Data Science Skills

Group 1

March 20, 2016

Contents

version Control This document will be versioned to track down the changes and edits. See link below for reference.
current version = version 1.1
version control

Data Science and Data Science Skills

As aspiring **Data Scientists** a very common question is *what skills are important in today's market place?*, Data Science has reputation for rapidly changing and so do the tools. Since we all want to be as well trained as possible at the completion of the CUNY MSDA program, we set out to answer "Which skills are the most valued data science skills?".

Let's begin by looking at what *Data Science* is:

```
(library(xml2))
```

[1] "xml2" "rmarkdown" "stats" "graphics" "grDevices" "utils"
[7] "datasets" "methods" "base"

```
library(XML)
library(stringr)
library(RODBC)
data.science <- read_html("https://en.wikipedia.org/wiki/Data_science")
data.science.html <- htmlTreeParse(data.science, useInternal = TRUE)
data.science.text <- unlist(xpathApply(data.science.html, '//body', xmlValue))
gsub("\\[[0-9].+\\]", "", str_trim(str_extract(data.science.text, "Data Science is.+")))
```

[1] "Data Science is an interdisciplinary field about processes and systems to extract knowledge or insights from data in various forms, either structured or unstructured, which is a continuation of some of the data analysis fields such as statistics, data mining, and predictive analytics, similar to Knowledge Discovery in Databases (KDD)."

Essentially, Data Science is a mix of many fields and diverse skills. It lies at the intersection of math, computer science, and domain expertise.

The Data Science Venn Diagram is Creative Commons licensed as

Attribution-NonCommercial.

While the previous formal definition and visual representation is fine, it is too vague. Data science has many applications in many fields.

If we are ever going to be able to answer our question of **"Which skills are the most valued data science skills?"** and in the process become better **Data Scientists**, we need to find out what these specific skills might be.

Since **Data Science** is fundamentally the process of taking data in any format and gaining insight, it is the perfect process to answer a question about itself.

Approach

Data Science Process Diagram Source : Wikipedia

1. Literature Review
    - We reviewed many resources to establish a general idea of valued Data Science skills
2. Sourcing
    - The best sources were singled out to be scraped for detailed information
    - Three sources proved to yielded the greatest results - Indeed.com, Google Search, and Kaggle Job Board
3. Transforming
    - The collective information from the sources was messy and noisy
    - The pool of words and phrases was processed down to specific skill based words/phrases
4. Store information in Data Base
    - The transformed data was stored in normalized data base
    - The transformed data was made ready for consumption by creating views
5. Visualization
    - Detailed results were converted to a meaningful, understandable, telling and aesthetically pleasing format
6. Lesson Learned

The diagram below is a representation of our overall methodology. Two factors were determinant in how the team approached this project;
Overall duration of project: 2 Weeks
How the work would flow downstream

We were trying to avoid a "waterfall" methodology were each group would have to wait for the results from the group upstream of them. To this effect, we identify .csv format for "Sourcers"" to adhere to and for the "Transformers" to model and build the DB. We also identify "sample.csv" set for "Presenters" to start visualizing the results. We were not entirely successful and we experienced some crush time issues, mostly for the "Transformers".

Team Introduction

Team Lead - Jeff Nieman

- Sourcing and webpage scraping - *"Sourcers"* - Dan F., Scott, Arindam, Valerie, Yadu
- Data assembly and tidying - *"Transformers"* - Armenoush, Keith, Dan B., Rob, Adia
- Data visualization and analysis - *"Presenters"* - Antonio, Christophe, Musa, Daniel, Valerie, Yadu, Dan B.

Team Communication

The communications strategy consisted of *all-hands on deck* team meetings on the first Monday of each week using Webex.

We further communicated through emails, texts, phone calls, and the occasional small sub group meetings throughout the weeks.

Data and code was shared using a combination of github and rpubs.

---

1. Review of literature

At the onset of the project, we all had ideas of what would consistute most valued skills for data scientists but did our due diligence and explore some of recent literatures available on the web to consolidate our understanding.

Shared Sources File

2. Scrapping data from Web

Each member of the "Sourcing" team pursued an individual source of data. We were conscious of the time constraint and we had time box our efforts to have raw data to provide downstream by the end of first week. We were concerns that if we only restrict our efforts to one or two sources and encountered some difficulties in the scrapping process, we would have not enough time to change course.

We identify few sources to mine; Google Search, Indeed.com, Kaggle Job Board, and a few other job postings sites. The general approach was to initiate a search for "data science" terms and collect the results by scrapping, identify the skills, and store the outcome. A preliminary filtering of noise and irrelevant data was done. The result set with agreed upon fields was then provided to the group in charge of transforming the data in a .csv format containing the following attributes: source, URL, Skill Type, Skill, rating (count)

a. Indeed.com

The approach by Dan to obtain data from Indeed is elegant and scalable. The premise is to construct an URL search from combining entries for Data Science Term" + "Skill" + "City" + "Radius which are the parameters of a search request. Using function to encapsulate the code, Dan was able to build a solution that would iterate through various lists for base terms ('Data Scientist', 'Data Analytics'), skill term (stored in IndeedDataScienceSkillsList.csv), cities (stored in IndeedDataScienceCitiesList.csv), radius (25, 50), send these get requests and read the number of jobs returned for the search. The counts with skill terms are stored in .csv file.

This approach offers a clear and concise results. There is no need to scrape html page to find relevant information. Also, this approach is very scalable by varying the entries in the various lists that are used to build URL. Since this approach makes use of a directed search, the "noise" and irrelevant data is reduced in the result set. However, a directed search poses the problem that the results may be biases due to limitation from input parameters. For example, a city included in the search with high concentration of a type of industry may skew the result in favor of specific skill required for this industry.

www.Indeed.com Search Details

b. Google Search

The approach by Scott to obtain data from Google Search is novel and interesting. The premise is to search for the term "data science skills" and collect the resulting set of URL's. The resulting URL's are parse for meaningful key words.

A significant challenge in this approach was to separate relevant and meaningful information from the noise. The advantage of this approach is that it cast of wide net and does not restrict the domain search to Job Posting sites or specific Data Science articles.

Google Search Details

c. Kaggle Job Board

Arindam followed a more traditional text scrapping approach using corpus and document-term matrix. It consists of querying a job hosting site for Data Science jobs and obtaining a list of URL each representing an individual job posting, iterating over this list and retrieving the individual job posting, parsing the body of each job posting identifying keys words (such as 'required skills', 'Requirements', .) and scraping the text and storing each extract into a corpus. Then the corpus is then processed to convert everything to lower-case, remove all punctuation, strip extra white-space, and finally remove "stop-words" of the English language (prepositions, articles, possessive, ...). Finally, a document-term matrix is created. This approach makes use of the TM package. The raw data is then filter to remove "noise" and irrelevant terms. The result set is then stored in .csv file.

This approach highlights the web scrapping and text mining potential of R. It is not dependent on a directive search to obtain the raw data.

