Most Valued Data Science Skills

Spring 2017

—

Team 2

Jaan Bernberg

Cesar Espitia

Georgia Galanopoulos

Ilya Kats

Michael Muller

Nkasi Nedd

Nnaemezue Obi-Eyisi

Matheesha Thambeliyagoda

*World cloud above was generated by our data analysis team

Project Objective

The objective is to evaluate the current state of data science job market and to analyze and report on the most valued data science skills. The evaluation is based on collecting and analyzing skills requested by employers since having the most commonly requested skills increases probability of finding a desirable position thus making those skills more valuable.

Data Scraping & Collection

After researching and evaluating several potential data sources, we have decided to concentrate on analyzing jobs posted on *Indeed* site. *Indeed* is a world-wide job aggregation site and it is highest traffic job site in the United States, so it can be considered representative of the job market.

Data scraping was based on the code created for a similar project by Yuanyuan Shi (https://github.com/yuanyuanshi/Data_Skills). However, because the code is a year old some parts were outdated due to changes in *Indeed*'s API and had to be tested and modified by our data scraping team. Data scraping script is included in Appendix A.

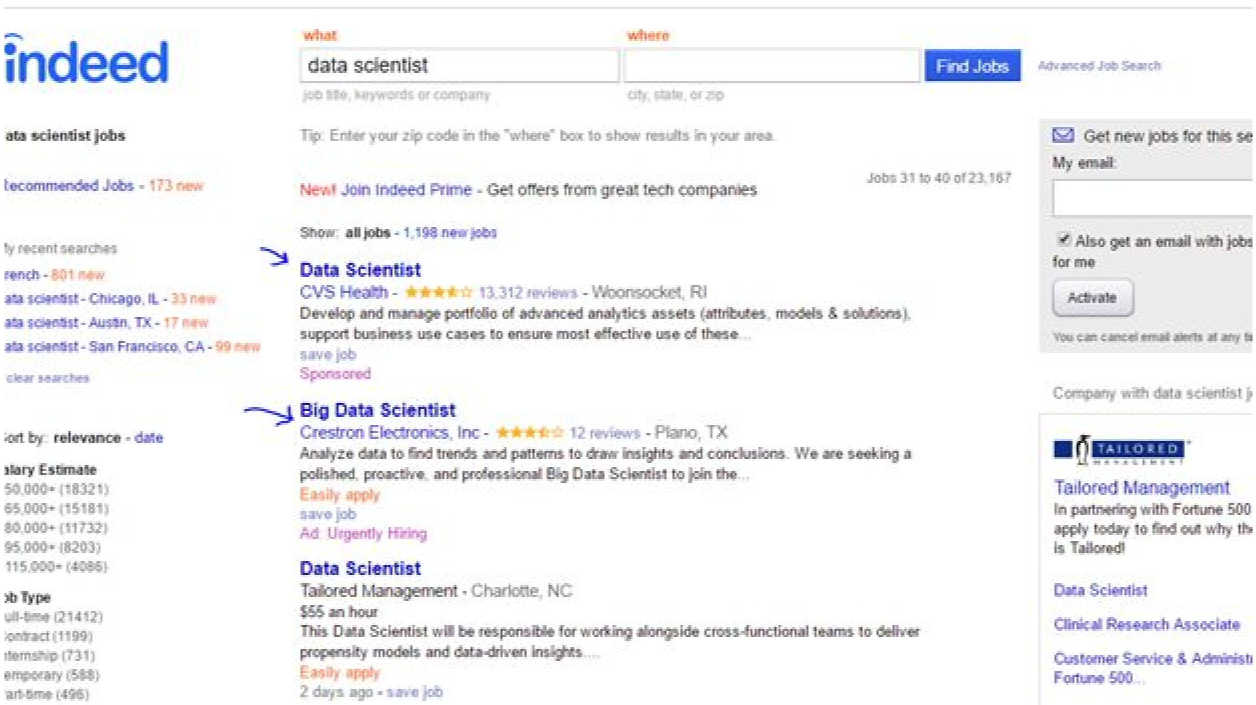
The Python script takes *Indeed*'s URL as an input. Locale is included in the URL, so it is possible to run multiple collections for various cities and countries. The script then calculates the number of pages of results. It opens each page and loops through each listing opening it. There are usually 15 listings to a page. The data scraping is based on a predefined set of keywords organized in broad categories. The script opens each listing and searches for keywords. It returns three variables - skill name, count and rating. Count represents number of listings containing a given skill. Rating is a ratio of listings containing a given skill and all listings searched. The higher the rating the more often the skill is included in the job requirements.

Keywords and Categories Used in Data Collection

```
program_languages = ['bash', 'r', 'python', 'java', 'c++', 'ruby', 'perl',
'matlab', 'javascript', 'scala', 'php']

analysis_software = ['excel', 'tableau', 'd3.js', 'sas', 'spss', 'd3', 'saas',
'pandas', 'numpy', 'scipy', 'sps', 'spotfire', 'scikits.learn', 'splunk',
'powerpoint', 'h2o']

bigdata_tool = ['hadoop', 'mapreduce', 'spark', 'pig', 'hive', 'shark',
'oozie', 'zookeeper', 'flume', 'mahout']
```

```
databases = ['sql', 'nosql', 'hbase', 'cassandra', 'mongodb', 'mysql',
'mssql', 'postgresql', 'oracle db', 'rdbms']

degrees = ['bachelor', 'master', 'b.sc', 'phd', 'mba', 'associates', 'ph.d',
'bachelor\'s', 'master\'s', 'masters', 'bachelors']

field_of_study = ['mathematics', 'statistics', 'computer', 'science',
'computer science', 'engineering', 'math', 'comp sci','stats', 'physics',
'operations research', 'operations', 'research', 'data', 'data science']
```

Below is a sample of *Indeed*'s search results.



The script is fairly slow and dealing with looping through results of a search is tedious. Some searches return significant number of results (for example, search for Atlanta, GA returns close to 100 pages). We have limited collection to 10 pages of results (150 listings). This lets us collect most relevant postings and gives us a representative sample for analysis.

Additional challenge faced by our team was setting up Python environment (installing components, drivers, executables, required libraries, etc.) Our team was new to Python and had to learn quickly on-the-fly.

Our approach also required coming up with a good list of keywords that cover common and not so common skills and that can be easily scraped as well as grouping them into appropriate categories. Soft skills presented unique challenges. Technical skills are usually non-language specific (R will be listed as R in any language); however, soft skills for non-English speaking countries will obviously use native language. The script had an issue searching for keywords containing spaces (for example, “Computer Science”) which limited usefulness of soft skill data collection.

After collection data was uploaded to a SQL database set up by our DBA group using Google Cloud SQL. Database was set up with several tables covering categories, skills and locale with proper relational links between them. See database scheme in the Project Resources section below. SQL code is included in Appendix B.

