

DA 5020
COLLECT, STORE AND RETRIEVE DATA

LINKEDIN DATA ANALYSIS

Under the Guidance of
DR. KATHLEEN DURANT

-BY

Kavya Menon: menon.k@husky.neu.edu

Mrinal Payannavar: payannavar.m@husky.neu.edu



Northeastern University

TABLE OF CONTENT

1. Introduction.....	3
2. Data Collection	
2.1. Data Collection using Web-scraping tools.....	4
2.2. Data from web sources.....	5
3. Storage using MongoDB	
3.1. MongoDB and the related packages.....	6
3.2. Other packages used.....	8
4. Retrieval and Analysis of Data.....	10
5. Issues Faced	17
6. Learning Outcomes.....	18
7. References.....	19

INTRODUCTION

LinkedIn is one of the most common business and employment oriented social networking websites today. With about 467 million users worldwide and 3 million active job listings, the website is a storehouse of resources.

LinkedIn serves as the perfect platform for professionals to interact and network- be it to seek advice from industry professionals, looking for job opportunities in industries with a specific skill set or even to understand/ learn the top job skills required in the market. Moreover, LinkedIn supports bridging the gap between the job-seekers and employers by the provision of job-listings with additional benefits of Easy-apply and Level-based jobs.

As Graduate students looking for job opportunities post-Graduation, we believe the LinkedIn website has a lot more data that can be utilized to benefit our job-search process. Through our project, we plan to mine this very data, analyze and break-down the industry-skill relationship for a job-seeker and represent it in an easy to comprehend format.

The purpose of our project is to collect the data from LinkedIn using web-scraping tools and other readily available data sources online to pull useful information. Further, store them in a compatible database-storage structure, here MongoDB. The data so collected is cleaned and standardized for making it easy to work on.

Lastly, the data is analyzed using R to draw out analyses and detect job trends.

This is done using R-supported plots and graphs, which convey the insights of the study in a concise and complete form.

COLLECTION

We collected data for our project using two major sources:

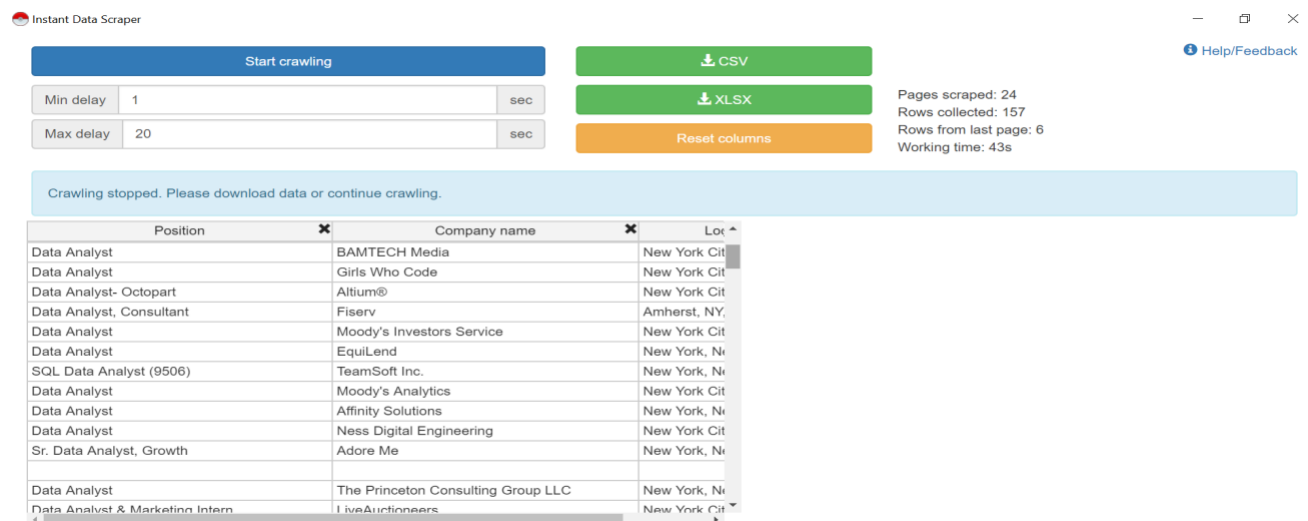
- Web-scraping using built-in tools
- Using data sources online