Kaggle Job Board Details

d. Other

We had another attempt to extract Data Science skills from job sites (similar approach to Kaggle job Board). For the first job site: www.glassdoor.com, Valerie experienced HTTP error related to SSL (Secure Sockets Layer). The problem could not be resolve by setting sslverify to FALSE nor by updating the Security Certificate on local PC. On the 2nd job site: www.dice.com, Valerie was able to extract the list of job posting URL and compile a .csv of these. However, when iterating the list of URL to retrieve the individual job postings Valerie encountered some error when attempting to "get" the data. An "HTTP/1.1 505 HTTP Version Not Supported" error was received. Time constraints forced Valerie to abandon the attempt.

we have another source of data that was "manually" scrapped that represents the results from a study made based on "LinkedIn" data. This data set was compiled using "LinkedIn" profiles. This data set "Top 20 Skills of a Data Scientist" is meant to be used by the "presenters" team as a comparison with our mined data. For additional information on this data set can be found at:

rjmetrics.com/resources/reports/the-state-of-data-science

3. Transforming & Storing in Data Base

The "Transformers" group has the responsibility to take the raw data, clean it, process it, and store it in MySQL Data base.

- The first step in this process is to standardize the data coming from "Sourcers" group in terms of columns header, structure, consolidation of skill name when possible, and mapping of skill type so that a common methodology could be used to transform the data.
- The 2nd step is to load the standardized .csv files into the staging area of the DB and load the relevant master tables; tbl_source, tbl_skill_type, and tbl_skill
- The 3nd step is to the map the skill to a skill type (high level category) if not previously done and to a skill set.
- The 4th step is to load the data into the data base and apply an initial scaling.
- The 5th step is to apply an algorithm to weight the data to allow for comparison across sources, skill type, skill set.
- The 6th step is to provide presentation-ready views to be consumed by the "Presenters" team.

a. Standardization of .csv files

Once the first .csv files were produced out of the sourcing effort, the "transformers" went to work on processing the raw data. A classification of skills was agreed upon and each .csv file was processed. The following steps were applied for each raw data sets providing from the "transformers" group. Please note that this transformation was done per data set and the integrity of each data set was preserved.

i. Standardize format
   The individual .csv files were modified to have the same structure for header rows and columns.

ii. Identify skill names
   The skill name column was clearly identified, any extra data was removed beyond source and skill name.

iii. Consolidate skill names
   Were applicable, the result for similar skill name were aggregated. For example; "Map/Reduce", "Map_Reduce", or even "MapReduce" were consolidated into one entry.

iv. Map skill name to skill type
   All skill names were assigned one of the five skill type: Business, Communication, Visualization, Math, Programming

The outcome of this preliminary clean-up effort was 4 standardized .csv files, one for each data set (Indeed, Google, Kaggle, rjmetrics). All with the same format: "source id", "skil_type_name", "skill_name", "rating". In addition, the two rows of each of these table consists of the source name (row=1) and main URL where information was found (row=2). The hand-off of these files are done manually at this point by pushing them into a publicly accessible github repository. These files are the starting point of the next transformation process to load the raw data into the staging area of the data base.

Standardization Details

b. Loading the raw data into the staging area of Data Base

To safeguard the integrity of the process, the "Transformers" team design the next step to run uninterrupted and to reload the data base with each run to ensure the integrity of the final data (steps 2 - 6 above).

A mechanism was design to accommodate for scalability (adding a new source) by having a list of source files to be loaded (and URL where file resides) stored in a control .csv file residing in a publicly accessible github repository This control file would also have a go/stop flag to indicate whether the transformation process could proceed (for example, if a data "sourcing" job(s) was running, the go/stop indicator would have been set to "stop"). Unfortunately, due to time constraints we could not implement this mechanism and we had to "hard-code" the retrieval of the .csv files.

In this step we load the data into the staging area of the data base and reload the master tables: tbl_source, tbl_skill_type, and tbl_skill.

```
#################################################################
#
# Read the csv data files and load to normalized database
# ROB
#################################################################

# Establish connection to the database
library(RODBC)
#cnString <- "MySQL_ANSI;SERVER=localhost;DATABASE=skill;UID=root;PASSWORD=CUNYRBridge4!;OPTION=3;" # t
cnString <- "MySQL_ANSI;SERVER=db4free.net;DATABASE=skill;UID=project3;PASSWORD=CUNYRBridge4;OPTION=3;"
db <- odbcConnect(cnString, case="nochange")

#manually input the list of csv files on github
sources <- c("https://raw.githubusercontent.com/LovinSpoonful/IS607-Project3/master/08-barman-data-clean
             "https://raw.githubusercontent.com/LovinSpoonful/IS607-Project3/master/09-briot-data-cleane
             "https://raw.githubusercontent.com/LovinSpoonful/IS607-Project3/master/10-karr-data-cleaned
             "https://raw.githubusercontent.com/LovinSpoonful/IS607-Project3/master/11-fanelli-data-clea

# Import all the source data into the database
sqlQuery(db,"DELETE FROM tbl_import;") # first truncate the imported records table
sqlQuery(db,"DELETE FROM tbl_source;") # also empty the source table

# Loop through every source data file
for (i in 1:length(sources)){

  #record the source of the data in the source table
  dat <- read.csv(file = sources[i], header=FALSE, sep=",")
  source_id <- i
  source_name <- as.character(dat[1,1])  # get the source name from the top row of the data file
  source_url <- as.character(dat[2,1])   # get the source URL from 2nd row
  sSQL <- paste("INSERT INTO tbl_source VALUES (", source_id, ", '", gsub(" ", "_", source_name), "', '
  sqlQuery(db,sSQL)

  #export all the data for this source into the database (the Import table)
  dat <- dat[-1:-3,] #remove the header lines
  source_id <- rep(i,nrow(dat)) # create a column to show which data set these samples came from
  dat <- data.frame(source_id,dat) # add that column to the front of the dataframe
  names(dat) <- c("source_id","skill_type_name","skill_name","rating")  #hardcode the column names.  sh
  sqlSave(db, dat, tablename = "tbl_import", rownames=FALSE, append=TRUE) # write the dataframe to the
}
```

```
#De-duplicate the imported data and setup the Skill Types, Skill Names tables
# retrieve the imported records into a data frame
df <- sqlQuery(db,"SELECT source_id, skill_type_name, skill_name, ROUND(AVG(rating),3) rating FROM tbl_

# write the de-duplicated import records back into the database
sqlQuery(db,"DELETE FROM tbl_import;") # first truncate the imported records table
sqlSave(db, df, tablename = "tbl_import", rownames=FALSE, append=TRUE) # write the dataframe to the myS

# create a master list of skill types (using r)
sqlQuery(db,"TRUNCATE TABLE tbl_data;") # first truncate it
sqlQuery(db,"TRUNCATE TABLE tbl_data_n;") # first truncate it
sqlQuery(db,"TRUNCATE TABLE tbl_skill;") # first truncate the imported records table
sqlQuery(db,"TRUNCATE TABLE tbl_skill_type") # first truncate the table
skill_type_name <- unique(df$skill_type_name,incomparables = FALSE, stringsAsFactors = FALSE)
skill_type_id <- c(1:length(skill_type_name))
skill_type_description <- skill_type_name # enrich later if can get descriptions
df1 <- data.frame(skill_type_id,skill_type_name, skill_type_description) # create a dataframe that matc
sqlSave(db, df1, tablename = "tbl_skill_type", rownames=FALSE, append=TRUE) # write the dataframe to th

# create a master list of skill names (using my sql)
sqlQuery(db,"INSERT INTO tbl_skill (skill_id, skill_type_id, skill_name) SELECT NULL, st.skill_type_id,
```

c. mapping of detail skills to skill set and skill type

The "Transformers" team, after initial analysis of the raw data set coming out of the "sourcing" process, concluded that the granularity of the skills was uneven across the data sets and some mechanism needed to be devised to correct this so that the data could be compared meaningfully. For example, the skills: "Big Data", "Hadoop", "Pig", "Hive" are all assigned the skill set "Big and Distributed Data".