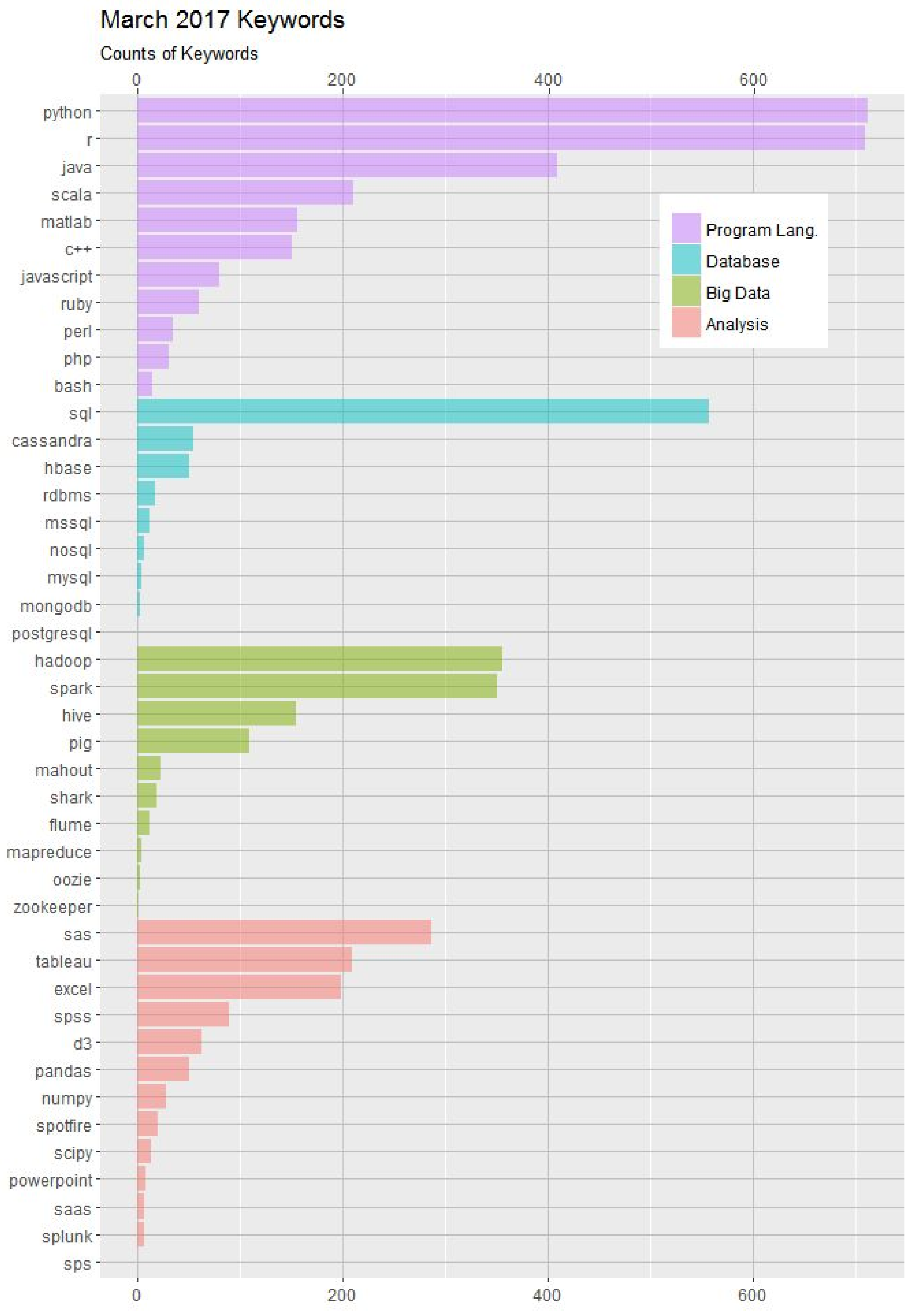
Sample of Data Scraping Script Output (Austin, TX)

```
Skill,Count,Ranking
nosql,3,0.0193548387097
oozie,1,0.00645161290323
pig,17,0.109677419355
hive,8,0.0516129032258
sas,8,0.0516129032258
tableau,22,0.141935483871
hbase,4,0.0258064516129
d3,4,0.0258064516129
cassandra,2,0.0129032258065
```

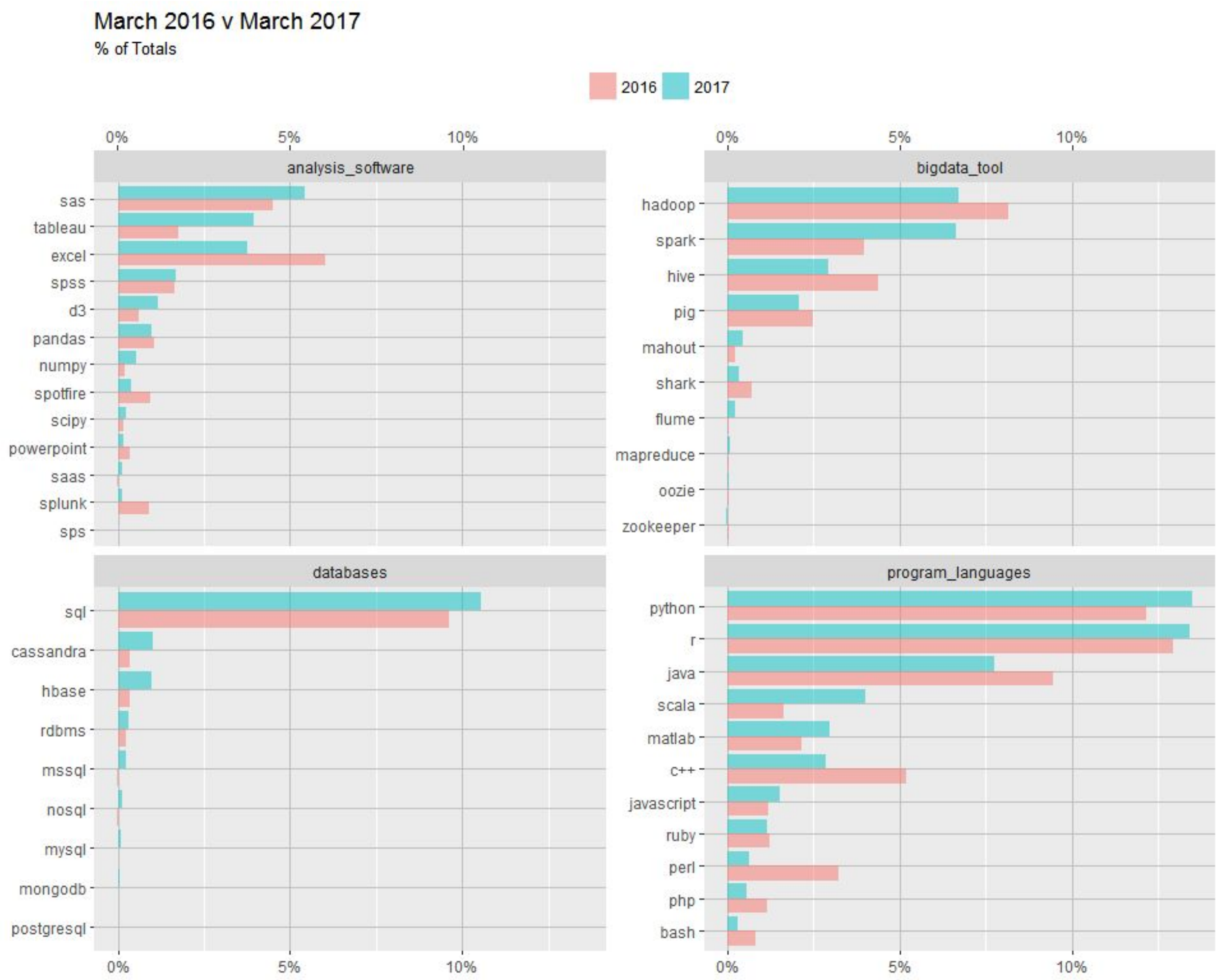
Data Analysis

Our data analysis team used R to analyze collected data in a number of ways. Data analysis R code is included in Appendix C.

