are numbers denoted in the button, that up to choose 10 different pages in total.

I choose to stick with the first method, I first locate the pattern that has ‘next’, and extracted it, since this href link is not in the complete form, I used getRelativeURL to get the real completed link.

Here, I met a difficulty that the link I got from getRelativeURL can’t be used. The pages direct to an HTML document that useless obtained information. To fix this, I observed the actual pattern of the links by directly click the next buttons on the web pages and used the regular expression to reshape the links. Also, I indicated that if the functions processed to the end pages of the search, It should return NULL.

```
linktest = 'https://www.cybercoders.com/search/?searchterms=data&searchlocation=&news
earch=true&originalsearch=true&sorttype='
search = getForm(linktest,searchterms = 'Data' )
document = htmlParse(search)

getNextPage =
function(doc)
{
    link = 'https://www.cybercoders.com/search/'
    nxt = getNodeSet(doc, "//li/..//a[@rel = 'next']/@href")
     if(length(nxt) == 0){
      return(NULL)
    }
     else{
      fixed_next_page_link = str_extract(nxt[[1]],"\\?.*=&") #used the regular expres
sion to fix the link here.
      getnextlink = getRelativeURL(fixed_next_page_link, link)
      return(getnextlink)}
}
```

Here, I am working on obtaining all the links by applying getNextPage functions on each page. Thus, I created an empty list in the function and appended the results I got from each page to this list. At the end, I will get a list with all the links directed to the next pages.

```
linktest = 'https://www.cybercoders.com/search/?searchterms=data&searchlocation=&news
earch=true&originalsearch=true&sorttype='
link = 'https://www.cybercoders.com/search/'

get_all_urls =
 function(link){

    search = getForm(link,searchterms = 'Data' )
    document = htmlParse(search)
    all_urls = list()
    next_url = getNextPage(document)
    all_urls = append(all_urls,link)
    all_urls = append(all_urls,next_url)

    while(length(next_url) != 0){
      next_page = getForm(next_url )
      next_document = htmlParse(next_page)
      next_url = getNextPage(next_document)
      all_urls = append(all_urls,next_url)
    }
    return (all_urls)
 }

all_links = get_all_urls(linktest)
```

Since, the input of my Post functions at the top, is the predivded nodeset of each job posts on the pages. Thus, I need to processed all the links I got above, go intoo each links and extracted the node for each job posts and have it append to a empty list with all job posts nodes.

```
pagecontentjoblist = function(links){
  linklength = length(links)
  all_pages_job_lists = list()
  for(i in 1:(linklength)){
    ind_link = links[[i]]
    searchlinks = getForm(ind_link,searchterms = 'Data' )
    documentlinks = htmlParse(searchlinks)
    all_job_lists = getNodeSet(documentlinks,"//div[@class = 'job-listing-item']")
    all_pages_job_lists = append(all_pages_job_lists,all_job_lists)
  }
    return(all_pages_job_lists)
  }
```

Finally, I am able to obtained all the posts related to data in the cybercoders.com

```
all_page_job_lists = pagecontentjoblist(all_links)
datapost_all = lapply(all_page_job_lists, Post)
```