A Master table for this mapping was compiled and can be found at: Mapping Cross-Reference for skill set

```
#SECTION 2 -  skills were mapped to a master set of skill sets

#############################################################################################################
#KEITH F   SKILL SETS AND SKILL -- SET CROSS REFERENCE
# ------------------------------------------------------------
# This R code process the csv file Skill_Keyword_Map.csv into a dataframe
# with two columns: skill_category and skill_name
#
# The input file is in the format of:
#
#     Skill Category,       Skill Keyword
#     --------------        --------------
#     Machine Learning,     "decision trees, neural nets, SVM, clustering, predictive analytics, machin
#
#
# The output will be a dataframe:
#
#  skill_category           skill_keyword
#  --------------          ---------------
#  Machine Learning         decision trees
#  Machine Learning         neural nets
```

```
#  Machine Learning         SVM
#  Machine Learning         clustering
#  Machine Learning         predictive analytics
#  Machine Learning         machine learning
#  Machine Learning         clustering
#
#  This script produces the file called Skill_Category_Xref.csv (loaded to GitHub)
# ------------------------------------------------------------------------------------------

library(stringr)
library(RMySQL)
library(dplyr)

#test
# proj_user <- "root"
# proj_pwd  <- "CUNYRBridge4!"
# proj_db   <- "skill"
# proj_host <- "localhost"

# live
proj_user <- "project3"
proj_pwd  <- "CUNYRBridge4"
proj_db   <- "skill"
proj_host <- "db4free.net"

#####################################################################
# Step 1 - File Parsing
# Parse the Skill_Keyword_Map file located on GitHub
#####################################################################

# load in Skill_Keyword_Map.csv file from GitHub
# this file matches skill keywords with a skill set category

URL <- "https://raw.githubusercontent.com/LovinSpoonful/IS607-Project3/master/Skill_Keyword_Map.csv"

skill_map <- read.csv(URL, header=TRUE, stringsAsFactors = FALSE)[, 1:3]

#initialize an empty dataframe to rbind all the keywords in column format
ret <- data.frame(Raw.Skill = character(), Category = character(), Skill.Type = character())

for (i in 1:nrow(skill_map))  {

  keywords <- data.frame(str_split(skill_map$Skill.Keyword[i], ","))

  keywords$skill_category <- skill_map$Skill.Category[i]
  keywords$Skill.Type     <- skill_map$Skill.Type[i]

  colnames(keywords) <- c("Raw.Skill", "Category", "Skill.Type")

  ret <- rbind(ret, keywords)

}
```

```r
# trim whitespace; convert factor to characater
ret$Raw.Skill <- str_trim(ret$Raw.Skill, side = "both")
ret$Raw.Skill <- as.character(ret$Raw.Skill)

str(ret)

# dedupe in case there are duplicate skills in the file
ret <- distinct(ret)

# create the output file which is the file Skill_Category_Xref.csv

# write.csv(ret, "Skill_Category_Xref.csv", row.names=FALSE)

################################################################################
# Step 2 - MySQL DB: Load tbl_skill_set
# ---------------------------------------
# Using the parsed Skill-Map file, load the skill sets (categories)
# into tbl_skill_set
################################################################################

# establish the connection to the skill DB on db4free.net
skilldb = dbConnect(MySQL(), user=proj_user, password=proj_pwd, dbname=proj_db, host=proj_host)

skill_set <- data.frame(skill_set_name = ret$Category,
                        skill_set_description = NA,
                        skill_type = ret$Skill.Type
)
# we only want the distinct skill_sets (categories) and skill_types
skill_set <- distinct(skill_set)

skill_set$skill_set_id <- rownames(skill_set)
skill_set[,"skill_set_description"] <- as.character(NA)

# convert everything to a character
skill_set[] <- lapply(skill_set, as.character)


# pull down skill types from MySQL
skill_types <-  dbGetQuery(skilldb, "select skill_type_id, skill_type_name from tbl_skill_type")

str(skill_types)

skill_set_insert <-
  skill_set %>%
  inner_join(skill_types, by=c("skill_type" = "skill_type_name")) %>%
  select(skill_set_id, skill_type_id, skill_set_name, skill_set_description)


## delete the table tbl_skills_categories
del <- dbGetQuery(skilldb, "delete from tbl_skill_set")

# write the dataframe to the mySQL table
dbWriteTable(skilldb, value = skill_set_insert, name = "tbl_skill_set", append = TRUE, row.names = NA, I
```

```
rs <- dbGetQuery(skilldb, "select count(*) from tbl_skill_set")

if (rs == 0) print("No Rows Loaded into tbl_skill_set!")


##############################################################################
# Step 3 - MySQL DB: Load tbl_skill_set_xref
# ------------------------------------------
# Load tbl_skill_set_xref Using the parsed Skill-Map file for skills to skill set.
# This DB tables associates a skill to a skill set based on the Skill-Map file.
##############################################################################

# pull down the skill table for the skill_id and name
skills <-  dbGetQuery(skilldb, "select skill_id, skill_name from tbl_skill")

# pull down the skill set table for the sKill_set_id and name
skill_sets <- dbGetQuery(skilldb, "select skill_set_id, skill_set_name from tbl_skill_set")

# use dplr to build the dataframe to insert into tbl_skill_set_xref
skill_set_xref_insert <-
  ret %>%
  inner_join(skills, by=c("Raw.Skill"="skill_name")) %>%
  inner_join(skill_sets, by=c("Category"="skill_set_name"))  %>%
  select(skill_set_id, skill_id)

str(skill_set_xref_insert)

## delete the table tbl_skills_categories
del <- dbGetQuery(skilldb, "delete from tbl_skill_set_xref")

# write the dataframe to the mySQL table
dbWriteTable(skilldb, value = skill_set_xref_insert, name = "tbl_skill_set_xref", append = TRUE, row.nar

rs <- dbGetQuery(skilldb, "select count(*) from tbl_skill_set_xref")

if (rs == 0) print ("No Rows Loaded into tbl_skill_set_xref!")

# close the connection
dbDisconnect(skilldb)
```

d. Loading the data into Data Base and apply initial scaling

We are now ready to load the data into the main table of the data base, the tbl_data table. An initial scaling is applied and is stored in the rating_scalar attribute on the tbl_data table.
This scaling is determined by the following formula for each row (max and min are evaluated by source id):