WEB SCRAPING:

To extract job profile data from the LinkedIn web-page, we worked on three built-in web-scrappers: Instant Data Scraper, Grepsr and Agenty. Of these three, we chose Instant Data Scraper to create our database of Job name, Industry name and the location.

Instant Data Scraper

Instant Data Scraper is a Chrome extension, that crawls listing type data from multiple pages. It uses AI to guess which data is most relevant on a page and allows alternative selections. Data can be exported as Excel or CSV files.



Instant Data Scraper

Start crawling

Min delay 1 sec

Max delay 20 sec

Download CSV

Download XLSX

Reset columns

Pages scraped: 24
Rows collected: 157
Rows from last page: 6
Working time: 43s

Crawling stopped. Please download data or continue crawling.

Position	Company name	Location
Data Analyst	BAMTECH Media	New York City
Data Analyst	Girls Who Code	New York City
Data Analyst- Octopart	Altium®	New York City
Data Analyst, Consultant	Fiserv	Amherst, NY
Data Analyst	Moody's Investors Service	New York City
Data Analyst	EquiLend	New York, NY
SQL Data Analyst (9506)	TeamSoft Inc.	New York, NY
Data Analyst	Moody's Analytics	New York City
Data Analyst	Affinity Solutions	New York, NY
Data Analyst	Ness Digital Engineering	New York City
Sr. Data Analyst, Growth	Adore Me	New York, NY
Data Analyst	The Princeton Consulting Group LLC	New York, NY
Data Analyst & Marketing Intern	LiveAurionneers	New York City

DATA SOURCES:

Apart from using Instant Data Scraper, we used folders of zipped JSON files available on Reddit for profile information on users. Within our project, we have chosen the top three folders to limit the data so extracted.

```
# Store the links to the zipped json files containing profile data
file_list <- c( "https://s3.amazonaws.com/michaelfy_linkedinquire/a/1339943811.zip",
               "https://s3.amazonaws.com/michaelfy_linkedinquire/a/1340908651.zip",
               "https://s3.amazonaws.com/michaelfy_linkedinquire/a/1341183840.zip")
```

The JSON files were within zipped folders, which were unzipped using R.

```
# download the online LinkedIn data and unzip them into Json files
path <- "~/Desktop/LinkdinData" #creating a general path to create folder linkedin
path1 <- "~/Desktop/LinkdinData/JsonFile"
# download the online LinkedIn data and unzip them into Json files
for (i in 1:length(file_list)){
  file_name_1 <- substr(file_list[i], nchar(file_list[i])-15, nchar(file_list[i]))
  file_name_2 <- substr(file_list[i], nchar(file_list[i])-13, nchar(file_list[i]))
  file_name <- paste(path, file_name_1, "_", file_name_2, sep="")
  download.file(file_list[i], destfile=file_name, method="curl")
  unzip(file_name, exdir=path1)
}
```

Paths have been specified as path and path1 to store the zipped files and the unzipped JSON files respectively. The above code downloads the file in the links provided in the file list, unzipping and storing the JSON files, creating an unzipped folder on the Desktop called Linkdin Data, that has a collection of multiple JSON files, each with data of individual users.

STORAGE

As our project involved using data on JSON files, we have used MongoDB as the database system to store the data so collected.