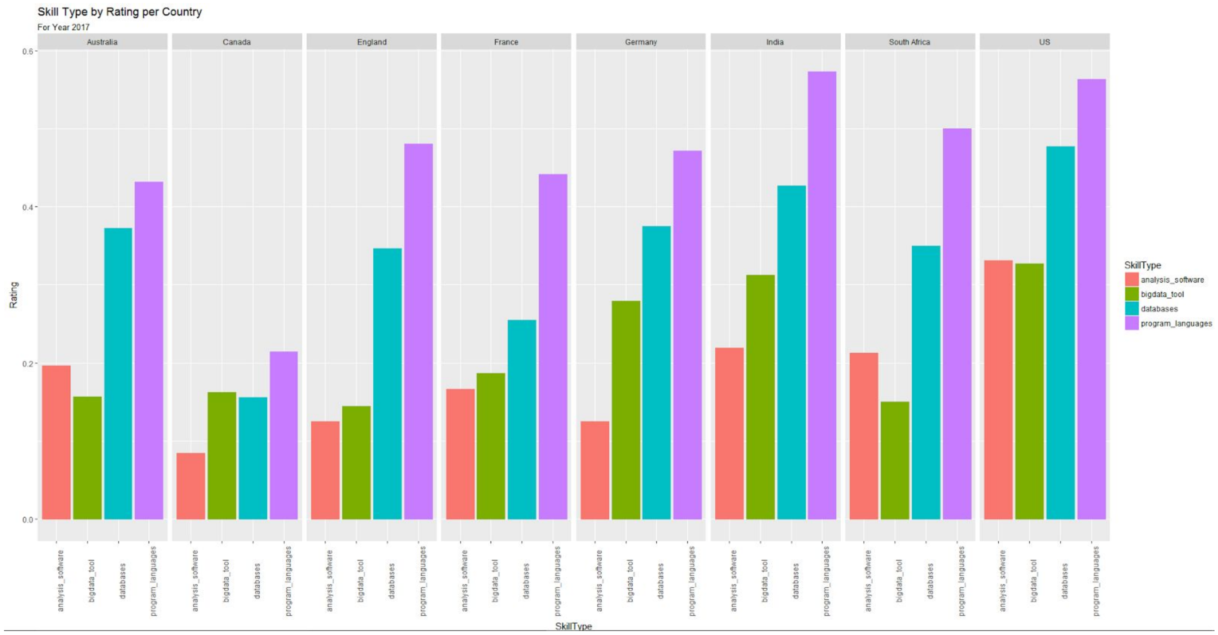
The chart below shows counts of all technical skills for current 2017 data.


March 2017 Keywords
Counts of Keywords
0
200
400
600
python
r
java
scala
matlab
c++
javascript
ruby
perl
php
bash
sql
cassandra
hbase
rdbms
mssql
nosql
mysql
mongodb
postgresql
hadoop
spark
hive
pig
mahout
shark
flume
mapreduce
oozie
zookeeper
sas
tableau
excel
spss
d3
pandas
numpy
spotfire
scipy
powerpoint
saas
splunk
sps
Program Lang.
Database
Big Data
Analysis

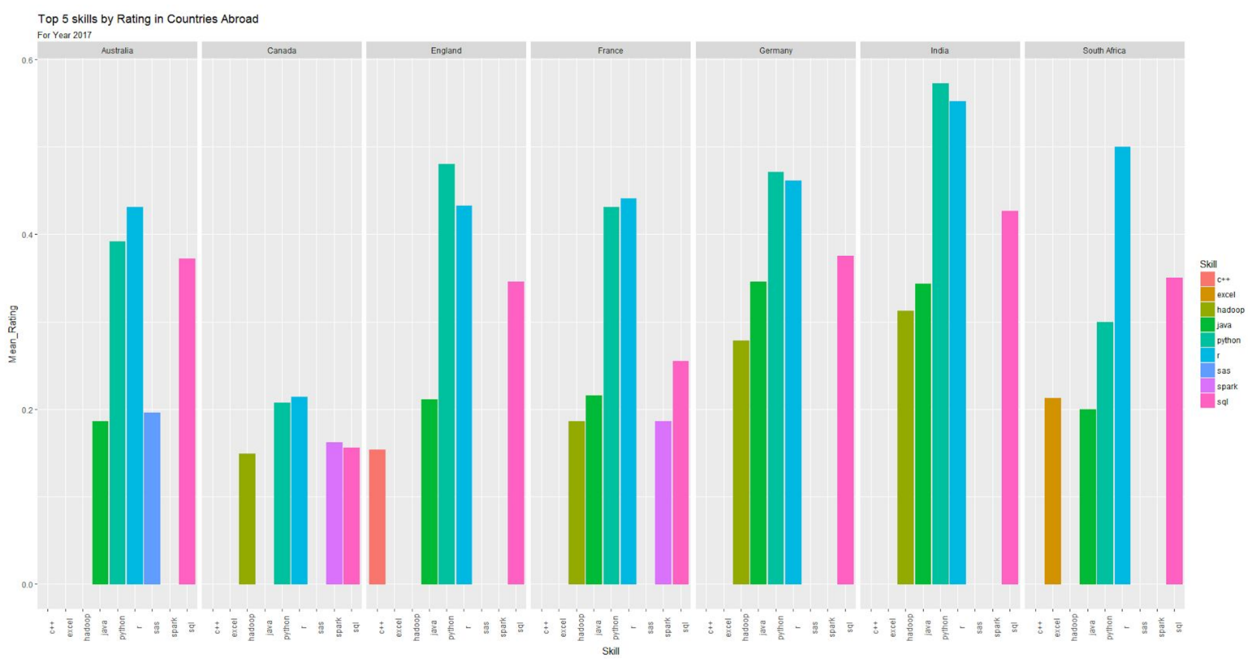
We were able to import Shi's 2016 data. This data contained a larger number of data points, so our data analysis team normalized 2016 and 2017 data in order to compare the two. The chart below shows the comparison.



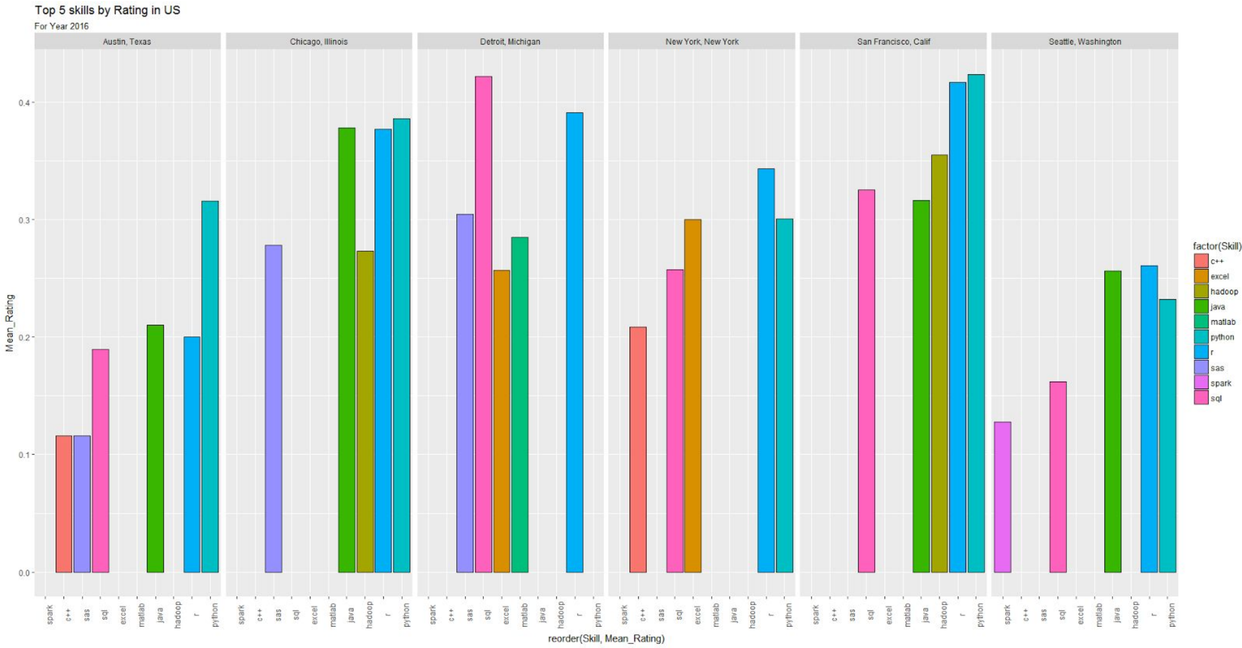
Skill Categories per Country (2017)