$$rating\_scalar = \frac{rating - minimum(rating)_{source}}{maximum(rating)_{source} - minimum(rating)_{source}} \times 100$$

```
# SECTION 3:  data was loaded into tbl_data and scaled

##############################################################################
#ROB
```

```
#Populate the normalized data table of all ratings values across all sources
sqlQuery(db,"DELETE FROM tbl_data_n;") # first truncate it
# populate the data table
sSQL <- paste("INSERT INTO tbl_data_n (skill_type_id, skill_set_id, skill_id, source_id, rating) ",
              "SELECT s.skill_type_id, sx.skill_set_id, s.skill_id, i.source_id, i.rating         ",
              "FROM tbl_skill s                                                                   ",
              "JOIN tbl_skill_type st ON s.skill_type_id = st.skill_type_id      ",
              "JOIN tbl_skill_set_xref sx ON s.skill_id = sx.skill_id            ",
              "JOIN tbl_import i ON i.skill_name = s.skill_name;                 ", sep = "")
sqlQuery(db,sSQL)

#To make it easier to create analysis, also create a denormalized table with descriptors
sqlQuery(db,"DELETE FROM tbl_data;") # first truncate it
# populate the table
sSQL <- paste(
  "INSERT INTO tbl_data (skill_type_id, skill_set_id, skill_id, source_id, skill_type_name, skill_set_na
  "SELECT d.skill_type_id, d.skill_set_id, d.skill_id, d.source_id, st.skill_type_name, ss.skill_set_nam
  "FROM tbl_data_n d ",
  "JOIN tbl_skill_type st ON d.skill_type_id = st.skill_type_id ",
  "JOIN tbl_skill_set  ss ON d.skill_set_id  = ss.skill_set_id ",
  "JOIN tbl_skill       s ON d.skill_id      =  s.skill_id ",
  "JOIN tbl_source     so ON d.source_id     = so.source_id;", sep = "")
sqlQuery(db,sSQL)

#copy the skill set id to the skill table
sSQL <- "UPDATE tbl_skill s JOIN tbl_skill_set_xref sx  ON sx.skill_id = s.skill_id  SET s.skill_set_id
sqlQuery(db,sSQL)

skilldb = dbConnect(MySQL(), user=proj_user, password=proj_pwd, dbname=proj_db, host=proj_host)

#calculate the max and min ratings within each source and post to temporary table
df <- dbGetQuery(skilldb, "SELECT source_id, MAX(rating) rating_max, MIN(rating) rating_min FROM tbl_dat
dbSendQuery(skilldb, "DROP TABLE IF EXISTS df_temp;")
dbWriteTable(skilldb, name="df_temp", value=df)

#scale the ratings from 0 to 100
dbSendQuery(skilldb, "UPDATE tbl_data SET rating_scalar = NULL;")
dbSendQuery(skilldb, "
            UPDATE tbl_data d, df_temp t
            SET d.rating_scalar = ROUND((d.rating - t.rating_min) / (t.rating_max - t.rating_min),2)*100
            WHERE d.source_id = t.source_id;")

#since this is a discrete series, set all the minimum values to one, instead of zero
dbSendQuery(skilldb, "UPDATE tbl_data SET rating_scalar = 1 WHERE rating_scalar < 1;")
```

e. Weighting Algorithms

The 3 resulting data sets (Indeed, Google, and Kaggle) resulted in very different results. In order to have meaningful comparison across data set, a weighting algorithm was devised. This algorithm is based on the number of times a particular skill, source, or type appeared within the given data. Similar algorithms were applied to all three sets of data that was collected (weighted by source, by skill type, and skill set).
The following is a detail explanation for the skill type using Google source as an example. The process can be repeated for each data set and for each variable of interest (source, skill set).

The 160+ skills are segregated by the three sources (Google, Indeed, and Kaggle). Each of those sources had skills that were found on those websites, Google had around 39, Indeed around 80, and Kaggle around 25. Within each of those sources there are a varying numbers of "hits" for each of the skills.
(Please note: Indeed results are divided by 4 to make it comparable to other sets in numbers).

Considering Google source, there are about 25 programming skills, 2 math skills, 1 visualization etc. Since there is not an even number across the 5 skill types within Google, they cannot have equal weight. Google will produce more programming skills than math, or business skills because it yields a ratio of 25/39 programming skills while 14/39 for the other 4 types combined. That means that a programming skill with 14 hits is a better skill on Google than a math skill that got 14 hits, because Google is finding more programming skills than math skills. Therefore, we need to weight them so that programming Google skill is better than a math skill on Google with the same rating.

We do this by considering each skill totals per type (25 programming 2 math etc.) and divide it by the total number of skills found by the website (39 total skills). Then for each skill, the rating is adjusted by multiplying it to the ratio corresponding to the skill type (Programming, math, etc.). This will give and edge to the programming skills because they are being multiplied by a higher decimal. This will separate the 14 hits in programming from the 14 hits in math and it will also bring the rating numbers down to similar level.