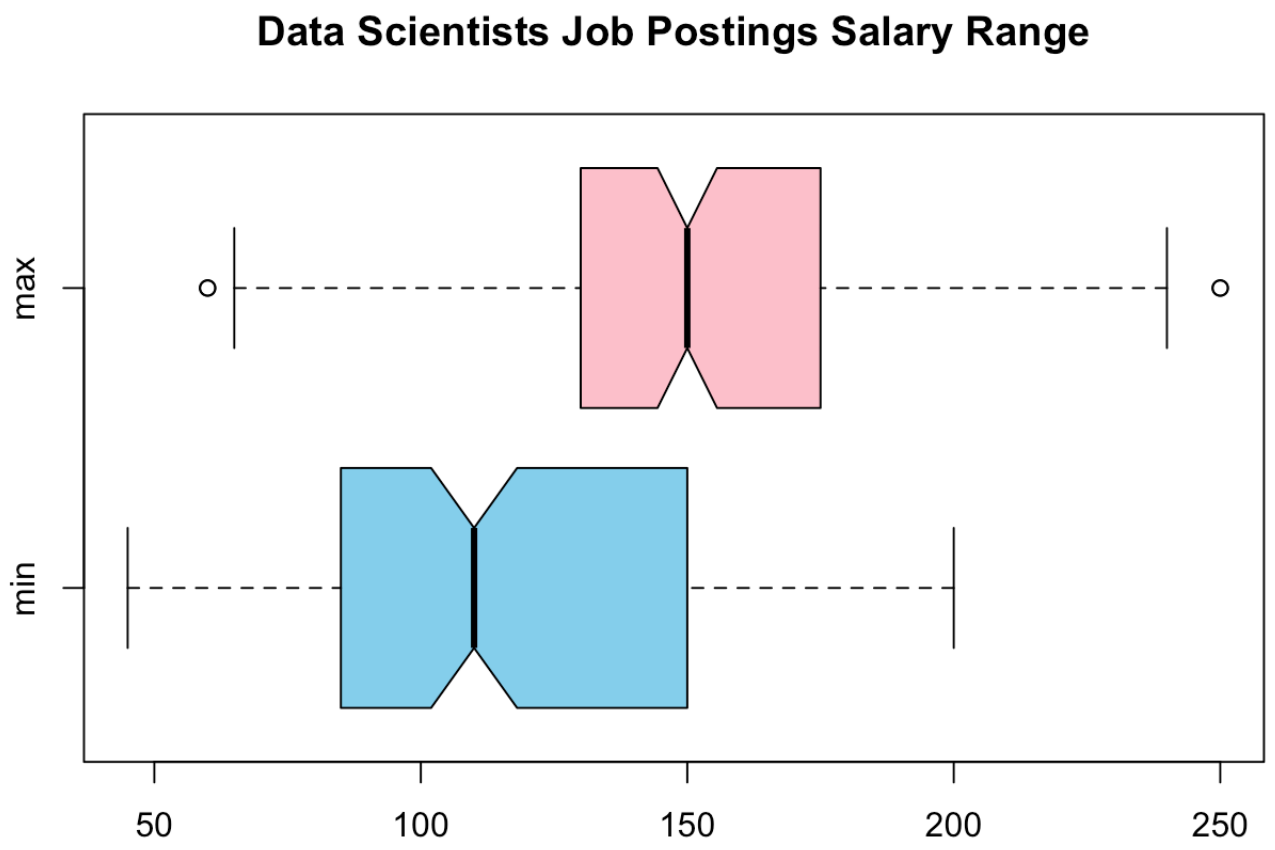
Then, I take the organized dataset and made some analysis based on it. Since I have the data arranged as a string, I need to first pull out or the number to make is plotable. Thus,I extract each salary information row, and have it organized in a dataframe with minmum wage and maximum wage (only including number).As it shown in the graph, the mean range of salary is about a minmum at 110k to 150k.

```
Salary <- sapply(1:length(datapost_all),function(i){
  salary<- datapost_all[[i]][["salary"]]

  return(salary)
  })

salary_na <- na.omit(Salary)

salary_min_max <- sapply(1:length(salary_na),function(i){
  salary <- salary_na[[i]]
  min <- str_split(salary,'-')[[1]][1]
  min_number <- regmatches(min, gregexpr("[[:digit:]]+", min))
  max <- str_split(salary,'-')[[1]][2]
  max_number <- regmatches(max, gregexpr("[[:digit:]]+", max))
  salary <- c(min_number,max_number)
  return(salary)
  })
salary_min_max <- as.data.frame(t(salary_min_max))
salary_min <- as.numeric(salary_min_max$V1)
salary_max <- as.numeric(salary_min_max$V2)
boxplot(salary_min,salary_max,
        main = 'Data Scientists Job Postings Salary Range',
        names = c('min','max'),
        col = c('skyblue','pink'),
        horizontal = TRUE,
        notch = TRUE
)
```