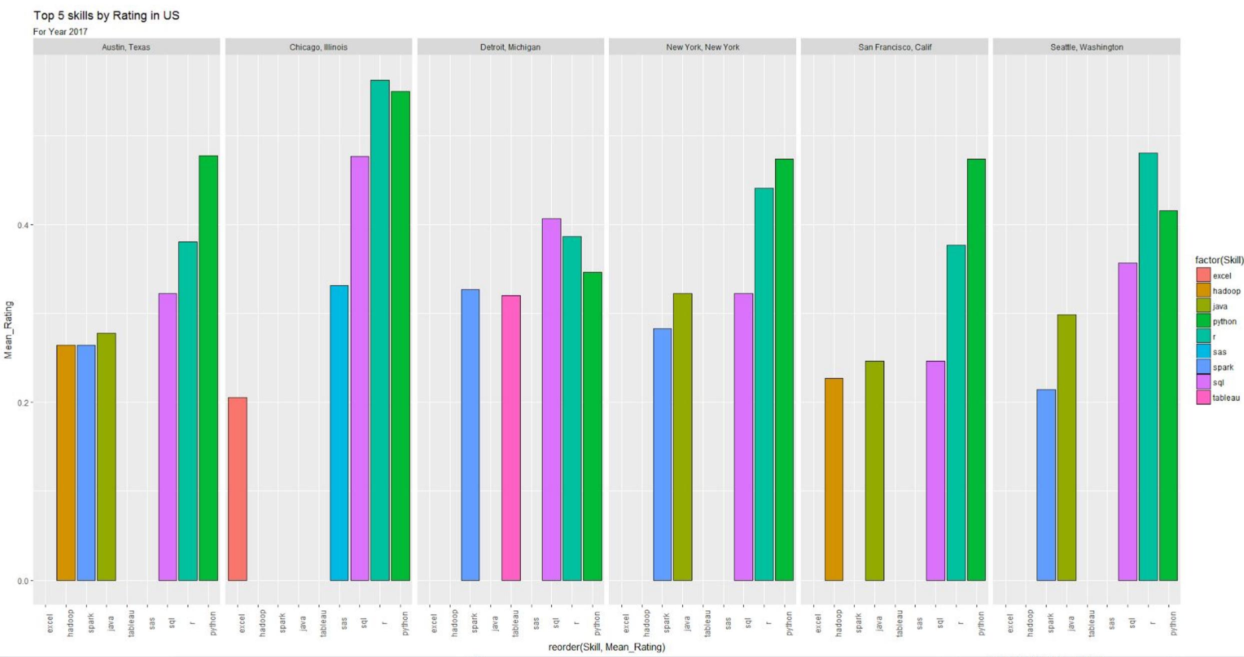
Top 5 Skills per Country (2017)



Top 5 Skills per US City (2016)



Top 5 Skills per US City (2017)



Conclusion

Among programming languages, **Python** and **R** are almost expected base skills for data scientists. **SQL** dominates the database world. **Hadoop** and **Spark** are primary skills for big data. Finally, **SAS**, **Tableau** and **Excel** are top data analysis tools.

Comparing 2016 and 2017 data we notice that even though only one year passed there are some changes in technical skills desired by employers. **Excel** had a sharp drop-off while **Tableau** had an equally big rise in ratings. It is possible that in 2017 Excel is consider a base skill everybody should know without even mentioning it. Another possibility is that **Tableau** which provides a more advanced way to create visuals comparing to **Excel** is rising in popularity due to added functionality and/or marketing hype. **Java**, **C++** and **Perl** all dropped off in 2017 comparing to 2016. It is possible that employers are concentrating on finding **Python** and **R** programmers which are core of data science. In Big Data, **Spark** had a significant rise from 2016 and is ranked similarly to **Hadoop**. Database category observed relatively small changes with **SQL** far outweighing anything else.

When comparing countries or US cities we see that the spread of desired skills is very similar. Perhaps some markets have more data science positions than others, but as we would expect that all seek similar skills. Considering the top 5 skills, for all markets, **SQL**, **R** and **Python** are included among the top. The other 2 skills vary and include **SAS**, **Java**, **Hadoop**, **Spark**, and **Excel**. However, this is likely more due to slight variations in skill counts with some making the top 5 and some just barely missing the top. It is unlikely that any market has a stand out skill that defines just that particular job market.

Project Resources

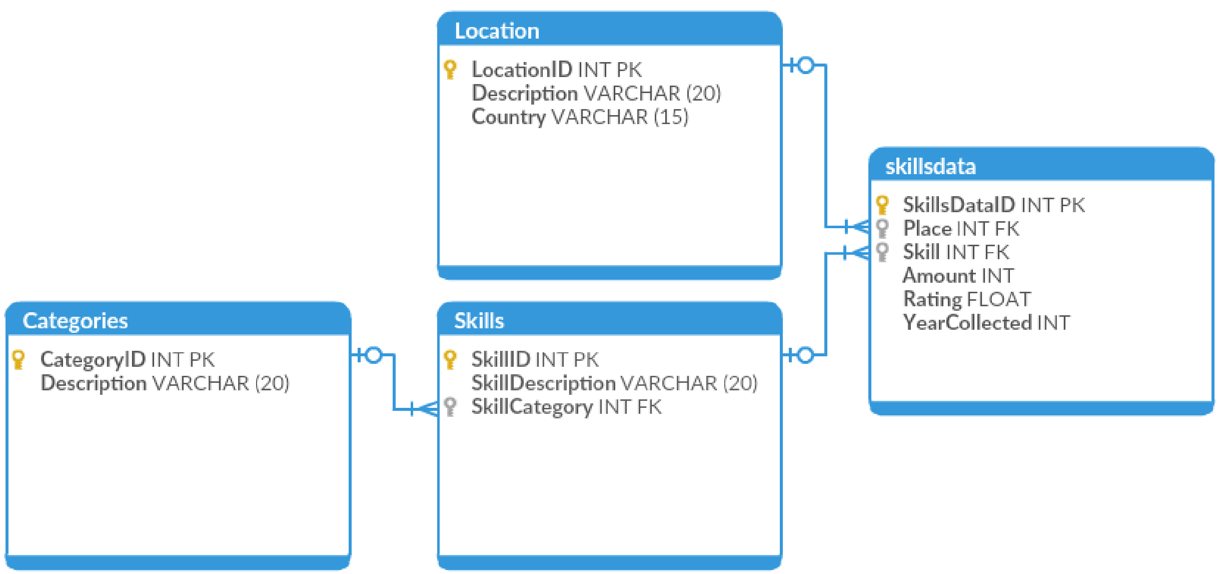
Collaboration Tool: **Slack**

https://data607team2.slack.com

MySQL Server: **Google Cloud SQL**

Hostname: 35.185.104.222
Port: 3306
User: root
Password: data607pw

MySQL Database: **datascienceskills**



File Repository: **GitHub**

https://github.com/NNedd/DATA-607-Project-3

Project Progress

Project Initiation

Following our class meetup on March 9th, our team has connected via email. Following suggestion made during the meetup, we have decided to try using Slack tool to collaborate within the team. During the first days of the project we have started probing potential data sources and collection methods sharing our findings with the team. Nkasi set up a SQL database on Google Cloud SQL. A team meetup has been scheduled for Saturday, March 18.