The same process is repeated for the other two rating systems that we want. We took the number in each source and divide it by the total number of skills. And then the number in each skill set is divided by the total number of skills. We multiply each ratio by their designated source or skill set.

```
# SECTION 4: data was weight ranked
# DANIEL BROOKS

library(RMySQL)
library(dplyr)

# MySQL DB info
proj_user <- "project3"
proj_pwd  <- "CUNYRBridge4"
proj_db   <- "skill"
proj_host <- "db4free.net"

## -----------------------------------------
## Using RMYSQL
## -----------------------------------------

# establish the connection to the skill DB on db4free.net
skilldb = dbConnect(MySQL(), user=proj_user, password=proj_pwd, dbname=proj_db, host=proj_host)

# load the processed source data into a dataframe from the tbl_data
rs <- dbGetQuery(skilldb, "select * from tbl_data")


## -----------------------------------------
## Start the Ratings calculations
## -----------------------------------------
i <- 1
for (i in 1:NROW(rs))
{
  if(rs$source_name[i] == "Indeed")
  {
    rs$rating[i] <- rs$rating[i]/4
  }
```

```
  i <- i + 1
}
#get all google information
google <- subset(rs, rs$source_name == "Google")

#get all indeed information
indeed <- subset(rs, rs$source_name == "Indeed")

#get all kaggle information
kaggle <- subset(rs, rs$source_name == "Kaggle")

#Linkedin information
linkedin <- subset(rs, rs$source_name == "RJMETRICS")

#get all programming information
program <- subset(rs, rs$skill_type_name == "programming")

## WEIGHT SKILL TYPE BY SOURCE
#get the skill type name google
pgoogle <- subset(google, google$skill_type_name == "programming")
mgoogle <- subset(google, google$skill_type_name == "math")
bgoogle <- subset(google, google$skill_type_name == "business")
cgoogle <- subset(google, google$skill_type_name == "communication")
vgoogle <- subset(google, google$skill_type_name == "visualization")

totalgoogle <- NROW(google)

cpgoogle <- NROW(pgoogle)
cmgoogle <- NROW(mgoogle)
cbgoogle <- NROW(bgoogle)
ccgoogle <- NROW(cgoogle)
cvgoogle <- NROW(vgoogle)

pgoogle$weight_skill_type_source <- cpgoogle/totalgoogle
mgoogle$weight_skill_type_source <- cmgoogle/totalgoogle
bgoogle$weight_skill_type_source <- cbgoogle/totalgoogle
cgoogle$weight_skill_type_source <- ccgoogle/totalgoogle
vgoogle$weight_skill_type_source <- cvgoogle/totalgoogle

pgoogle$rating_skill_type_source <- pgoogle$rating * pgoogle$weight_skill_type_source
mgoogle$rating_skill_type_source <- mgoogle$rating * mgoogle$weight_skill_type_source
bgoogle$rating_skill_type_source <- bgoogle$rating * bgoogle$weight_skill_type_source
cgoogle$rating_skill_type_source <- cgoogle$rating * cgoogle$weight_skill_type_source
vgoogle$rating_skill_type_source <- vgoogle$rating * vgoogle$weight_skill_type_source

#get the skill type name indeed
pindeed <- subset(indeed, indeed$skill_type_name == "programming")
mindeed <- subset(indeed, indeed$skill_type_name == "math")
bindeed <- subset(indeed, indeed$skill_type_name == "business")
cindeed <- subset(indeed, indeed$skill_type_name == "communication")
vindeed <- subset(indeed, indeed$skill_type_name == "visualization")

totalindeed <- NROW(indeed)
```

```
cpindeed <- NROW(pindeed)
cmindeed <- NROW(mindeed)
cbindeed <- NROW(bindeed)
ccindeed <- NROW(cindeed)
cvindeed <- NROW(vindeed)

pindeed$weight_skill_type_source <- cpindeed/totalindeed
mindeed$weight_skill_type_source <- cmindeed/totalindeed
bindeed$weight_skill_type_source <- cbindeed/totalindeed
cindeed$weight_skill_type_source <- ccindeed/totalindeed
vindeed$weight_skill_type_source <- cvindeed/totalindeed

pindeed$rating_skill_type_source <- pindeed$rating * pindeed$weight_skill_type_source
mindeed$rating_skill_type_source <- mindeed$rating * mindeed$weight_skill_type_source
bindeed$rating_skill_type_source <- bindeed$rating * bindeed$weight_skill_type_source
cindeed$rating_skill_type_source <- cindeed$rating * cindeed$weight_skill_type_source
vindeed$rating_skill_type_source <- vindeed$rating * vindeed$weight_skill_type_source

#get the skill type name Kaggle
pkaggle <- subset(kaggle, kaggle$skill_type_name == "programming")
mkaggle <- subset(kaggle, kaggle$skill_type_name == "math")
bkaggle <- subset(kaggle, kaggle$skill_type_name == "business")
ckaggle <- subset(kaggle, kaggle$skill_type_name == "communication")
vkaggle <- subset(kaggle, kaggle$skill_type_name == "visualization")

totalkaggle <- NROW(kaggle)

cpkaggle <- NROW(pkaggle)
cmkaggle <- NROW(mkaggle)
cbkaggle <- NROW(bkaggle)
cckaggle <- NROW(ckaggle)
cvkaggle <- NROW(vkaggle)

pkaggle$weight_skill_type_source <- cpkaggle/totalkaggle
mkaggle$weight_skill_type_source <- cmkaggle/totalkaggle
bkaggle$weight_skill_type_source <- cbkaggle/totalkaggle
ckaggle$weight_skill_type_source <- cckaggle/totalkaggle
vkaggle$weight_skill_type_source <- cvkaggle/totalkaggle

pkaggle$rating_skill_type_source <- pkaggle$rating * pkaggle$weight_skill_type_source
mkaggle$rating_skill_type_source <- mkaggle$rating * mkaggle$weight_skill_type_source
bkaggle$rating_skill_type_source <- bkaggle$rating * bkaggle$weight_skill_type_source
ckaggle$rating_skill_type_source <- ckaggle$rating * ckaggle$weight_skill_type_source
vkaggle$rating_skill_type_source <- vkaggle$rating * vkaggle$weight_skill_type_source

dfrankskilltypesource <- rbind(mkaggle,pkaggle,ckaggle,bkaggle,vkaggle,pindeed, mindeed, bindeed, cinde
################################################################################################################

##WEIGHT SKILL TYPE OVERALL
#get all programming information
program <- subset(rs, rs$skill_type_name == "programming"&rs$source_name != "RJMETRICS")

#get all Math information
```

```
math <- subset(rs, rs$skill_type_name == "math"&rs$source_name != "RJMETRICS")

#get all business information
business <- subset(rs, rs$skill_type_name == "business"&rs$source_name != "RJMETRICS")

#get all communication information
comm <- subset(rs, rs$skill_type_name == "communication"&rs$source_name != "RJMETRICS")

#get all visualization information
visual <- subset(rs, rs$skill_type_name == "visualization"&rs$source_name != "RJMETRICS")

totalrs <- NROW(rs)

cpro <- NROW(program)
cmath <- NROW(math)
cbusiness <- NROW(business)
ccomm <- NROW(comm)
cvisual <- NROW(visual)

program$weight_skill_type_name_overall <- cpro/totalrs
math$weight_skill_type_name_overall <- cmath/totalrs
business$weight_skill_type_name_overall <- cbusiness/totalrs
comm$weight_skill_type_name_overall <- ccomm/totalrs
visual$weight_skill_type_name_overall <- cvisual/totalrs

program$rating_skill_type_overall <- program$rating * program$weight_skill_type_name_overall
math$rating_skill_type_overall <- math$rating * math$weight_skill_type_name_overall
business$rating_skill_type_overall <- business$rating * business$weight_skill_type_name_overall
comm$rating_skill_type_overall <- comm$rating * comm$weight_skill_type_name_overall
visual$rating_skill_type_overall <- visual$rating * visual$weight_skill_type_name_overall

dfrankskiltypeoverall <- rbind(program, math, business, comm, visual)
##############################################################################################################

##WEIGHT BY SKILL SET NAME
rs <- subset(rs, rs$source_name != "RJMETRICS")
skillsetnames <- unique(rs$skill_set_name)

cs <- subset(rs, rs$skill_set_name == "Classical Statistics")
isk <- subset(rs, rs$skill_set_name == "Industry-Specific Knowledge")