Next, I first, listsed out the requirments for jobs(took first 10) and I am interesting in one skills that is required by data related job posting, I count the keywords such as Python, R, SQL in the job skills. As it shown in the table, out of 315 posts, about nearly half of them required R, and the next popular program languages is SQL, then is Python, and the last one is JAVA.

```
library(plyr)
Required_skills <- sapply(1:length(datapost_all),function(i){
  required_skills <- datapost_all[[i]][["required_skills"]][[1]]
  return(required_skills)
  })
Required_skills[1:10]
```

```
## [[1]]
## [1] "- Data Engineer- Data Visualization" "- Data Warehousing"
## [3] "- ETL"                               "- SQL"
## [5] "- Data Integrity"                    "- Data Profiling"
## [7] "- Data Analysis"                     "- Data/ETL Solutions"
## [9] "- Shell scripts"
##
```

```
## [[2]]
## [1] "- Data Engineer- Data Visualization" "- Data Warehousing"
## [3] "- ETL"                               "- SQL"
## [5] "- Data Integrity"                    "- Data Profiling"
## [7] "- Data Analysis"                     "- Data/ETL Solutions"
## [9] "- Shell scripts"
##
## [[3]]
## [1] "- Data Engineer- Data Visualization" "- Data Warehousing"
## [3] "- ETL"                               "- SQL"
## [5] "- Data Integrity"                    "- Data Profiling"
## [7] "- Data Analysis"                     "- Data/ETL Solutions"
## [9] "- Shell scripts"
##
## [[4]]
## [1] "- Data Engineer- Data Visualization" "- Data Warehousing"
## [3] "- ETL"                               "- SQL"
## [5] "- Data Integrity"                    "- Data Profiling"
## [7] "- Data Analysis"                     "- Data/ETL Solutions"
## [9] "- Shell scripts"
##
## [[5]]
## [1] "NA"
##
## [[6]]
## [1] "- Data Engineer- Data Visualization" "- Data Warehousing"
## [3] "- ETL"                               "- SQL"
## [5] "- Data Integrity"                    "- Data Profiling"
## [7] "- Data Analysis"                     "- Data/ETL Solutions"
## [9] "- Shell scripts"
##
## [[7]]
## [1] "- Data Engineer- Data Visualization" "- Data Warehousing"
## [3] "- ETL"                               "- SQL"
## [5] "- Data Integrity"                    "- Data Profiling"
## [7] "- Data Analysis"                     "- Data/ETL Solutions"
## [9] "- Shell scripts"
##
## [[8]]
## [1] "- Data Engineer- Data Visualization" "- Data Warehousing"
## [3] "- ETL"                               "- SQL"
## [5] "- Data Integrity"                    "- Data Profiling"
## [7] "- Data Analysis"                     "- Data/ETL Solutions"
## [9] "- Shell scripts"
##
## [[9]]
## [1] "- Big Data: Application Development/Support, Visualization/Reporting, Plannin
```

```
g/Management- Cloud PaaS , iPaaS and Integrating Systems"
## [2] "- R language, Python. SSIS, Informatica experience"
## [3] "- Azure Environment knowledge"
## [4] "- SQL"
## [5] "- REST/SOAP API"
## [6] "- BI"
##
## [[10]]
## [1] "Qualifications:- 3+ years of experience in Telecom support/provisioning"
## [2] "- Prior experience with Telecom data circuits"
## [3] "- M-A-C-D's experience (Moves, Adds, Changes, Disconnects)"
## [4] "- Ability to provide support in a complex infrastructure"
## [5] "- Proficiency with Microsoft office (Sharepoint is a plus)"
## [6] "Nice to haves:"
## [7] "- BS in telecommunications or a related field (data also a plus)"
## [8] "- Knowledge of MPLS, VoIP, TDM, and/or PBX"
## [9] "- Ability to speak/write in Chinese and/or Japanese"
```

```
R <- length(grep('R',Required_skills))
python <- length(grep('python|Python',Required_skills))
SQL <- length(grep('SQL',Required_skills))
JAVA <- length(grep('Java',Required_skills))
program_name <- c('R','Python','SQL','Java')
program_count <- c(R,python,SQL,JAVA)
program.data <- data.frame(program_name,program_count)
program.data
```

```
##   program_name program_count
## 1            R           142
## 2       Python            51
## 3          SQL            62
## 4         Java            34
```

Then, I investigated on the location. I extarcted state abbreviation from each job-posts, as it can be seen, CA,IL,TX,VA,WA,NY have more demands for data related jobs, especially in CA, with a outstanding number of job requirements compared to other states.

```
library(tidyverse)
```