March 18 Meetup Minutes

- Discussed the type of skills we want to capture and analyze - technical skills or soft skills. Analyzing soft skills presents several issues. Since there is no definitive list of soft skills, the data is hard to collect. Depending on data source, same soft skills may be named differently and various skills may be grouped in categories that are difficult to compare. Finally, same soft skills are generally sought by employers in all fields. They are considered required for general employment especially if we limit it to IT/IS industry. We will concentrate on analyzing the technical skills.
- Discussed potential data sources (see Data Sources section below). Several were identified prior to meetup. The most promising is Indeed.com. Indeed is one of the leading job websites (in fact, it has the highest traffic in the US). It aggregates job listings from thousands of websites and is available in most countries. We believe it a good representational sample of the current job market.
- Discussed potential differences between Data Scientist and Data Analyst positions. Although they can be treated as different positions with different requirements and responsibilities, in the real world they often overlap greatly. We have decided that for our analysis we will treat these positions as equal.
- Discussed possible analyses we would like to perform:
    - Discussed comparing skills between various related fields, for example, Data Science and Computer Science or Software Development, to identify skills most relevant for Data Science and not for other fields. Because of variations in describing titles, positions and job requirements decided not to pursue for this project.

    - Discussed comparing skills between various level - Data Scientist vs. Data Scientist Manager. Again it may be hard to differentiate the two positions, so decided not to pursue for this project.
    - One of the identified data sources contains data for several US metropolitan areas collected almost a year ago in April 2016. Decided to compare current 2017 postings with previously collected data to identify trends.
    - Discussed comparing skills posted for job openings in various countries. We would like to attempt to capture postings from all countries. Most technical skills are independent of language - R, Python, Java, C++, Perl are likely spelled the same in all languages. Alternatively, it is possible to concentrate and compare data from English-speaking countries - US, Canada, UK, South Africa, Australia, etc.
    - Discussed looking at what industries are searching for data scientists and what cities have the most positions listed.
- Discussed how to divide the work to complete this project. One approach is to split the workflow into phases - data identification, collection, tidying, analysis, visualization, presentation - and have the entire team work on one phase at a time merging individual results into the final product. It may be difficult to accomplish everything within the given timeframe. Instead we have decided to split into smaller groups for individual tasks. Some tasks may overlap, so these are just guidelines. Everybody is welcome to contribute to any group at any time.
    - SQL Database Administration: *Nkasi, Nnaemezue*
    - Web Scraping/Data Collection: *Michael, Nnaemezue*
    - Data Tidying/Modeling/Visualization: *Georgia, Jaan*
    - Documentation/Presentation Structure: *Ilya, Cesar*
- **Next steps:**
    - Ability to collect data from Indeed will be tested on Sunday, March 19. Other data sources (such as R package *Rlinkedin*) will be considered, but we should commit to data source(s) by Monday, March 20.
    - Data format will be agreed upon and database structure will be set up, so that we can start creating data modeling workflows.
    - Data collection will continue to fit potential analyses (documented above).
    - Next meetup is being planned for Wednesday, March 22.

March 22 Meetup Minutes

- The team got together to discuss status and progress. Between two meetups the team has decided to concentrate the analysis on the Indeed data. Web scraping and data collection has been completed. 2017 data has been loaded into the database.
- Discussed the next big phase of the project - data analysis:
    - Current data collection has been limited to 150 postings per city due to limitations in API. Historical data from 2016 was collected without this restriction and we have more data available. The difference in data set sizes needs to be accounted for in any comparison. One option is to compare ratings which describes how often a skill is mentioned in the listings. Another option is to extract a random sample from historical data to match the size of the current data set.
    - Discussed creating a general master chart/visualization based on all current data to identify the most common (valued) data science skills.
    - Discussed comparing data for different cities/geographical regions.
- We have originally discussed concentrating on technical skills, but our web scraping team was able to come up with an excellent list of soft skills and collect related data. We may not necessarily use it in the presentation, but it is helpful to be able to test this aspect as well.
- Briefly discussed presentation format. The presentation is being created using Google Docs. We will need to confirm presentation length and exact requirements during class meetup.
- **Next steps:**
    - Historical 2016 data needs to be added to the database.
    - Soft skills need to be added to the database.
    - Concentrating on the next phase of the project - analysis and visualization.
    - Next meetup is being planned for Saturday, March 25.

March 25 Meetup Minutes

- Data collection and data analysis are complete. Discussed our findings.
- Cesar showed a rough draft of the presentation and we discussed it slide by slide. At this point the structure of the presentation is finalized and it just needs some polishing.
- Discussed who will be presenting results in class and what we should be concentrating on.

- Discussed whether if there is anything else we can do for this project. Although we have other ideas (such as scraping data from the *paysa* site or analyzing soft skills), given time constraints we decided to concentrate on finalizing existing findings.
- **Next steps:**
    - Finalize the presentation slides.
    - Finalize documentation.
    - Submit assignment via BlackBoard.
    - Next meetup is being planned for Sunday, March 26, to go over finalized slides.

March 26 Presentation Meetup Minutes

- Documentation/report has been finalized. Presentation slides have been finalized.
- Reviewed slides for minor adjustments.
- Submitted slides and report to the team for final review.
- **Next steps:**
    - Next meetup is being planned for early next week to prepare for class presentation.

Other Data Sources, Tools & Methods Considered

- CrowdFlower 2016 Data Science Report. http://visit.crowdflower.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf.
- *Data Scientist Core Skills* blog post by Mitchell A. Sanders at Data Science Central. http://www.datasciencecentral.com/profiles/blogs/data-scientist-core-skills
- NYC Jobs data set containing current job postings available on the City of New York's official jobs site. https://data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t
- Data Skills analysis by Yuanyuan Shi using Indeed data. Contains historical data from April 2016 for several US cities. https://github.com/yuanyuanshi/Data_Skills
- R Package *Rlinkedin* - a series of functions that allow users to access the 'LinkedIn' API to get information about connections, search for people and jobs, share updates with their network, and create group discussions. https://cran.r-project.org/web/packages/Rlinkedin/
- *Web Scraping Indeed for Key Data Science Job Skills* by Jesse Steinweg-Woods, Ph.D. https://jessesw.com/Data-Science-Skills/