c <- subset(rs, rs$skill_set_name == "Communication")
ct <- subset(rs, rs$skill_set_name == "Creative Thinking")
b <- subset(rs, rs$skill_set_name == "Business")
pd <- subset(rs, rs$skill_set_name == "Product Development")
m <- subset(rs, rs$skill_set_name == "Math")
bdd <- subset(rs, rs$skill_set_name == "Big and Distributed Data")
bep <- subset(rs, rs$skill_set_name == "Back-End Programming")
ml <- subset(rs, rs$skill_set_name == "Machine Learning")
mp <- subset(rs, rs$skill_set_name == "Mathematical Programming")
sm <- subset(rs, rs$skill_set_name == "Surveys and Marketing")
gp <- subset(rs, rs$skill_set_name == "General Programming")
oop <- subset(rs, rs$skill_set_name == "Object-Oriented Programming")
sp <- subset(rs, rs$skill_set_name == "Statistical Programming")
```

```
st <- subset(rs, rs$skill_set_name == "Structured Data")
v <- subset(rs, rs$skill_set_name == "Visualization")
a <- subset(rs, rs$skill_set_name == "Algorithms")
is <- subset(rs, rs$skill_set_name == "Information Security")
ud <- subset(rs, rs$skill_set_name == "Unstructured Data")
sa <- subset(rs, rs$skill_set_name == "Systems Administration")
md <- subset(rs, rs$skill_set_name == "Mobile Devices")
gm <- subset(rs, rs$skill_set_name == "Graphical Models")
fep <- subset(rs, rs$skill_set_name == "Front-End Programming")
ts <- subset(rs, rs$skill_set_name == "Temporal Statistics")
o <- subset(rs, rs$skill_set_name == "Optimization")
s <- subset(rs, rs$skill_set_name == "Science")
bs <- subset(rs, rs$skill_set_name == "Bayesian/Monte-Carlo Statistics")
sim <- subset(rs, rs$skill_set_name == "Simulation")
ss <- subset(rs, rs$skill_set_name == "Spatial Statistics")
rd <- subset(rs, rs$skill_set_name == "Relational Databases")
dm <- subset(rs, rs$skill_set_name == "Data Manipulation")

total <- NROW(rs)

cs$weight_skill_set_name <- NROW(cs)/total
isk$weight_skill_set_name <- NROW(isk)/total
c$weight_skill_set_name <- NROW(c)/total
ct$weight_skill_set_name <- NROW(ct)/total
b$weight_skill_set_name <- NROW(b)/total
pd$weight_skill_set_name <- NROW(pd)/total
m$weight_skill_set_name <- NROW(m)/total
bdd$weight_skill_set_name <- NROW(bdd)/total
bep$weight_skill_set_name <- NROW(bep)/total
ml$weight_skill_set_name <- NROW(ml)/total
mp$weight_skill_set_name <- NROW(mp)/total
sm$weight_skill_set_name <- NROW(sm)/total
gp$weight_skill_set_name <- NROW(gp)/total
oop$weight_skill_set_name <- NROW(oop)/total
sp$weight_skill_set_name <- NROW(sp)/total
st$weight_skill_set_name <- NROW(st)/total
v$weight_skill_set_name <- NROW(v)/total
#a$weight_skill_set_name <- NROW(a)/total
is$weight_skill_set_name <- NROW(is)/total
ud$weight_skill_set_name <- NROW(ud)/total
sa$weight_skill_set_name <- NROW(sa)/total
md$weight_skill_set_name <- NROW(md)/total
gm$weight_skill_set_name <- NROW(gm)/total
fep$weight_skill_set_name <- NROW(fep)/total
ts$weight_skill_set_name <- NROW(ts)/total
o$weight_skill_set_name <- NROW(o)/total
s$weight_skill_set_name <- NROW(s)/total
bs$weight_skill_set_name <- NROW(bs)/total
sim$weight_skill_set_name <- NROW(sim)/total
rd$weight_skill_set_name <- NROW(rd)/total
dm$weight_skill_set_name <- NROW(dm)/total
ss$weight_skill_set_name <- NROW(ss)/total
```

```
cs$ranking_skill_set_name <- cs$weight_skill_set_name * cs$rating
isk$ranking_skill_set_name <- isk$weight_skill_set_name * isk$rating
c$ranking_skill_set_name <- c$weight_skill_set_name * c$rating
ct$ranking_skill_set_name <- ct$weight_skill_set_name * ct$rating
b$ranking_skill_set_name <- b$weight_skill_set_name * b$rating
pd$ranking_skill_set_name <- pd$weight_skill_set_name * pd$rating
m$ranking_skill_set_name <- m$weight_skill_set_name * m$rating
bdd$ranking_skill_set_name <- bdd$weight_skill_set_name * bdd$rating
bep$ranking_skill_set_name <- bep$weight_skill_set_name * bep$rating
ml$ranking_skill_set_name <- ml$weight_skill_set_name * ml$rating
mp$ranking_skill_set_name <- mp$weight_skill_set_name * mp$rating
sm$ranking_skill_set_name <- sm$weight_skill_set_name * sm$rating
gp$ranking_skill_set_name <- gp$weight_skill_set_name * gp$rating
oop$ranking_skill_set_name <- oop$weight_skill_set_name * oop$rating
sp$ranking_skill_set_name <- sp$weight_skill_set_name * sp$rating
st$ranking_skill_set_name <- st$weight_skill_set_name * st$rating
v$ranking_skill_set_name <- v$weight_skill_set_name * v$rating
#a$ranking_skill_set_name <- a$weight_skill_set_name * a$rating
is$ranking_skill_set_name <- is$weight_skill_set_name * is$rating
ud$ranking_skill_set_name <- ud$weight_skill_set_name * ud$rating
sa$ranking_skill_set_name <- sa$weight_skill_set_name * sa$rating
md$ranking_skill_set_name <- md$weight_skill_set_name * md$rating
gm$ranking_skill_set_name <- gm$weight_skill_set_name * gm$rating
fep$ranking_skill_set_name <- fep$weight_skill_set_name * fep$rating
ts$ranking_skill_set_name <- ts$weight_skill_set_name * ts$rating
o$ranking_skill_set_name <- o$weight_skill_set_name * o$rating
s$ranking_skill_set_name <- s$weight_skill_set_name * s$rating
bs$ranking_skill_set_name <- bs$weight_skill_set_name * bs$rating
sim$ranking_skill_set_name <- sim$weight_skill_set_name * sim$rating
rd$ranking_skill_set_name <- rd$weight_skill_set_name * rd$rating
dm$ranking_skill_set_name <- dm$weight_skill_set_name * dm$rating
ss$ranking_skill_set_name <- ss$weight_skill_set_name * ss$rating

#dfrankskillsetname <- rbind(cs, isk, c, ct, b, pd, m, bdd, bep, ml, mp, sm, gp, sp, oop, st, v, a, is,
dfrankskillsetname <- rbind(cs, isk, c, ct, b, pd, m, bdd, bep, ml, mp, sm, gp, sp, oop, st, v, is, ud,


################################################################################
## KEITH F
## MySQL DB Section
################################################################################

# Limit the 3 dataframes to skill_id, source_id, and the ratings values

df1 <- select(dfrankskiltypeoverall, skill_id, source_id, rating_skill_type_overall)
df2 <- select(dfrankskilltypesource, skill_id, source_id, rating_skill_type_source)
df3 <- select(dfrankskillsetname, skill_id, source_id, ranking_skill_set_name)

# Combine the three dataframes into one table joining on skill_id and source_id
#df1 <- merge(df1, df2, by=c("skill_id"="skill_id", "source_id"="source_id")) # alternate method
#df1 <- merge(df1, df3, by=c("skill_id"="skill_id", "source_id"="source_id"))
update_df <-
  df1 %>%
```

```
  inner_join(df2, by=c("skill_id"="skill_id", "source_id"="source_id")) %>%
  inner_join(df3, by=c("skill_id"="skill_id", "source_id"="source_id") )
update_df

dim(update_df)
str(update_df)

# push the dataframe into a temp table in MySQL
dbSendQuery(skilldb, "DROP TABLE IF EXISTS df_temp;")
dbWriteTable(skilldb, name="df_temp", value=update_df)

# Null out any existing Rating values in the DB
dbSendQuery(skilldb, "
            UPDATE tbl_data
            SET WEIGHTED_RATING_OVERALL = NULL,
            WEIGHTED_RATING_BY_SKILL_TYPE = NULL,
            WEIGHTED_RATING_BY_SKILL_SET = NULL;")

# Update the rating values in tbl_data using the values in the temp table
dbSendQuery(skilldb, "
            UPDATE tbl_data T, df_temp R
            SET T.WEIGHTED_RATING_OVERALL = R.rating_skill_type_overall,
            T.WEIGHTED_RATING_BY_SKILL_TYPE =  R.rating_skill_type_source,
            T.WEIGHTED_RATING_BY_SKILL_SET  = R.ranking_skill_set_name
            WHERE T.SKILL_ID = R.SKILL_ID
            AND T.SOURCE_ID = R.SOURCE_ID;")

dbSendQuery(skilldb, "DROP TABLE IF EXISTS df_temp;")

#close the connection
dbDisconnect(skilldb)
```

