<u>Name</u>: Siddharth Mahesh Patil

<u>Course number</u>: DA5020 – Collecting, Storing, and Retrieving Data

<u>Semester</u>: Spring 2018

Project title: Collect, Store, and Retrieve data related to jobs listings on indeed.com

Description of the problem: A job seeker typically must spend a major chunk of his/her time in finding a job listing which suits their educational background, professional background, skills, and career interests. Jobs are posted on numerous employment websites, and performing analysis on these job listings is a time-consuming task. This is because of the absence of a centralized source which holds data about all the job listings posted online.

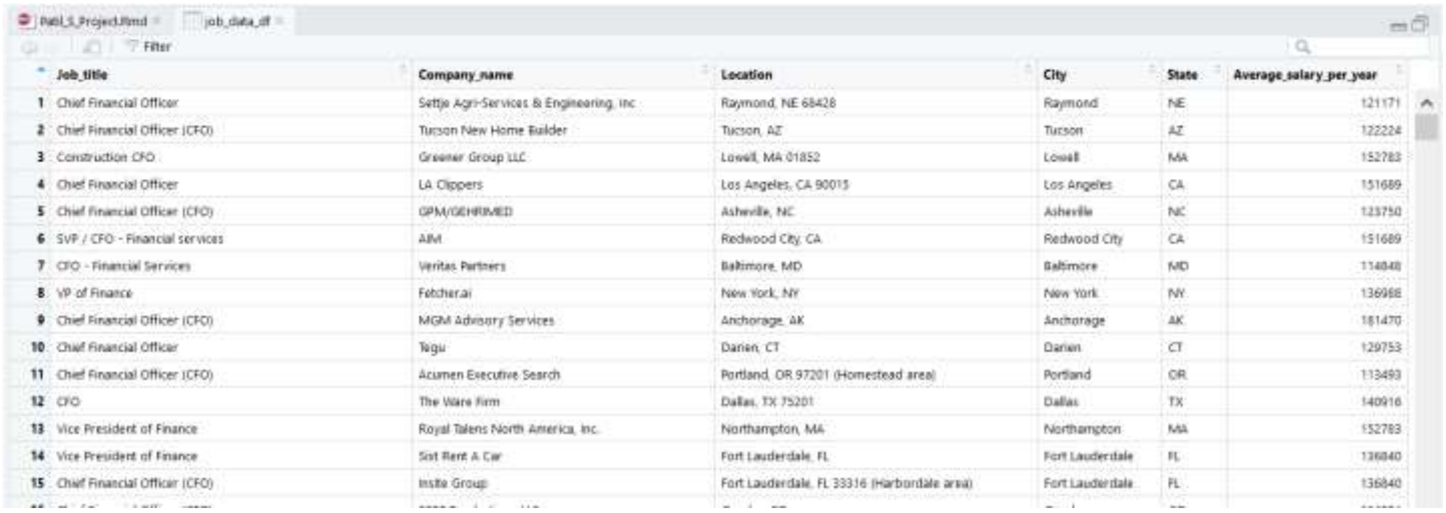
This project aims to help job seekers to take well-informed decisions based on analysis done on a centralized database containing all the job listings which are available online.

Rationale for the data collected to solve this problem: A job search engine searches the internet for all the jobs related to the search query, and is, therefore, the best data source for this project. I have chosen indeed.com to collect data about job listings since it fetches data from multiple sources, and it is highly unlikely that any job posted online does not show up in the search results on indeed.com

---

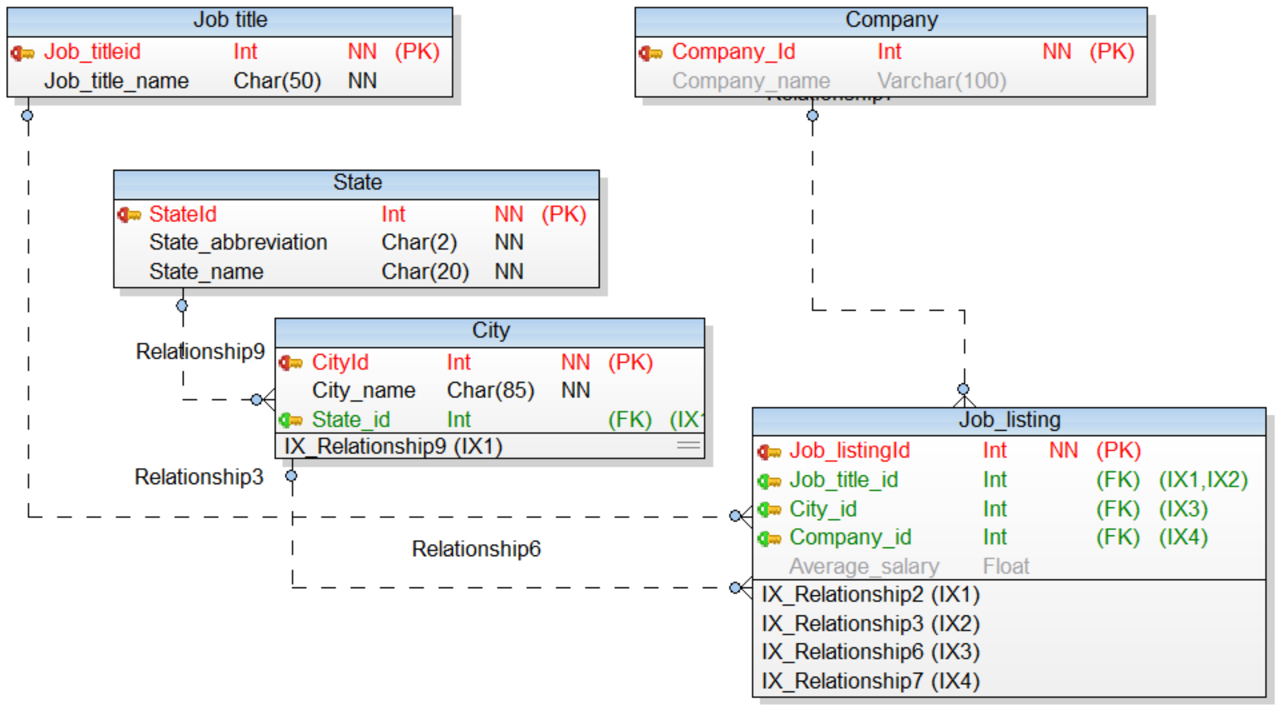
Justification for the processes implemented, and the technical choices made: The project is divided in the following three parts:

1) ***Part 1: Data Collection*** – This part consists of building a function ('indeed_job_info (keyword, location)') to scrape data from indeed.com, clean the scraped data, and organizing the clean data in a data frame. The function will take 'keyword' and 'location' as arguments. The 'keyword' must consist of one, or two character strings (e.g. 'CFO', 'Data Analyst'). The 'location' must be a city name, a state name, or an abbreviation for a state (e.g. 'Austin', 'Texas', 'TX'). A sample output of this function is seen in the following image:



	Job_title	Company_name	Location	City	State	Average_salary_per_year
1	Chief Financial Officer	Settje Agri-Services & Engineering, Inc	Raymond, NE 68428	Raymond	NE	121171
2	Chief Financial Officer (CFO)	Tucson New Home Builder	Tucson, AZ	Tucson	AZ	122224
3	Construction CFO	Greener Group LLC	Lowell, MA 01852	Lowell	MA	152783
4	Chief Financial Officer	LA Clippers	Los Angeles, CA 90015	Los Angeles	CA	151689
5	Chief Financial Officer (CFO)	GPM/GEHRIMED	Asheville, NC	Asheville	NC	123750
6	SVP / CFO - Financial services	AIM	Redwood City, CA	Redwood City	CA	151689
7	CFO - Financial Services	Veritas Partners	Baltimore, MD	Baltimore	MD	114848
8	VP of Finance	Fetcher.ai	New York, NY	New York	NY	136988
9	Chief Financial Officer (CFO)	MGM Advisory Services	Anchorage, AK	Anchorage	AK	181470
10	Chief Financial Officer	Tegu	Darien, CT	Darien	CT	129753
11	Chief Financial Officer (CFO)	Acumen Executive Search	Portland, OR 97201 (Homestead area)	Portland	OR	113493
12	CFO	The Ware Firm	Dallas, TX 75201	Dallas	TX	140916
13	Vice President of Finance	Royal Talens North America, Inc.	Northampton, MA	Northampton	MA	152783
14	Vice President of Finance	Sixt Rent A Car	Fort Lauderdale, FL	Fort Lauderdale	FL	136840
15	Chief Financial Officer (CFO)	Insite Group	Fort Lauderdale, FL 33316 (Harbordale area)	Fort Lauderdale	FL	136840

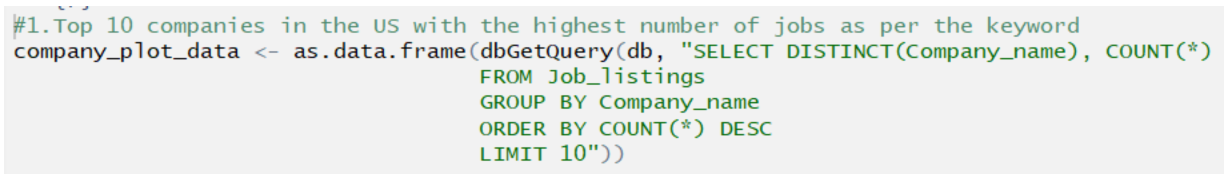
2) ***Part2: Data Storage*** – This part consists of creating a relational database to store the data that is collected in Part 1. A relational model ensures data integrity, and helps in data retrieval through 'Structured Query Language (SQL)' which is an industry standard. The structure of this relational database is seen in the following image:



3) ***Part 3: Data Retrieval*** – This part consists of retrieving the stored data by building queries using SQL and RSQLite – the R package which enables SQL queries to be written in RStudio. The results of these queries are stored in data frames, and these data frames will serve as a reference for the visualizations built to assist the job seeker. Following are two sample questions, the queries used to answer these questions, and the results of these queries:

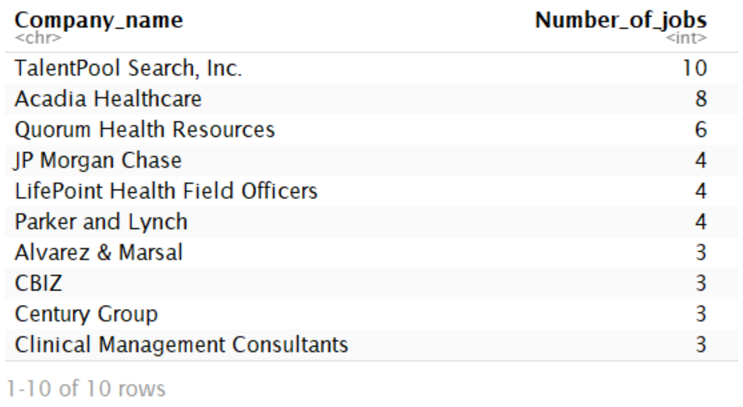
Question 1: Which are the top 10 companies with the most number of jobs as per the 'keyword' and the 'location' arguments for the function in Part 1?

Query:



```
#1.Top 10 companies in the US with the highest number of jobs as per the keyword
company_plot_data <- as.data.frame(dbGetQuery(db, "SELECT DISTINCT(Company_name), COUNT(*)
                                   FROM Job_listings
                                   GROUP BY Company_name
                                   ORDER BY COUNT(*) DESC
                                   LIMIT 10"))
```

__Result:__

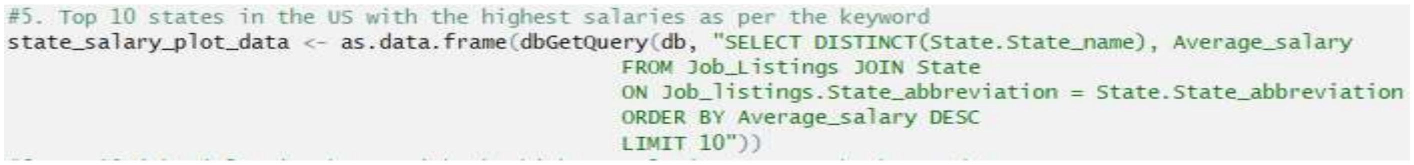


Company_name <chr>	Number_of_jobs <int>
TalentPool Search, Inc.	10
Acadia Healthcare	8
Quorum Health Resources	6
JP Morgan Chase	4
LifePoint Health Field Officers	4
Parker and Lynch	4
Alvarez & Marsal	3
CBIZ	3
Century Group	3
Clinical Management Consultants	3

1-10 of 10 rows

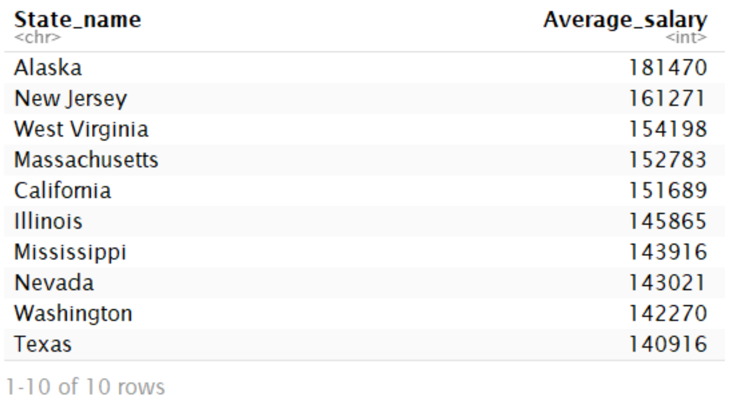
__Question 2:__ Which are the top 10 States in the US with the highest salaries as per the ‘keyword’?

__Query:__



```
#5. Top 10 states in the US with the highest salaries as per the keyword
state_salary_plot_data <- as.data.frame(dbGetQuery(db, "SELECT DISTINCT(State.State_name), Average_salary
                                        FROM Job_Listings JOIN State
                                        ON Job_listings.State_abbreviation = State.State_abbreviation
                                        ORDER BY Average_salary DESC
                                        LIMIT 10"))
```

__Result:__



State_name <chr>	Average_salary <int>
Alaska	181470
New Jersey	161271
West Virginia	154198
Massachusetts	152783
California	151689
Illinois	145865
Mississippi	143916
Nevada	143021
Washington	142270
Texas	140916

1-10 of 10 rows

Technical choices: 1) Web scraping was limited to just one website, since the page structure for each website is unique. This would require customized functions to be written for each website that we are interested in. Hence, to achieve a tradeoff between time required to build this application and the value that it will add to the overall job hunting process, I decided to scrape data only from indeed.com.

2) I have designed the function to scrape data from the first 50 pages of the search result.