MONGODB:

MongoDB is an open-source document-oriented free database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas. MongoDB supports field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Queries can also be configured to return a random sample of results of a given size

Package Mongolite

We create two collections, one for the JSON files called, profiles and one for the scraped data, scraped. Inserting the data through standard syntax and storing it as non-relational documents in this document-style database. The data from the NoSQL database is further retrieved and analyzed separately

```
#Storing the Linkedin Data in Mongodb
```

```
library(mongolite)
mon_linkedin <- mongo("profiles")
mon_linkedin$drop()
mon_linkedin$insert(linkedin_data)
```

```
# Storing the data in Mongolite
```

```
mon_scraped <- mongo("scraped")
mon_scraped$drop()
mon_scraped$insert(scraped)
```

Package jsonlite

A fast JSON parser and generator optimized for statistical data and the web. The package offers flexible, robust, high performance tools for working with JSON in R and is particularly powerful for building pipelines and interacting with a web API.

the package jsonlite helps in reading the json files to R, for further cleaning and analysis procedure with the function fromJSON(). This reads the profile information from the downloaded json files. We split the dataset by columns and eliminate the rows that have NULL values.

Combining the columns to form a dataset that has information of positions, skills, locations, industry and education. Columns like skills, education consist of rows containing list of elements. Another reason for preferring MonogDB to store the data.

```
library(jsonlite)
# extracting files that match the regular expression
alphabet <- c("a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z")
for (j in 1:length(alphabet)){
  regex <- paste("^", alphabet[j], ".*json$", sep="")
  file_list <- list.files(path= path1, pattern=regex)
  print(length(file_list))

  # keep the data only have full information on location, positions, industry, skills and educations
  linkedin_data <- NULL #giving null to data frame
  for (i in 1:length(file_list)){
    tryCatch({
      jsonData <- fromJSON(paste("~/Desktop/LinkdinData/JsonFile/", file_list[i], sep=""))
      field_name <- names(jsonData)
      # Selecting files that do not have Null values
      json_loc <- (jsonData$location != "NULL")
      json_pos <- (jsonData$positions != "NULL")
      json_ind <- (jsonData$industry != "NULL")
      json_sk <- (jsonData$skills != "NULL")
      json_edu <- (jsonData$educations != "NULL")
      # Logically combining them to eliminate risk of missing rows and mismatch
      json <- (json_loc & json_pos & json_ind & json_sk & json_edu) |
      jsonData <- jsonData[json, c("location", "positions", "industry", "skills", "educations")]
      linkedin_data <- rbind(linkedin_data, jsonData)
    },
    error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  }
}
```

OTHER PACKAGES USED:

DPLYR: A fast, consistent tool for working with data frame like objects, both in memory and out of memory. dplyr provides a flexible grammar of data manipulation, focused on tools for working with data frames.

The attributes used within dplyr include select, ggplots, filter, mutate, group_by and other functions.

TIDYVERSE: The 'tidyverse' is a set of packages that work in harmony because they share common data representations and 'API' design. This package is designed to make it easy to install and load multiple 'tidyverse' packages in a single step. Tidyverse packages include many functionalities, but the project primarily uses it for tidying our data. Separate is one of the functions used to tidy the data.

```
library(tidyverse)
scraped <- as.data.frame(read_csv("project.csv"))
# Tidying up the data
colnames(scraped) <- c("Position", "Company", "Location")
scraped$Location <- gsub("(.*),.*", "\\1", scraped$Location)
scraped <- scraped %>%
  separate(Location, into = c("City", "State"), sep = ",", na.rm = TRUE)
```

STRINGR: A package for simple, Consistent Wrappers for Common String Operations. The arguments used in the project under STINGR include str_detect.