f. Presentation-Ready views

Finally, various views were constructed to facilitate the consumption and visualization of the data by the "Presenters" team.
currently, the following views are available in addition to the detail data.

View Names

vw_bottom10_skill_sets_overall
vw_bottom10_skills_overall
vw_skill_type_frequency_percent
vw_top10_skill_sets_overall
vw_top10_skills_by_source
vw_top10_skills_overall
vw_top10_skills_overall_excluding_rjmetrics
vw_top3_skills_by_skill_set
vw_top5_skill_sets_by_skill_type

```
# SECTION 5:  presentation-ready views were created (using scalar values)


################################################################################
```

```
#ROB
# establish the connection to the skill DB on db4free.net
skilldb = dbConnect(MySQL(), user=proj_user, password=proj_pwd, dbname=proj_db, host=proj_host)

# make presentable views
dbSendQuery(skilldb, "DROP VIEW IF EXISTS vw_skill_type_frequency_percent;")
dbSendQuery(skilldb,"
            CREATE VIEW `vw_skill_type_frequency_percent` AS
            SELECT skill_type_name, ROUND(sum(rating_scalar) / (SELECT SUM(rating_scalar) FROM tbl_data
            FROM tbl_data
            GROUP BY skill_type_name ORDER BY sum(rating_scalar) DESC;")

dbSendQuery(skilldb, "DROP VIEW IF EXISTS vw_top10_skills_overall;")
dbSendQuery(skilldb, "
            CREATE VIEW `vw_top10_skills_overall` AS
            SELECT skill_name, SUM(rating_scalar)
            FROM tbl_data
            GROUP BY skill_name
            ORDER BY SUM(rating_scalar) DESC LIMIT 10;")

dbSendQuery(skilldb, "DROP VIEW IF EXISTS vw_top10_skills_overall_excluding_rjmetrics;")
dbSendQuery(skilldb, "
            CREATE VIEW `vw_top10_skills_overall_excluding_rjmetrics` AS
            SELECT skill_name, SUM(rating_scalar)
            FROM tbl_data
            WHERE source_name <> 'RJMETRICS'
            GROUP BY skill_name
            ORDER BY SUM(rating_scalar) DESC LIMIT 10;")

dbSendQuery(skilldb, "DROP VIEW IF EXISTS vw_top10_skill_sets_overall;")
dbSendQuery(skilldb, "
            CREATE VIEW `vw_top10_skill_sets_overall` AS
            SELECT skill_set_name, SUM(rating_scalar)
            FROM tbl_data
            GROUP BY skill_set_name
            ORDER BY SUM(rating_scalar) DESC LIMIT 10;")

dbSendQuery(skilldb, "DROP VIEW IF EXISTS vw_bottom10_skills_overall;")
dbSendQuery(skilldb, "
            CREATE VIEW `vw_bottom10_skills_overall` AS
            SELECT skill_name, SUM(rating_scalar)
            FROM tbl_data
            GROUP BY skill_name
            ORDER BY SUM(rating_scalar) LIMIT 10;")

dbSendQuery(skilldb, "DROP VIEW IF EXISTS vw_bottom10_skill_sets_overall;")
dbSendQuery(skilldb, "
            CREATE VIEW `vw_bottom10_skill_sets_overall` AS
            SELECT skill_set_name, SUM(rating_scalar)
            FROM tbl_data
            GROUP BY skill_set_name
            ORDER BY SUM(rating_scalar) LIMIT 10;")
```

```
# Create each complex view (rankings within categories)
parent  <- c("skill_set_name","source_name", "skill_type_name")
child   <- c("skill_name","skill_name", "skill_set_name")
top_btm <- c("top","top", "top")
label   <- c("skills_by_skill_set","skills_by_source", "skill_sets_by_skill_type")
reps    <- c(3,10,5)

for (i in 1:length(parent)){
  sSQL <-  paste("SELECT ", parent[i], ", ", child[i], ", SUM(rating_scalar) rating_scalar  FROM tbl_da
  df <- dbGetQuery(skilldb, sSQL)
  dbSendQuery(skilldb, "DROP TABLE IF EXISTS df_temp;")
  dbWriteTable(skilldb, name="df_temp", value=df)
  sSQL <- paste("SELECT ", parent[i], ", ", child[i],
                ", rating_scalar, rank FROM (SELECT ", parent[i], ", ", child[i],
                ", rating_scalar, @rank := IF(@current = ", parent[i], ", @rank+1,1) AS rank, @current
                " FROM df_temp ORDER BY ", parent[i], ", rating_scalar DESC) ranked WHERE rank <= ", rep
  df <- dbGetQuery(skilldb, sSQL)
  df <- dbGetQuery(skilldb, sSQL) # sometimes the ranks don't work the first time, just repeat and they
  obj <- paste(top_btm[i],reps[i],"_", label[i], sep = "") # name of table or view
  dbSendQuery(skilldb, paste("DROP TABLE IF EXISTS tbl_", obj, ";", sep = ""))
  dbWriteTable(skilldb, name=paste("tbl_", obj, sep = ""), value=df) #create a temp table for this view
  dbSendQuery(skilldb, paste("DROP VIEW IF EXISTS vw_", obj, ";", sep = "")) # drop the view
  sSQL = paste("CREATE VIEW `vw_", obj, "` AS SELECT * FROM tbl_", obj, " ORDER BY ", parent[i], ", ran
  dbSendQuery(skilldb, sSQL)
}
```

4. MySQL Data Base

a. SQL Schema and script

The SQL script to create the data base can be found at the link below: SQL table creation script

b. Cloud hosting

We selected to host our MySQL Data Base in the cloud so that everyone on the "Presenters"" team could access the data and pulled the most compelling data subset. Because of the nature of this project (student project using publicly available data vs production project using proprietary data), the customary factors that hinder cloud implementation; Security, Access, Back-Up, Technical Support Availability did not apply. We selected: DB4Free.Net has our MySQL DB cloud host. The service is free and the set-up is quite simple. The following is an extract from DB4Free website:
db4free.net provides a testing service for the latest - sometimes even development - version of the MySQL Server. You can easily create an account for free and test your applications, for example to make sure that they still work after a MySQL version update. db4free.net is also a good resource for education and to make yourself familiar with new features that were introduced in new versions.