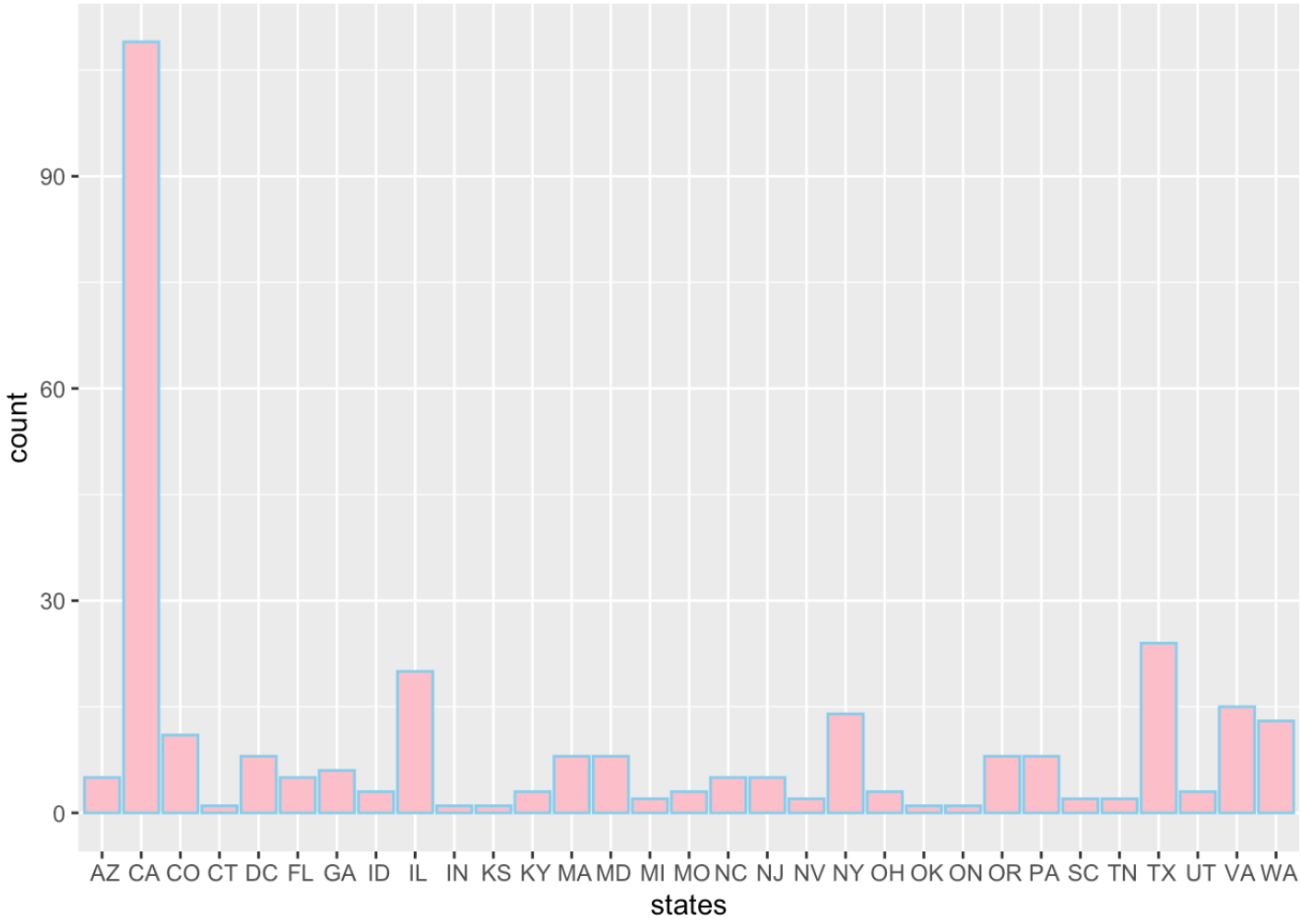
```
## ── Attaching packages ─────────────────────────────────────────────────────
──────────── tidyverse 1.3.0 ──
```

```
## ✓ tibble  2.1.3     ✓ purrr   0.3.3
## ✓ tidyr   1.1.2     ✓ dplyr   0.8.5
## ✓ readr   1.3.1     ✓ forcats 0.5.0
```

```
## ── Conflicts ─────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────

──── tidyverse_conflicts() ──
## x dplyr::arrange()   masks plyr::arrange()
## x purrr::compact()   masks plyr::compact()
## x tidyr::complete()  masks RCurl::complete()
## x dplyr::count()     masks plyr::count()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::mutate()    masks plyr::mutate()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
```

```
location <- sapply(1:length(datapost_all),function(i){
  location <- datapost_all[[i]][["location"]]
  stateabbre <- str_split(location,',')[1][[1]][[2]]
  return(stateabbre)
  })
location_state <- as.data.frame(unlist(location))
names(location_state)[1]<- "states"
ggplot(location_state,aes(states)) + geom_bar(position = 'identity',color = 'skyblue'
,fill = 'pink')
```



Then I am interested in how many jobs explicitly addressed data scientists or data analysts or statisticians in the job title. I first extracted all the tile and have it organized. I found out there are only 13 jobs explicitly mentioned data scientists in the title and only 11 jobs explicitly mentioned data analysts in the title. There is no 'statistician' mentioned explicitly in the title. It's interesting that many jobs related in the data do not explicitly mention as the title we known as 'data scientists' or 'data analysts' in the title. Also, when I took a detailed look at the job title, there are some job postings that are repetitive but posted on different dates.

```
title <- sapply(1:length(datapost_all),function(i){
  title <- datapost_all[[i]][["title"]]
  return(title)
  })

data_scientists <- grep('data scientist|Data Scientist|Data Scientists',title)
length(data_scientists)
```

```
## [1] 13
```

```
statistician <- grep('statistician',title)
length(statistician)
```

```
## [1] 0
```

```
data_analysts <- grep('data analyst|Data Analyst|Data Analysts|data analysts',title)
length(data_analysts)
```

```
## [1] 12
```

```
title_data <- data.frame('data_scientists' = length(data_scientists),
                         'data_analysts' = length(data_analysts),
                        'statistician' = length(statistician))
title_data
```

```
##   data_scientists data_analysts statistician
## 1              13            12            0
```