APPENDIX A: Data Scraping Code

```
import re
from nltk.corpus import stopwords
from goose import Goose
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.firefox.firefox_binary import FirefoxBinary
import time
import requests
import random
import pandas as pd
import matplotlib.pyplot as plt
import os

print os.getcwd()
count = 0
count2= 0
fp = webdriver.FirefoxProfile()
fp.set_preference("http.response.timeout", 5)
fp.set_preference("dom.max_script_run_time", 5)

#I get these keywords from the first page search result of data scientist at
indeed; they're not whole but already tell a story.
program_languages=['visualization','communication',"data
driven","analysis","analytical","visual","statistics","mathematics","leadership
","senior","developer","programmer","firmware","software","solving","critical
thinking","translate","translation","scientific","reasoning","query","mastery",
"curious","creative","inquisitive","persuasive","communicative","communication"
,"practices","manage","derive","development","articulate","insight","decision",
"challenge","diverse","diversity"]
analysis_software=['Bachelor','Master','B.sc','Phd','
MBA','Ph.D','MSc','associates']
bigdata_tool=[]
databases=['Mathematics', 'Statistics', 'Computer
Science','Engineering','Math','Comp Sci','Stats','Physics','Operations
Research','Data Science']
overall_dict = program_languages + analysis_software + bigdata_tool + databases
# the following two functions are for webpage text processing to extract the
skill keywords.
def keywords_extract(url):
    g = Goose()
    article = g.extract(url=url)
    text = article.cleaned_text
    text = re.sub("[^a-zA-Z+3]"," ", text) #get rid of things that aren't
words; 3 for d3 and + for c++
    text = text.lower().split()
    stops = set(stopwords.words("english")) #filter out stop words in english
language
    text = [w for w in text if not w in stops]
```

```
    text = list(set(text))
    keywords = [str(word) for word in text if word in overall_dict]
    return keywords

#for this function, thanks to this
blog:https://jessesw.com/Data-Science-Skills/
def keywords_f(soup_obj):
    for script in soup_obj(["script", "style"]):
        script.extract() # Remove these two elements from the BS4 object
    text = soup_obj.get_text()
    lines = (line.strip() for line in text.splitlines()) # break into line
    chunks = (phrase.strip() for line in lines for phrase in line.split("  "))
# break multi-headlines into a line each
    text = ''.join(chunk for chunk in chunks if chunk).encode('utf-8') # Get
rid of all blank lines and ends of line
    try:
        text = text.decode('unicode_escape').encode('ascii', 'ignore') # Need
this as some websites aren't formatted
    except:
        return
    text = re.sub("[^a-zA-Z+3]"," ", text)
    text = re.sub(r"([a-z])([A-Z])", r"\1 \2", text) # Fix spacing issue from
merged words
    text = text.lower().split()  # Go to lower case and split them apart
    stop_words = set(stopwords.words("english")) # Filter out any stop words
    text = [w for w in text if not w in stop_words]
    text = list(set(text)) #only care about if a word appears, don't care about
the frequency
    keywords = [str(word) for word in text if word in overall_dict] #if a skill
keyword is found, return it.
    return keywords


base_url = "http://www.indeed.com"
#change the start_url can scrape different cities.
start_url =
"https://www.indeed.com/jobs?q=data+scientist&l=San+Francisco%2C+CA"
resp = requests.get(start_url)
start_soup = BeautifulSoup(resp.content)
urls = start_soup.findAll('a',{'rel':'nofollow','target':'_blank'}) #this are
the links of the job posts
urls = [link['href'] for link in urls]
num_found = start_soup.find(id = 'searchCount').string.encode('utf-8').split()
#this returns the total number of results
num_jobs = num_found[-1].split(',')
if len(num_jobs)>=2:
    num_jobs = int(num_jobs[0]) * 1000 + int(num_jobs[1])
else:
    num_jobs = int(num_jobs[0])
num_pages = num_jobs/10 #calculates how many pages needed to do the scraping
job_keywords=[]
print 'There are %d jobs found and we need to extract %d
pages.'%(num_jobs,num_pages)
print 'extracting first page of job searching results'
```

```
# prevent the driver stopping due to the unexpectedAlertBehaviour.
webdriver.DesiredCapabilities.FIREFOX["unexpectedAlertBehaviour"] = "accept"
get_info = True
driver=webdriver.Firefox(firefox_profile=fp)
# set a page load time limit so that don't have to wait forever if the links
are broken.

#driver.set_page_load_timeout(5)
for i in range(len(urls)):
    count += 1
    print count
    print 'break point a \n \n \n \n '
    get_info = True
    try:
        driver.get(base_url+urls[i])
    except TimeoutException:
        get_info = False
        continue
    j = random.randint(1000,1300)/1000.0
    time.sleep(j) #waits for a random time so that the website don't consider
you as a bot
    if get_info:
        try:
            print count
            print 'berak point b \n \n \n \n'
            soup=BeautifulSoup(driver.page_source)
            print 'extracting %d job keywords...' % i
            single_job = keywords_f(soup)
            print single_job,len(soup)
            print driver.current_url
            job_keywords.append([driver.current_url,single_job])
        except:
            pass
#driver.set_page_load_timeout(35)
for k in range(1,10):
#this 5 pages reopen the browser is to prevent connection refused error.
    if k%5==0:
        print 'BREAKING CONNECTION FOR A SEC \n \n \n \n '
        driver.quit()
        driver=webdriver.Firefox()
        #driver.set_page_load_timeout(35)
    current_url = start_url + "&start=" + str(k*10)
    print 'extracting %d page of job searching results...' % k
    print count
    print 'break point C \n \n \n \n \n'
    resp = requests.get(current_url)
    current_soup = BeautifulSoup(resp.content)
    current_urls =
current_soup.findAll('a',{'rel':'nofollow','target':'_blank'})
    current_urls = [link['href'] for link in current_urls]
    print len(current_urls)
    count2 = 0
    for i in range(len(current_urls)):
        get_info = True
```

```
        count2+=1
        try:
            driver.get(base_url + current_urls[i])
        except TimeoutException:
            get_info = False
            continue
        j = random.randint(1500,2200)/1000.0
        time.sleep(j) #waits for a random time
        if get_info:
            try:
                soup=BeautifulSoup(driver.page_source)
                print count2
                print 'extracting %d job keywords...' % i
                print count2
                single_job = keywords_f(soup)
                print count2
                print single_job,len(soup)
                print count2
                print driver.current_url
                job_keywords.append([driver.current_url,single_job])
            except:
                pass
# use driver.quit() not driver.close() can get rid of the opening too many
files error.
driver.quit()
skills_dict = [w[1] for w in job_keywords]
dict={}
for words in skills_dict:
    for word in words:
        if not word in dict:
            dict[word]=1
        else:
            dict[word]+=1
Result = pd.DataFrame()
Result['Skill'] = dict.keys()
Result['Count'] = dict.values()
Result['Ranking'] = Result['Count']/float(len(job_keywords))

Result.to_csv('CA_SoftSkills_2017.csv',index=False)
```

APPENDIX B: Data Collection Code

Database and Table Creation SQL Code

```
CREATE SCHEMA IF NOT EXISTS datascienceskills;

USE datascienceskills;

DROP TABLE IF EXISTS skillsdata;
DROP TABLE IF EXISTS Skills;
DROP TABLE IF EXISTS Categories;Location
DROP TABLE IF EXISTS Location;

# Categories table to store information about categories of skills.
CREATE TABLE Categories
	(CategoryID INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
	Description VARCHAR(20) NOT NULL);

# Skills table to store data science skills.
CREATE TABLE Skills
	(SkillID INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
	SkillDescription VARCHAR(15) NOT NULL,
	SkillCategory INT,
	CONSTRAINT FOREIGN KEY (SkillCategory)
		REFERENCES Categories (CategoryID)
		ON DELETE SET NULL);

# Location table to store data about where jobs are found.
CREATE TABLE Location
	(LocationID INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
	Description VARCHAR(20) NOT NULL,
	Country VARCHAR (15) NOT NULL);

# SkillsData table to store data about skills found.
CREATE TABLE skillsdata
	(SkillsDataID INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
	Place INT,
	Skill INT,
	Amount INT,
	Rating FLOAT,
	YearCollected INT,
	CONSTRAINT FOREIGN KEY (Skill)
		REFERENCES Skills (SkillID)
		ON DELETE SET NULL,
	CONSTRAINT FOREIGN KEY (Place)
		REFERENCES Location (LocationID)
		ON DELETE SET NULL);
```