DATA.TABLE: Fast aggregation of large data, fast ordered joins, fast add/modify/delete of columns by group using no copies at all, list columns, a friendly file reader and parallel file writer. The project uses merge function to read the data as a data frame.

```
# Locations
library(stringr)
library(data.table)
loc <- mon_linkedin$find(fields = '{"industry":1, "location":1, "_id":0}')
location <- loc %>%
  filter(str_detect(location, "New York"))
#merged the top industries and the data in New york
location <- merge(topindustry, location, by.x = "Industry", by.y = "industry")
location <- location[,-2]
```

RETRIEVAL AND ANALYSIS

MongoDB uses JSON based syntax to query documents. Our data has 17,830 records of different industries, the positions and the skills required along with other attributes. Displaying a part of the first record in the data to show how the records are stored within MongoDB:

```
> mon_linkedin$iterate()$one()
$location
[1] "Hyderabad Area, India"

$positions
$positions[[1]]
$positions[[1]]$title
[1] "Relationship Manager- WM"

$positions[[1]]$`start-date`
[1] "2009-04-01"

$positions[[1]]$`is-current`
[1] TRUE
```

A. Retrieval and Analysis of Industries

Retrieving the data that contains a list of the top industries from the database, and then arranging them in the order of recurring frequency

```
# getting Industry data from MongoDB
industry <- mon_linkedin$find(fields = '{"industry" :1, "_id":0}')
library(dplyr)
industrydata <- as.data.frame(table(industry))
colnames(industrydata) <- c("Industry", "Frequency")
industrydata <- arrange(industrydata, desc(Frequency))

# pick the top industries
num <- 20
topindustry <- head(industrydata, 20)
data <- NULL
```

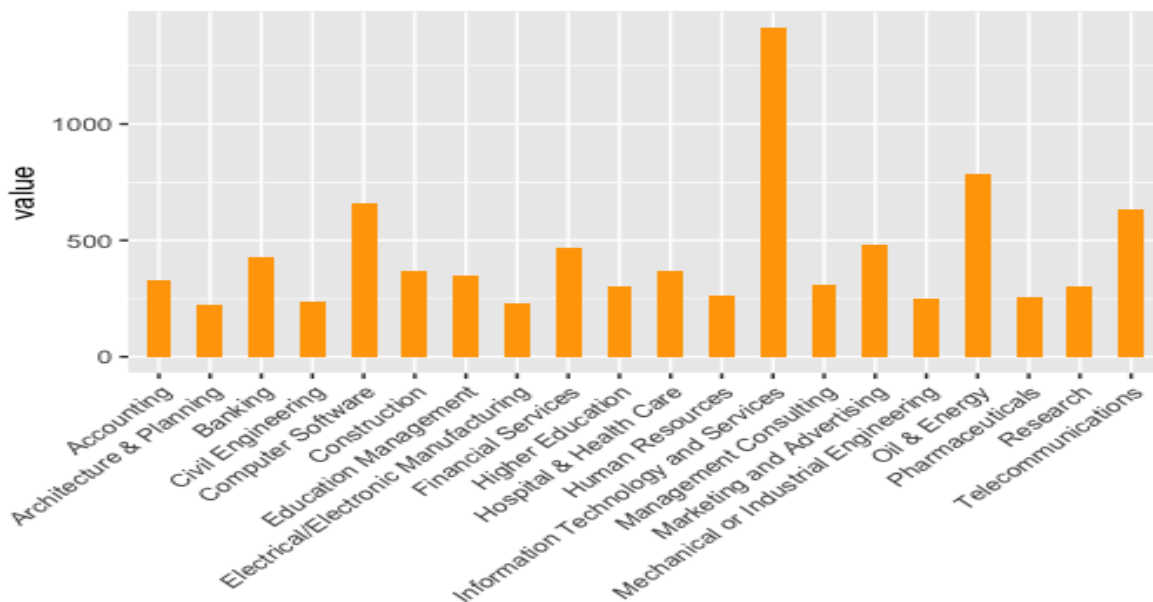
The Top industries table include the following,

	Industry	Frequency
1	Information Technology and Services	1414
2	Oil & Energy	787
3	Computer Software	661
4	Telecommunications	630
5	Marketing and Advertising	483
6	Financial Services	471

Data visualization for a bar plot of the top 20 industries, can be sketched using the R package ggplot2.

```
# Bar plot showing Top Industries
ggplot(topindustry, aes(x= Industry, y=Frequency))+
  theme(axis.text.x = element_text(angle = 45, hjust=1))+
  geom_bar(stat = "identity",position="stack", width = 0.5, fill = I("orange"))+
  labs(main = paste("Top", num , "Industries"), x = "", y = "value")
```

From the plot, we can see that Information Services and technology is the topmost industry that people work in. Next comes, Oil and Energy, Computer Software and so on.