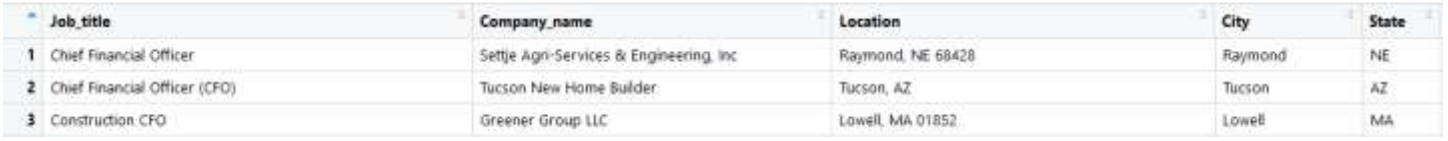
3) The salary information is scraped from simplyhired.com since this website offers more flexibility in terms of 'keyword' and 'location' as compared to other websites that provide salary information. E.g. indeed.com provides salary information only for a few cities. This would have put restrictions on the analysis that involved salary information.

The following two arguments are used to extract salary information from simplyhired.com:

- 'keyword' used in the function in Part 1
- 'State' in the data frame which is returned by the function in Part 1

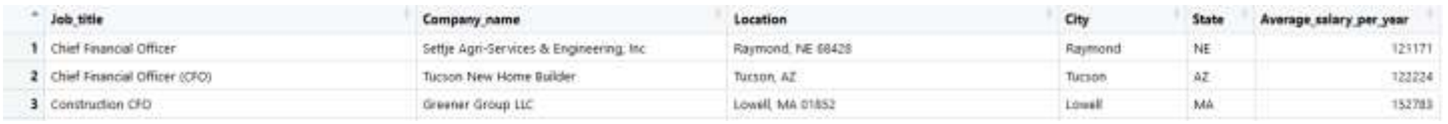
e.g.

`indeed_job_info('CFO', 'US')` returns the following data frame within the function itself:



	Job_title	Company_name	Location	City	State
1	Chief Financial Officer	Settje Agri-Services & Engineering, Inc	Raymond, NE 68428	Raymond	NE
2	Chief Financial Officer (CFO)	Tucson New Home Builder	Tucson, AZ	Tucson	AZ
3	Construction CFO	Greener Group LLC	Lowell, MA 01852	Lowell	MA

Salary information is, thus, scraped from simplyhired.com using the 'keyword' as 'CFO' and 'location' as 'NE' for the 1st row of the data frame. For the 2nd row, the 'keyword' remains the same, whereas the 'location' argument changes to 'AZ', and so on for the subsequent rows. The function then returns the following data frame:



	Job_title	Company_name	Location	City	State	Average_salary_per_year
1	Chief Financial Officer	Settje Agri-Services & Engineering, Inc	Raymond, NE 68428	Raymond	NE	121171
2	Chief Financial Officer (CFO)	Tucson New Home Builder	Tucson, AZ	Tucson	AZ	122224
3	Construction CFO	Greener Group LLC	Lowell, MA 01852	Lowell	MA	152783

The salaries scraped are average salaries per year in USD for that 'keyword' in that 'State'.

Debugging: I ran into issues only in the 'Data Collecting' part of the project. Following are the descriptions of the issues, and how I resolved them:

1) Incrementally moving on to the next webpage when scraping from the previous page is done

    - I build a 'for' loop which would run till the 1st 50 pages of the search result are scraped. I created an empty data frame with the same column names as the data frame which is returned by the 'for' loop. The loop incrementally updates the blank data frame every time it has created a data frame by scraping data from a page related to the search result.
    - The data frame that is returned after the complete execution of the 'for' loop has duplicate records, and missing values. Missing values are usually in the 'State' column, and are only in a very small number of records. Since we require the 'State' information to scrape the salary information, and since only a few rows do not have the 'State' information, I decided to drop the rows with missing values. Also, I dropped the duplicate rows. The following lines of code were used to get a cleaner data frame:

```
job_data_1 = job_data_1[!duplicated(job_data_1),]
job_data_df = job_data_1[complete.cases(job_data_1), ]
```

2) Scraping data from simplyhired.com

    - Reading the html from the webpages on simplyhired.com was leading to a connection error:400.
    - I solved this by removing the 'space' in between the string characters in 'keyword'. First, I split the 'keyword', and using the following 'if..else' statement created another keyword (i.e. 'kw_salary') argument by removing the 'space'. The following lines of code show how this was done:

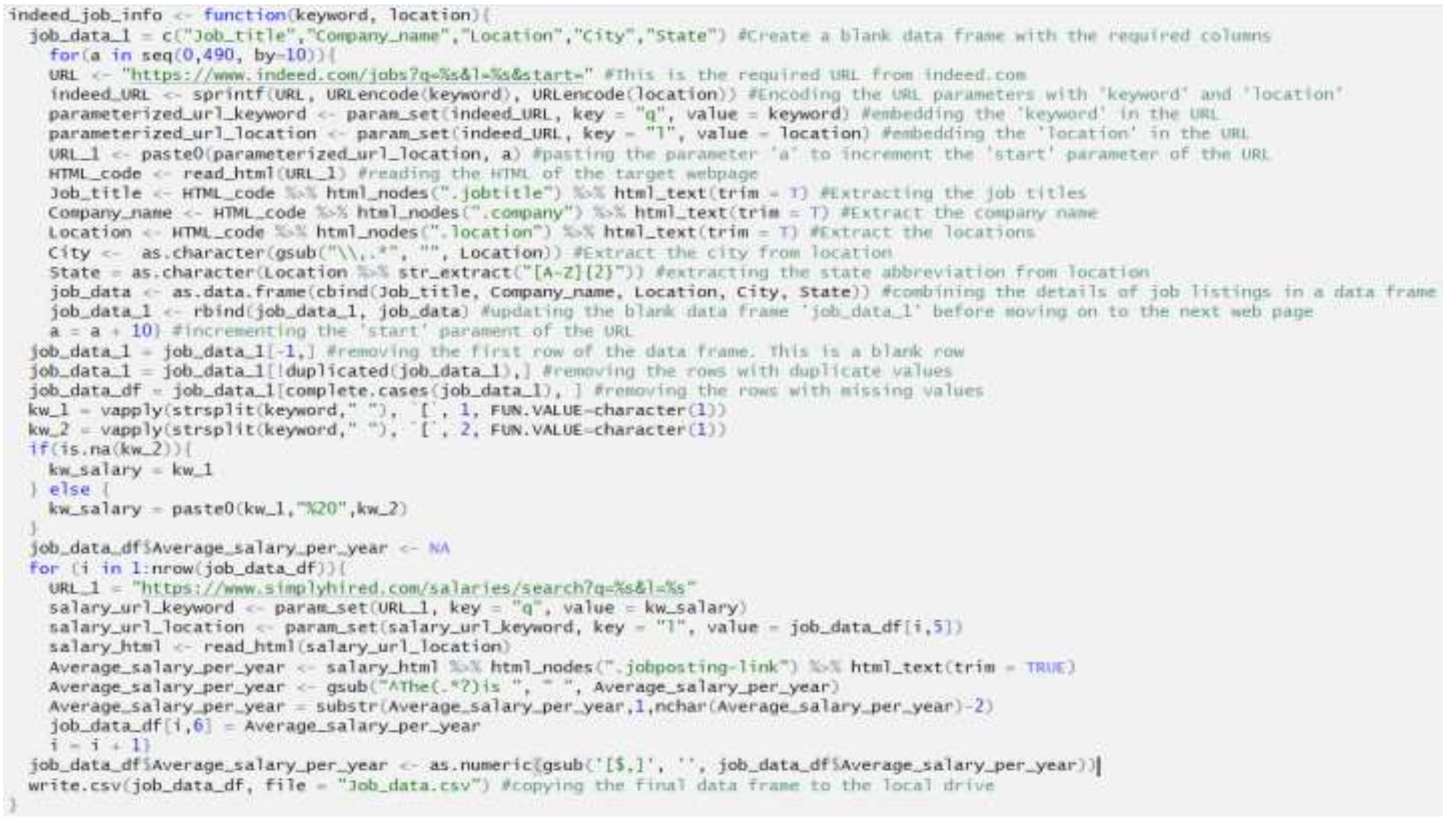
```
kw_1 = vapply(strsplit(keyword," "), `[`, 1, FUN.VALUE=character(1))
kw_2 = vapply(strsplit(keyword," "), `[`, 2, FUN.VALUE=character(1))

if(is.na(kw_2)){
  kw_salary = kw_1
} else {
  kw_salary = paste0(kw_1,"%20",kw_2)
}
```

    - This is also the reason why I have limited the number of character strings that can be entered as 'keyword' to two with a space in between them as mentioned in 'Part 1: Data Collection'.

Input and Output results from an example session of R code with insightful screenshots:

Following is the function:

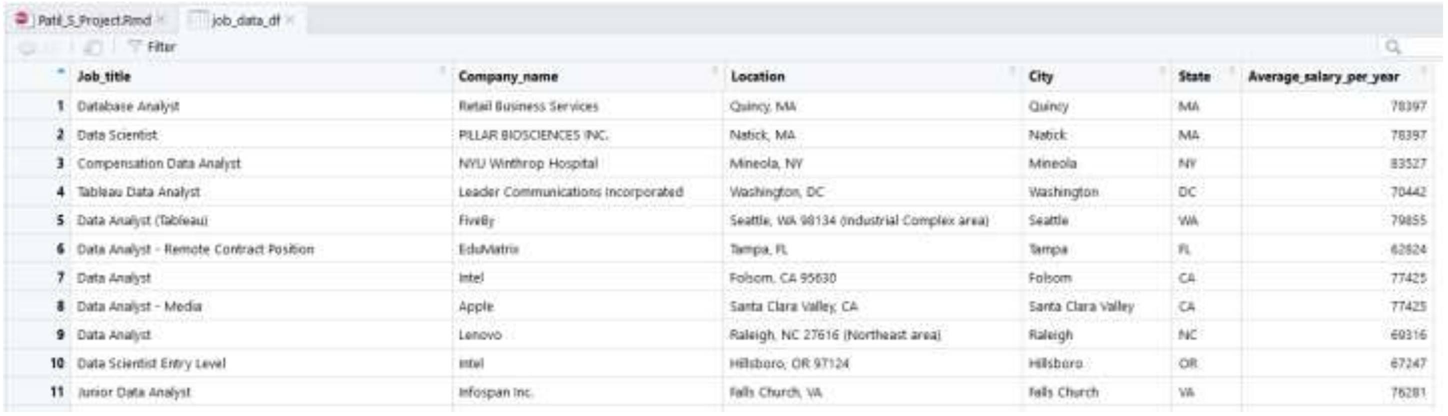


```r
indeed_job_info <- function(keyword, location){
  job_data_1 = c("Job_title","Company_name","Location","City","State") #Create a blank data frame with the required columns
    for(a in seq(0,490, by=10)){
    URL <- "https://www.indeed.com/jobs?q=%s&l=%s&start=" #This is the required URL from indeed.com
    indeed_URL <- sprintf(URL, URLencode(keyword), URLencode(location)) #Encoding the URL parameters with 'keyword' and 'location'
    parameterized_url_keyword <- param_set(indeed_URL, key = "q", value = keyword) #embedding the 'keyword' in the URL
    parameterized_url_location <- param_set(indeed_URL, key = "l", value = location) #embedding the 'location' in the URL
    URL_1 <- paste0(parameterized_url_location, a) #pasting the parameter 'a' to increment the 'start' parameter of the URL
    HTML_code <- read_html(URL_1) #reading the HTML of the target webpage
    Job_title <- HTML_code %>% html_nodes(".jobtitle") %>% html_text(trim = T) #Extracting the job titles
    Company_name <- HTML_code %>% html_nodes(".company") %>% html_text(trim = T) #Extract the company name
    Location <- HTML_code %>% html_nodes(".location") %>% html_text(trim = T) #Extract the locations
    City <-  as.character(gsub("\\,.*", "", Location)) #Extract the city from location
    State = as.character(Location %>% str_extract("[A-Z]{2}")) #extracting the state abbreviation from location
    job_data <- as.data.frame(cbind(Job_title, Company_name, Location, City, State)) #combining the details of job listings in a data frame
    job_data_1 <- rbind(job_data_1, job_data) #updating the blank data frame 'job_data_1' before moving on to the next web page
    a = a + 10) #incrementing the 'start' parament of the URL
  job_data_1 = job_data_1[-1,] #removing the first row of the data frame. This is a blank row
  job_data_1 = job_data_1[!duplicated(job_data_1),] #removing the rows with duplicate values
  job_data_df = job_data_1[complete.cases(job_data_1), ] #removing the rows with missing values
  kw_1 = vapply(strsplit(keyword," "), `[`, 1, FUN.VALUE=character(1))
  kw_2 = vapply(strsplit(keyword," "), `[`, 2, FUN.VALUE=character(1))
  if(is.na(kw_2)){
    kw_salary = kw_1
  } else {
    kw_salary = paste0(kw_1,"%20",kw_2)
  }
  job_data_df$Average_salary_per_year <- NA
  for (i in 1:nrow(job_data_df)){
    URL_1 = "https://www.simplyhired.com/salaries/search?q=%s&l=%s"
    salary_url_keyword <- param_set(URL_1, key = "q", value = kw_salary)
    salary_url_location <- param_set(salary_url_keyword, key = "l", value = job_data_df[i,5])
    salary_html <- read_html(salary_url_location)
    Average_salary_per_year <- salary_html %>% html_nodes(".jobposting-link") %>% html_text(trim = TRUE)
    Average_salary_per_year <- gsub("^The(.*?)is ", " ", Average_salary_per_year)
    Average_salary_per_year = substr(Average_salary_per_year,1,nchar(Average_salary_per_year)-2)
    job_data_df[i,6] = Average_salary_per_year
    i = i + 1}
  job_data_df$Average_salary_per_year <- as.numeric(gsub('[$,]', '', job_data_df$Average_salary_per_year))
  write.csv(job_data_df, file = "Job_data.csv") #copying the final data frame to the local drive
}
```

Following is a sample input:

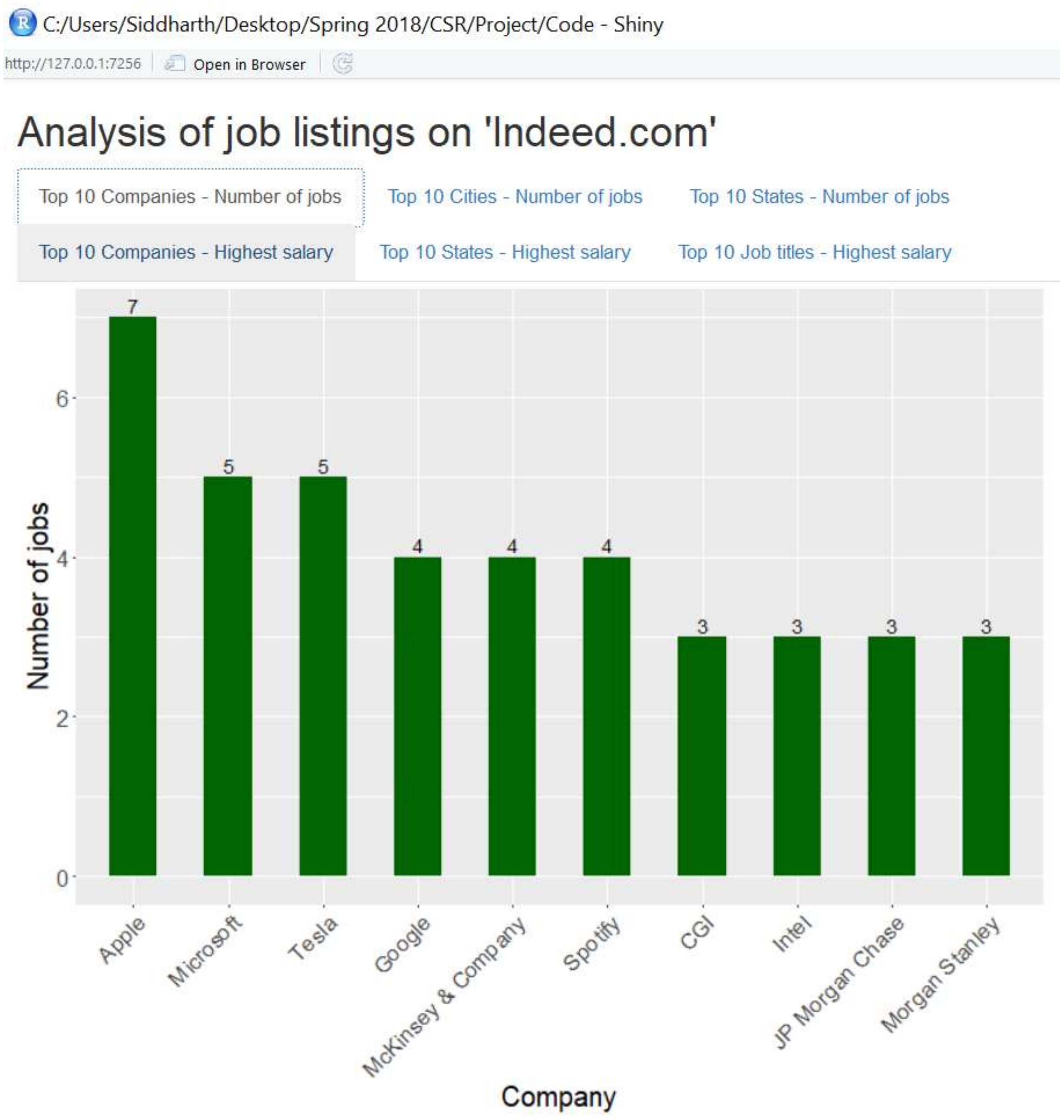
```r
indeed_job_info('Data Analyst', 'US') #Getting data for a sample keyword and location
```

And, following is the output for the sample input:


Patil_S_Project.Rmd
job_data_df
Filter

	Job_title	Company_name	Location	City	State	Average_salary_per_year
1	Database Analyst	Retail Business Services	Quincy, MA	Quincy	MA	78397
2	Data Scientist	PILLAR BIOSCIENCES INC.	Natick, MA	Natick	MA	78397
3	Compensation Data Analyst	NYU Winthrop Hospital	Mineola, NY	Mineola	NY	83527
4	Tableau Data Analyst	Leader Communications Incorporated	Washington, DC	Washington	DC	70442
5	Data Analyst (Tableau)	FiveBy	Seattle, WA 98134 (Industrial Complex area)	Seattle	WA	79855
6	Data Analyst - Remote Contract Position	EduMatrix	Tampa, FL	Tampa	FL	62824
7	Data Analyst	Intel	Folsom, CA 95630	Folsom	CA	77425
8	Data Analyst - Media	Apple	Santa Clara Valley, CA	Santa Clara Valley	CA	77425
9	Data Analyst	Lenovo	Raleigh, NC 27616 (Northeast area)	Raleigh	NC	69316
10	Data Scientist Entry Level	Intel	Hillsboro, OR 97124	Hillsboro	OR	67247
11	Junior Data Analyst	Infospan Inc.	Falls Church, VA	Falls Church	VA	76281

- The visualization created using ggplot2 and Shiny packages provide the job seeker with useful insights related to the 'Number of jobs' and 'Average salary'. Following are two of the visualizations:

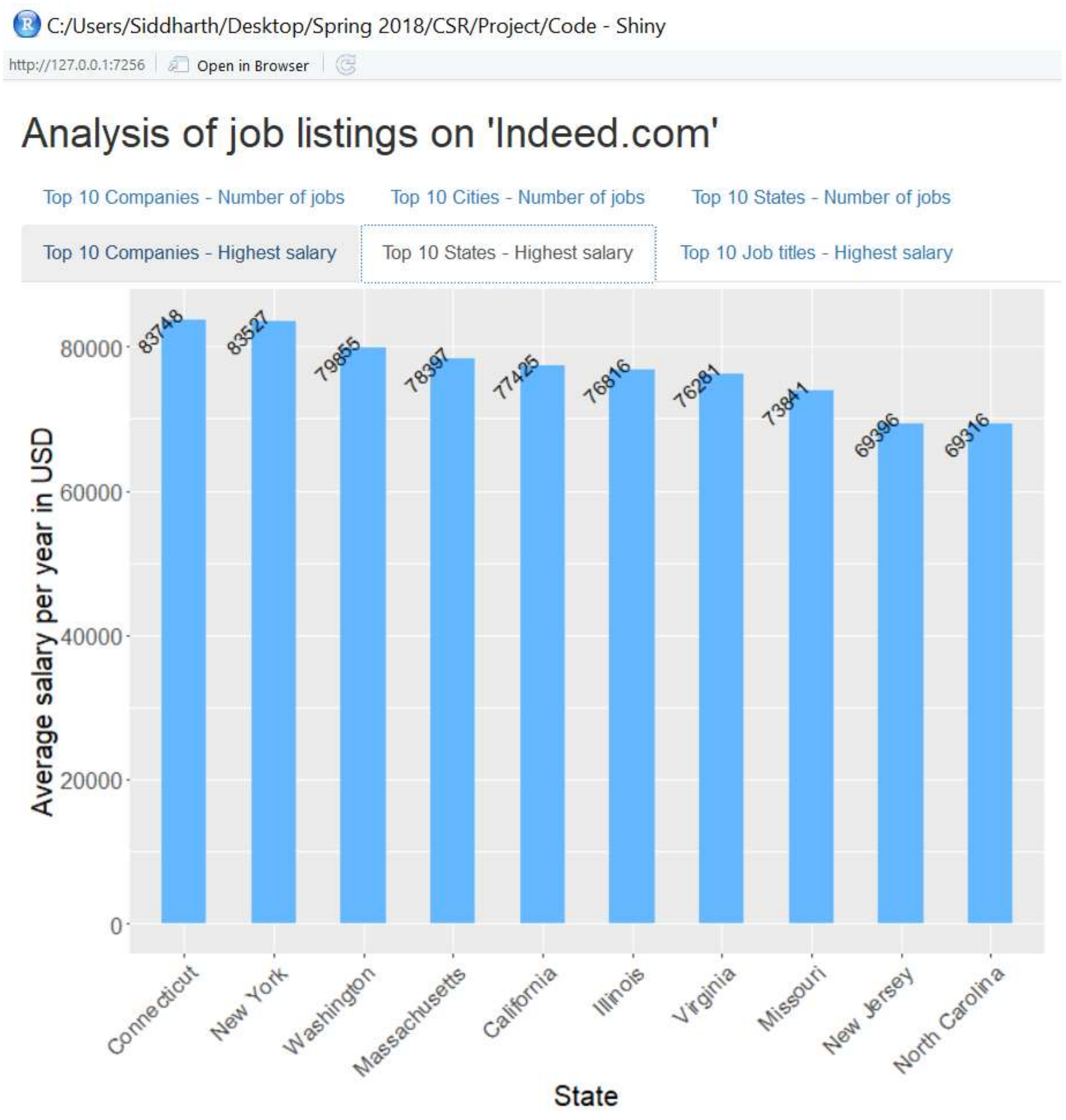


C:/Users/Siddharth/Desktop/Spring 2018/CSR/Project/Code - Shiny

http://127.0.0.1:7256 Open in Browser

Analysis of job listings on 'Indeed.com'

Top 10 Companies - Number of jobs
Top 10 Cities - Number of jobs
Top 10 States - Number of jobs
Top 10 Companies - Highest salary
Top 10 States - Highest salary
Top 10 Job titles - Highest salary
Number of jobs
Company
0
2
4
6
Apple 7
Microsoft 5
Tesla 5
Google 4
McKinsey & Company 4
Spotify 4
CGI 3
Intel 3
JP Morgan Chase 3
Morgan Stanley 3


C:/Users/Siddharth/Desktop/Spring 2018/CSR/Project/Code - Shiny
http://127.0.0.1:7256
Open in Browser
Analysis of job listings on 'Indeed.com'
Top 10 Companies - Number of jobs
Top 10 Cities - Number of jobs
Top 10 States - Number of jobs
Top 10 Companies - Highest salary
Top 10 States - Highest salary
Top 10 Job titles - Highest salary
Average salary per year in USD
State
0
20000
40000
60000
80000
83748
83527
79855
78397
77425
76816
76281
73841
69396
69316
Connecticut
New York
Washington
Massachusetts
California
Illinois
Virginia
Missouri
New Jersey
North Carolina

<u>Insights on what is learned, and potential future work:</u>

1) Insights:
   - R is highly versatile when we consider writing functions and loops to manipulate data.
   - R has very useful libraries that enable almost all activities from data collection, data cleaning, model building, and visualization. This makes R a very valuable tool for extracting value from data.
   - R Shiny enables interactive applications to be built in R which are useful to business users irrespective of their expertise in R.

2) Potential Future Work:
   - I want to reduce the time taken to scrape the data to not more than 30 seconds. As of now, the code takes a few minutes to return the data frame.
   - I want to keep the code completely separated from the user. The user must only be required to enter the 'keyword' and the 'location' in the Shiny application, and the application must display the plots as per these user inputs.
   - I want to add information related to cost of living on a city-level. This will also help the user in making decisions that are financially sound.

<u>References:</u>

1) Indeed.com
2) Simplyhired.com
3) stackoverflow.com