Data Import R Code

```
library(RMySQL)
library(dplyr)

connection <- dbConnect(MySQL(), user='root', password='data607pw',
host='35.185.104.222', dbname='datascienceskills')

categories <- c('program_languages', 'analysis_software', 'bigdata_tool',
'databases')

program_languages <- c('bash', 'r', 'python', 'java', 'c++', 'ruby', 'perl',
'matlab', 'javascript', 'scala', 'php')
analysis_software <- c('excel', 'tableau', 'd3.js', 'sas', 'spss', 'd3',
'saas', 'pandas', 'numpy', 'scipy', 'sps', 'spotfire', 'scikits.learn',
'splunk', 'powerpoint', 'h2o')
bigdata_tool <- c('hadoop', 'mapreduce', 'spark', 'pig', 'hive', 'shark',
'oozie', 'zookeeper', 'flume', 'mahout')
databases <- c('sql', 'nosql', 'hbase', 'cassandra', 'mongodb', 'mysql',
'mssql', 'postgresql', 'oracle db', 'rdbms')

# Convert Categories to Data Frame and write to database
categories_df <- as.data.frame(categories)
names(categories_df) <- "Description"
dbWriteTable(conn=connection, name='Categories', value=categories_df,
overwrite=FALSE, append=TRUE, row.names=0)

# Convert Program Languages to Data Frame and write to database
progam_languages_df <- data.frame(SkillDescription = program_languages,
      SkillCategory = 1)
dbWriteTable(conn=connection, name='Skills', value=progam_languages_df,
overwrite=FALSE, append=TRUE, row.names=0)


# Convert Analysis Software to Data Frame and write to database
analysis_software_df <- data.frame(SkillDescription = analysis_software,
      SkillCategory = 2)
dbWriteTable(conn=connection, name='Skills', value=analysis_software_df,
overwrite=FALSE, append=TRUE, row.names=0)


# Convert Big Data tools to Data Frame and write to database
bigdata_tool_df <- data.frame(SkillDescription = bigdata_tool,
      SkillCategory = 3)
dbWriteTable(conn=connection, name='Skills', value= bigdata_tool_df,
overwrite=FALSE, append=TRUE, row.names=0)


# Convert Databases to Data Frame and write to database
databases_df <- data.frame(SkillDescription = databases, SkillCategory = 4)
dbWriteTable(conn=connection, name='Skills', value=databases_df,
```

```
      overwrite=FALSE, append=TRUE, row.names=0)

# Read in All skills Data from sql database
skillsData <- dbGetQuery(connection, "SELECT * FROM Skills;")


# Setup Austin for data
Location_df <- data.frame(Description = "Austin, Texas", Country = "US")
dbWriteTable(conn=connection, name='Location', value=Location_df,
      overwrite=FALSE, append=TRUE, row.names=0)

URL_austin2017 <-
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/Austin_TX_0416.csv"

Austin2017_data <- read.csv(URL_austin2017)

M1 <- merge(Austin2017_data, skillsData,
      by.x = "Skill", by.y = "SkillDescription")

Austin2017_df <- select(m1, SkillID, Count, Ranking)
Austin2017_df['Place'] <- 1

Austin2017_df <- select(Austin2017_df, SkillID, Place, Count, Ranking)
Austin2017_df <- rename(Austin2017_df, Skill = SkillID,
      Amount = Count, Rating = Ranking)
dbWriteTable(conn=connection, name='skillsdata', value=Austin2017_df,
overwrite=FALSE, append=TRUE, row.names=0)

# ----------------------------------------------------------------------------
# Setup Chicago for data

Location_df <- data.frame(Description = "Chicago, Illinois", Country = "US")
dbWriteTable(conn=connection, name='Location', value=Location_df,
      overwrite=FALSE, append=TRUE, row.names=0)

URL_chicago2017 <-
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/Chicago_IL_0418.csv"

Chicago2017_data <- read.csv(URL_chicago2017)

m1<- merge(Chicago2017_data, skillsData,
      by.x = "Skill", by.y = "SkillDescription")

Chicago2017_df <- select(m1, SkillID, Count, Ranking)
Chicago2017_df['Place'] <- 2

Chicago2017_df <- select(Chicago2017_df, SkillID, Place, Count, Ranking)
Chicago2017_df <- rename(Chicago2017_df, Skill = SkillID,
      Amount = Count, Rating = Ranking)
dbWriteTable(conn=connection, name='skillsdata', value=Chicago2017_df,
overwrite=FALSE, append=TRUE, row.names=0)
```

```
# ----------------------------------------------------------------------------
# Setup Detroit for data

Location_df <- data.frame(Description = "Detroit, Michigan", Country = "US")
dbWriteTable(conn=connection, name='Location', value=Location_df,
      overwrite=FALSE, append=TRUE, row.names=0)

URL_Detroit2017 <-
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/DT_MI_2017.csv"

Detroit2017_data <- read.csv(URL_Detroit2017)

m1<- merge(Detroit2017_data, skillsData,
      by.x = "Skill", by.y = "SkillDescription")

Detroit2017_df <- select(m1, SkillID, Count, Ranking)
Detroit2017_df['Place'] <- 3

Detroit2017_df <- select(Detroit2017_df, SkillID, Place, Count, Ranking)
Detroit2017_df <- rename(Detroit2017_df, Skill = SkillID,
      Amount = Count, Rating = Ranking)
dbWriteTable(conn=connection, name='skillsdata', value=Detroit2017_df,
overwrite=FALSE, append=TRUE, row.names=0)

# ----------------------------------------------------------------------------
# Setup remaining data

Locations <- c("N/A", "N/A", "N/A", "N/A", "New York, New York", "Paris",
"Seattle, Washington", "San Francisco, California", "Sydney", "Toronto,
Ontario")

Countries <- c("England", "Germany", "India", "South Africa", "US", "France",
"US", "US", "Australia", "Canada")

Location_df <- data.frame(Description = Locations, Country = Countries)

dbWriteTable(conn=connection, name='Location', value=Location_df,
      overwrite=FALSE, append=TRUE, row.names=0)

URLs <-
c("https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_resul
ts_150%20cases_per_city/England_2017.csv",
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/Germany_0418.csv",
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/India_0418.csv",
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/SouthAfrica_0418.csv",
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/NY_NY_2017.csv",
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/Paris_2017.csv",
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
```

```
_150%20cases_per_city/SE_WA_2017.csv",
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/SF_CA_2017.csv",
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/Sydney_AU_2017.csv",
"https://raw.githubusercontent.com/NNedd/DATA-607-Project-3/master/2017_results
_150%20cases_per_city/Toronto_ON_2017.csv")

i <- 1
for (i in 1:10)
{
	location_data <- read.csv(URLs[i])
	m1<- merge(location_data, skillsData,
		by.x = "Skill", by.y = "SkillDescription")
	locationskillsdf <- select(m1, SkillID, Count, Ranking)
	locationskillsdf['Place'] <- i+3
	locationskillsdf <- select(locationskillsdf, SkillID,
		Place, Count, Ranking)
	locationskillsdf <- rename(locationskillsdf, Skill = SkillID,
		Amount = Count, Rating = Ranking)
	dbWriteTable(conn=connection, name='skillsdata', value=locationskillsdf,
		overwrite=FALSE, append=TRUE, row.names=0)
}

all_data <- dbGetQuery(connection,
	"SELECT Location.Description, Location.Country, Skills.SkillDescription,
		Categories.Description, skillsdata.Amount, skillsdata.Rating
	FROM skillsdata
	LEFT JOIN (Skills
	LEFT JOIN Categories
		ON Skills.SkillCategory = Categories.CategoryID, Location)
		ON (Skills.SkillID = skillsdata.Skill AND
			Location.LocationID = skillsdata.Place);")
dbDisconnect(connection)
```