B. Retrieval and Analysis of Skills

The skills data is stored as rows filled with lists of skills found in the LinkedIn profiles data. We examine the skills that are most used in the Information Services and Technology field, since it has proved to be the most sought after one.

	skills
1	c("Software Project Management", "E-commerce", "SC...
2	c("Web Design and Development", "Communication Sk...
3	c("SQL Server", "Unix", "MySQL", "Mobile Apps Testing"...
4	Teaching Writing
5	c("SEM", "SEO", "Website usability Testing", "Text opti...
6	c(".NET", "C#", "WPF,WCF,LINQ", "ASP.NET", "WPF", "WC...

After retrieving the column that contain lists of skills as rows, we separated them using the `separate_rows()` function of the tidyverse package. Filtering out the blank rows and a few unwanted anomalies, the set of skills are arranged according to the number of times they occur. The top 20 skills required for the IT services industry are stored in the table `top skills`.

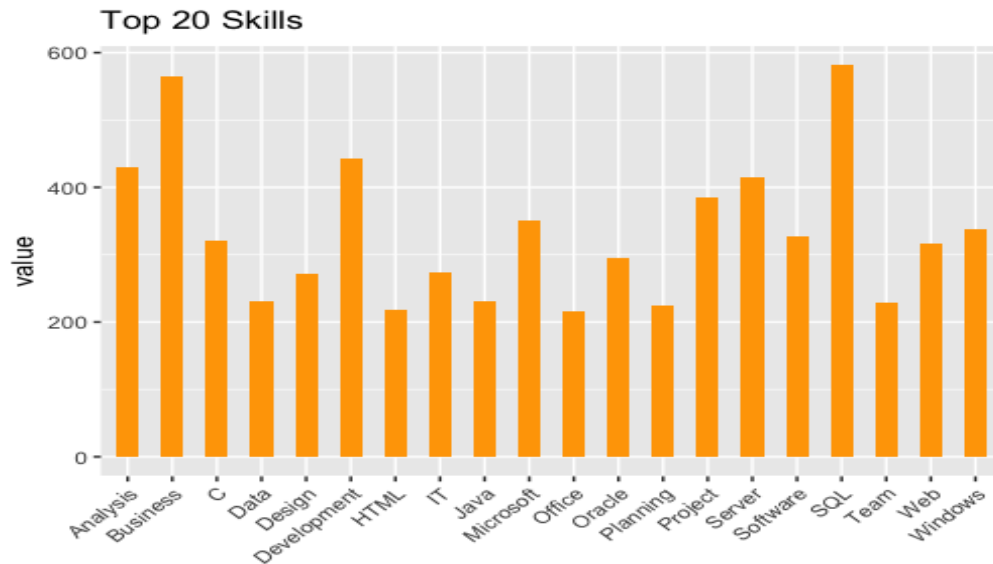
```
# Skills for Topmost Industry in demand

skills <- mon_linkedin$find(query = '{"industry": "Information Technology and Services"}',
                           fields = '{"skills":1, "_id" : 0}')

IT_skills <- skills %>%
  separate_rows(skills)%>%
  filter(skills != "[[:punct:]]" & skills!= "Management" & skills != "")
IT_skills <- as.data.frame(table(IT_skills$skills))
colnames(IT_skills) <- c("Skills", "freq")
topskills <- IT_skills %>% arrange(desc(freq)) %>%
  head(20)
```

`ggplot2` is again used to visualize the top 20 skills for Information Technology and Services

```
# Bar plot showing Top skills
ggplot(topskills, aes(x=Skills, y=freq))+
  theme(axis.text.x = element_text(angle = 45, hjust=1))+
  geom_bar(stat = "identity",position="stack", width = 0.5, fill = I("orange"))+
  labs(title = "Top 20 Skills", x = "", y = "value")
```



The above plot shows that SQL is the most required technical required, along with the knowledge of business. Next, analysis can be helpful to start a career in the industry. Among the rest is Development, Microsoft, SQL Server and more.

C. Retrieval and analysis of jobs by Location.

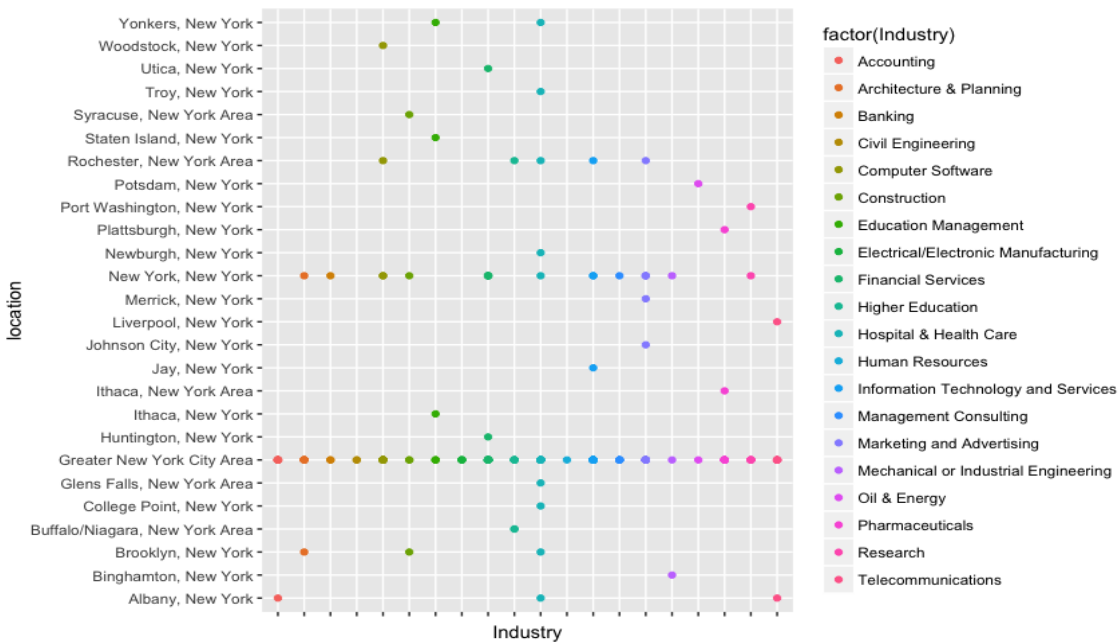
Shifting focus to the state of New York and showing the industries that are located in different cities within New York.

```
# Locations
library(stringr)
library(data.table)
loc <- mon_linkedin$find(fields = '{"industry":1, "location":1, "_id":0}')
location <- loc %>%
  filter(str_detect(location,"New York")) %>%
  group_by(industry)
#merged the top industries and the data in New york
location <- merge(topindustry, location, by.x = "Industry", by.y = "industry")
location <- location[,-2]
```

Visualizing the distribution of the top industries in demand throughout the state of New York, using ggplot2 we get,

```
# Bar plot showing demand Top industries in New York
ggplot(location, mapping = aes( x= Industry, y = location)) +
  geom_point(aes(color = factor(Industry))) +
  theme(axis.text.x = element_blank())+
  labs(main = "Top Industries in New York by Locations", xlab = "Industry")
```

The plot shows the distribution of industries over New York.



D. Retrieval of Current Job Postings

We scraped the LinkedIn searches for the jobs with positions in Information Technology and Services, and stored it in MongoDB.