APPENDIX C: Data Analysis Code

2017 Data Plot and 2016 vs. 2017 Comparison

```
library(dplyr)
library(tidyr)
library(RCurl)
library(RMySQL)
library(ggplot2)

#------ Obtain Data ----------------------------------------------------------
con <- dbConnect(RMySQL::MySQL(),
                 dbname = "datascienceskills",
                 host = "35.185.104.222",
                 port = 3306,
                 user = "root",
                 password = "data607pw")

all_data <- dbGetQuery(con, "SELECT Location.Description, Location.Country,
Skills.SkillDescription, Categories.Description, skillsdata.Amount,
skillsdata.Rating, skillsdata.YearCollected
                       FROM skillsdata LEFT JOIN (Skills LEFT JOIN Categories
ON Skills.SkillCategory = Categories.CategoryID, Location)
                       ON (Skills.SkillID = skillsdata.Skill AND
Location.LocationID = skillsdata.Place);")
dbDisconnect(con)

#------ 2017 data for plotting -----------------------------------------------
my_plot_data <- all_data[,3:7] %>%
  group_by(SkillDescription, Description, YearCollected) %>%
  select(SkillDescription, Description, Amount, YearCollected) %>%
  summarise(Amount = sum(Amount)) %>%
  spread(YearCollected, Amount)

# get totals of colums for denom:
tot_16 <- sum(my_plot_data$`2016`, na.rm = TRUE)
tot_17 <- sum(my_plot_data$`2017`, na.rm = TRUE)

# calculate rate columns:
my_plot_data <- my_plot_data %>%
  mutate(rt_2016 = `2016`/tot_16,
         rt_2017 = `2017`/tot_17) %>%
  mutate(All_16 = `2016`, All_17 = `2017`) %>%
  select(SkillDescription, Description, All_16, rt_2016, All_17, rt_2017)

# replace NA's with zeros
my_plot_data[is.na(my_plot_data)] <- 0
```

```
# arrange data from high to low on All_17, then by Description
my_plot_data <- my_plot_data %>% arrange(Description, All_17)

# set plot data as arranged:
my_plot_data$SkillDescription <- factor(my_plot_data$SkillDescription,
                                        my_plot_data$SkillDescription)

#------ Year Over Year data for plotting -------------------------------------

# arrange data from high to low on All_17, then by Description
my_plot_data_yoy <- my_plot_data %>% ungroup() %>%
  mutate(SkillDescription = as.character(SkillDescription)) %>%
  arrange(All_17) %>%
  select(SkillDescription, Description, starts_with("rt_"))

# set plot data as arranged:
my_plot_data_yoy$SkillDescription <- factor(my_plot_data_yoy$SkillDescription,

as.character(my_plot_data_yoy$SkillDescription))
# tidy for plot
my_plot_data_yoy <- my_plot_data_yoy %>%
  gather("Year", "% of Totals", starts_with("rt_"))

# for labels, later on.
descr_labels <- c("Analysis","Big Data", "Database", "Program Lang.")

#------ First plot, Bar Plot of Key Words by Description ---------------------
my_plot <- ggplot(my_plot_data, aes(x=SkillDescription, y=All_17))

my_plot + geom_bar(stat = "identity", alpha = .5, aes(fill = Description)) +
  coord_flip() +
  theme(panel.grid.major = element_line(colour = "gray"),
        legend.position = c(.8, .85),
        legend.title = element_blank(),
        axis.title.x = element_blank(),
        axis.title.y = element_blank()) +
  ggtitle("March 2017 Keywords", subtitle = 'Counts of Keywords') +
  scale_y_continuous(sec.axis = dup_axis()) +
  guides(fill = guide_legend(reverse=TRUE)) +
  scale_fill_discrete(name = "Area",
                      labels = descr_labels)


#------ 2nds plot, Bar Plot of Key Words YoY ---------------------------------
my_yoy_plot <- ggplot(my_plot_data_yoy, aes(x = SkillDescription,
                                            y = `% of Totals`,
                                            fill = `Year`))

my_yoy_plot + geom_bar(stat = "identity", alpha = .5, position = "dodge") +
  coord_flip() +
  ggtitle("March 2016 v March 2017", subtitle = '% of Totals') +
  scale_y_continuous(sec.axis = dup_axis(), labels = scales::percent) +
  theme(panel.grid.major = element_line(colour = "gray"),
        legend.position = "top",
```

```
        legend.title = element_blank(),
        axis.title.x = element_blank(),
        axis.title.y = element_blank()) +
  scale_fill_discrete(name = "Year", labels = c("2016","2017")) +
  facet_wrap(~Description, scales = "free_y")
```

Word Cloud

```
library(wordcloud2)

my_cloud_words <- all_data[,3:7] %>%
  filter(YearCollected == 2017) %>%
  select(SkillDescription, Amount) %>%
  group_by(SkillDescription) %>%
  summarise(Amount = sum(Amount)) %>%
  arrange(desc(Amount)) %>% as.data.frame()

rownames(my_cloud_words) <- my_cloud_words$SkillDescription

wordcloud2(my_cloud_words, size = 1.5,
           minRotation = -pi/6,
           maxRotation = -pi/6,
           rotateRatio = 1)
```

Top Skills by Country and US City

```
library(DBI)
library(RMySQL)
library(tidyr)
library(dplyr)
library(ggplot2)


connection = dbConnect(MySQL(), user='root', password='data607pw',
host='35.185.104.222', dbname='datascienceskills')

data = dbGetQuery(connection,
      "SELECT Location.Description, Location.Country,
            Skills.SkillDescription, Categories.Description,
            skillsdata.Amount, skillsdata.Rating, skillsdata.YearCollected
      FROM skillsdata
      LEFT JOIN (Skills
      LEFT JOIN Categories
            ON Skills.SkillCategory = Categories.CategoryID, Location)
            ON (Skills.SkillID = skillsdata.Skill
                  AND Location.LocationID = skillsdata.Place);")

colnames(data) = c("Location", "Country", "Skill",
      "SkillType", "Amount", "Rating", "Year")
```

```
# Let's take the Countries and make stats for each of them, okay?
# Non-US countries
# Make a chart with mean Amount and mean Rating for each skill
# No Year comparisons included because only 2017 data available for non-US
countries
data %>%
  filter(Country != "US") %>%
  group_by(Country, Location, Skill) %>%
  summarise(Mean_Amount = mean(Amount),
            Mean_Rating = mean(Rating)) %>%
  group_by(Country) %>%
  top_n(5, Mean_Rating) %>%
ggplot(aes(x=Skill, y=Mean_Rating, colour = Skill)) +
  geom_bar( aes(fill= Skill), stat="identity", position=position_dodge())+
  facet_grid(~ Country) +
  labs(title = "Top 5 skills by Rating in Countries Abroad") +
  labs(subtitle = 'For Year 2017')+
  theme(axis.text.x = element_text(angle=90))


# And now we can look at the US data
# and it's a mess side by side with the years in one graph,
data %>%
  filter(Country == "US") %>%
  group_by(Year, Location, Skill) %>%
  summarise(Mean_Amount = mean(Amount),
            Mean_Rating = mean(Rating)) %>%
  group_by(Year, Location) %>%
  top_n(5, Mean_Rating) %>%
ggplot(aes(x= Year, y=Mean_Rating)) +
  geom_bar(aes(fill= factor(Skill)), colour = "black", stat="identity",
position=position_dodge())+
  facet_grid(~ Location) +
  labs(title = "Top 5 skills by Rating in US") +
  labs(subtitle = 'For Years 2017 and 2016')

# So, we can split it up by years
# 2016
data %>%
  filter(Country == "US") %>%
  filter(Year == "2016") %>%
  group_by(Year, Location, Skill) %>%
  summarise(Mean_Amount = mean(Amount),
            Mean_Rating = mean(Rating)) %>%
  group_by(Year, Location) %>%
  top_n(5, Mean_Rating) %>%
ggplot(aes(x= reorder(Skill, Mean_Rating), y=Mean_Rating)) +
  geom_bar(aes(fill= factor(Skill)), colour = "black", stat="identity",
position=position_dodge())+
  facet_grid(~ Location) +
  labs(title = "Top 5 skills by Rating in US") +
  labs(subtitle = 'For Year 2016')+
  theme(axis.text.x = element_text(angle=90))
```

```
# and 2017
data %>%
  filter(Country == "US") %>%
  filter(Year == "2017") %>%
  group_by(Year, Location, Skill) %>%
  summarise(Mean_Amount = mean(Amount),
            Mean_Rating = mean(Rating)) %>%
  group_by(Year, Location) %>%
  top_n(5, Mean_Rating) %>%
ggplot(aes(x= reorder(Skill, Mean_Rating), y=Mean_Rating)) +
  geom_bar(aes(fill= factor(Skill)), colour = "black", stat="identity",
position=position_dodge())+
  facet_grid(~ Location) +
  labs(title = "Top 5 skills by Rating in US") +
  labs(subtitle = 'For Year 2017')+
  theme(axis.text.x = element_text(angle=90))
```