Retrieving the data that for the position of Data Analyst from MongoDB.

```
# Jobs of a data analyst role
analyst <- mon_scraped$find(query = '{"Position": { "$regex" : "^Data Analyst", "$options": "i" }}',
                           fields = '{"_id":0, "State": 0}')
```

	Position	Company	City
1	Data Analyst	Distinctive Workforce Solutions	New York
2	Data Analyst	Moody's Analytics	New York City
3	Data Analyst	EquiLend	New York
4	Data Analyst	Shutterstock	New York City
5	Data Analyst	ContentSquare	New York
6	Data Analyst	Ness Digital Engineering	New York City
7	Data Analyst	Intertek	New York City
8	Data Analyst Intern	Winsor Consult Group	New York City

Getting jobs by location at Buffalo, New York

```
# Jobs at Buffalo
Buffalo <- mon_scraped$find(query = '{"City": { "$regex" : "^Buffalo", "$options": "i" }}', fields = '{"_id":0, "State": 0}')
```

	Position	Company	City
1	Human Resources/ Customer Service Specialist	Delaware North	Buffalo
2	Data Reporting Analyst	ABC-Amega	Buffalo
3	QA ANALYST	Axiom Technologies Inc.	Buffalo
4	Business and Planning Manager III – Credit Data Soluti...	M&T Bank	Buffalo
5	Reimbursement Analyst I	Roswell Park Cancer Institute	Buffalo

Retrieving Jobs by a specific company, here we find a company called Horizon Media

```
# Jobs at Horizon media  
company <- mon_scraped$find(query = '{"Company": "Horizon Media"}',  
                             fields = '{"_id":0, "State": 0}')
```

	Position	Company	City
1	Digital Analyst, Data & Analytics	Horizon Media	Greater New York City Area
2	Data Architect, Media Technology	Horizon Media	Greater New York City Area

ISSUES FACED

1. In 2013, LinkedIn restricted the functionality of its open API as well, scraping of its contents using Rvest. Hence, after a lot of internet browsing, we found, a link that provided zipped JSON files of the information contained in the open profiles on LinkedIn.
2. When cleaning the JSON files to exclude the rows that are have NULL values, there was a risk of the disarraying the rows, i.e. mismatching the profiles and other columns. To overcome this, we used logical AND to combine all corresponding true values. Thus, eliminating the risk.
3. The skills column of the LinkedIn data contains rows that have lists of skills corresponding position and industries. Thus, enlisting them separately and displaying it in relation to the industries was something we had not done before. We used `separate_rows()` function of the dplyr package.
4. The data collected in the JSON files amounted to almost 5GB. Hence, during the download process, we need to stop the running of the code, after about 5 mins. And continue working with whatever data has been collected by then, which in itself is a huge amount.

LEARNING OUTCOMES

In conclusion, our project was an excellent medium to understand working with the following aspects of working with Data:

1. Collection of data from web sources in different formats and making them readable and storable in our database systems
2. Using various web-enabled built in web scrapers and choosing the best of them to scrape useful data off the website (Grepser, Instant Data Scraper, Agenty)
3. Finding differently structured databases and cleaning them to make them easy to work with.
4. Storage of the entire database into an appropriate database management system. Here, we explored working with MongoDB as we dealt with JSON structured data.
5. Analyze the data to create useful and clear insights
6. Explored various R packages (jsonlite, mongolite, data.table)

Our project is an attempt to use LinkedIn data in more resourceful ways. As students, LinkedIn serves as one of the most widely used networking sites for job-hunting and uncovering job trends helps us strategize our job-search efficiently. The future scope of our project can be to collect more data from sources like LinkedIn APIs.

REFERENCES:

- [1] LinkedIn By the Numbers: 2017 Statistics | <https://blog.hootsuite.com/LinkedIn-statistics-business/>

- [2] Contributed Packages - cran.r | <https://cran.r-project.org/web/packages/>

- [3] https://www.reddit.com/r/dataisbeautiful/comments/25qjpz/how_many_employees_are_moving_between_companies_oc/chjvd0g/