db4free.net aims to always provide either the latest production release or the latest development release. db4free.net's MySQL server will be updated very soon after a new version is released, usually on the same day or very soon after.

c. Pulling data from Data Base

The enclosed code snippet is to show how data can be pulled from the data base.

```r
# Load appropriate package
library(RMySQL)
```

```
## Loading required package: DBI
```

```r
library(knitr)

# MySQL DB info
proj_user <- "project3"
proj_pwd  <- "CUNYRBridge4"
proj_db   <- "skill"
proj_host <- "db4free.net"

# Establish connection
skilldb = dbConnect(MySQL(), user=proj_user, password=proj_pwd, dbname=proj_db, host=proj_host)

# Show how to query the data
Views_df <- dbGetQuery(skilldb, "SELECT TABLE_NAME as `views name` FROM INFORMATION_SCHEMA.TABLES WHERE
kable(Views_df, caption = "Available Views")
```

Table 1: Available Views

views name
vw_bottom10_skill_sets_overall
vw_bottom10_skills_overall
vw_skill_type_frequency_percent
vw_top10_skill_sets_overall
vw_top10_skills_by_source
vw_top10_skills_overall
vw_top10_skills_overall_excluding_rjmetrics
vw_top3_skills_by_skill_set
vw_top5_skill_sets_by_skill_type

```r
df <- dbGetQuery(skilldb,"SELECT * FROM tbl_data;")
kable(df[1:5, 4:8], caption = "Data Samle")
```

Table 2: Data Samle

source_id	skill_type_name	skill_set_name	skill_name	source_name
1	business	Business	analysis	Kaggle
2	business	Business	analysis	RJMETRICS
3	business	Business	analysis	Google
1	business	Business	business	Kaggle
2	business	Business	business intelligence	RJMETRICS

```r
# End connection
dbDisconnect(skilldb)
```

```
## [1] TRUE
```

5. Visualization and Analysis

a. Visualization and Graphs

```
skilldb <- dbConnect(MySQL(), user=proj_user, password=proj_pwd, dbname=proj_db, host=proj_host)
Views <- dbGetQuery( skilldb,"SELECT TABLE_NAME 'Views'
                              FROM INFORMATION_SCHEMA.TABLES
                              WHERE table_schema = 'skill' and table_type = 'view';")
#kable(Views)
```

We used the Data Science Skills Database to create bar charts, word clouds and other visualization tools that show and summarize the group's findings. The "Transformers" group provided database views to provide results to reduce the need for transformations in R (see above for listing).

i. What are the top skills? The "Transformers" team provided the skill names using two different methods. One involved a `scalar rating` that normalizes skill names across the entire database and the other used a `weighted rating` that weighted the ratings within their respective groups.

We found that the top Data Science Skills are different for the 3 sources. We took the top three skills for each source from the vw_top10_skills_by_source table.

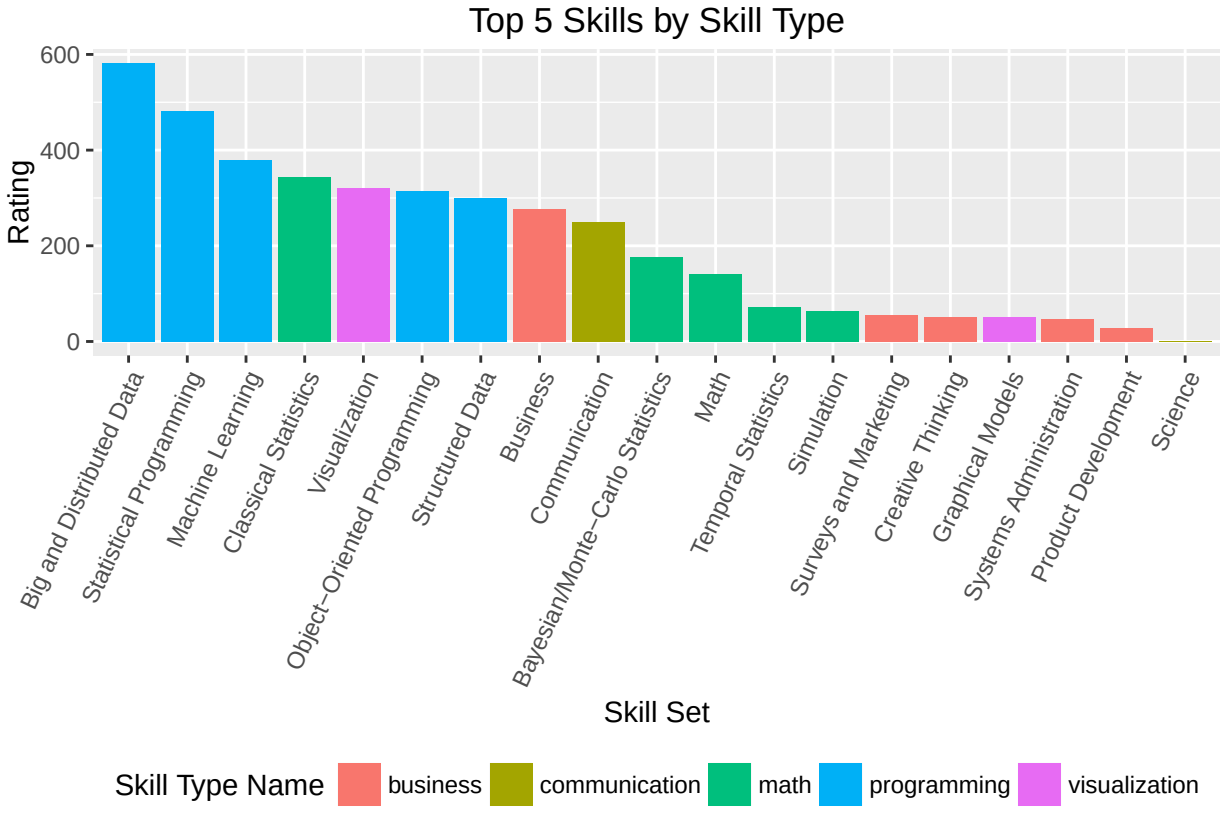
```
Top3_by_source <- dbGetQuery( skilldb, sprintf("SELECT * FROM %s", Views[5,]))

Top3_by_source <- Top3_by_source %>%
                  arrange(source_name, desc(rating_scalar)) %>%
                  group_by(`source_name`) %>%
                  top_n(n = 3, wt = rating_scalar) %>%
                  select(source_name, skill_name, rating_scalar)

kable(Top3_by_source)
```

source_name	skill_name	rating_scalar
Google	big data	100
Google	machine learning	32
Google	Hadoop	25
Indeed	GIS	100
Indeed	statistical graphics	79
Indeed	XML	79
Kaggle	Python	100
Kaggle	analytics	81
Kaggle	machine learning	67
RJMETRICS	analysis	100
RJMETRICS	R	79
RJMETRICS	Python	74

Unsurprisingly, we see that technical skills appear at the top of the lists for each of our data sources but not the exact same type.

Let's look at the top 5 skills sets by skill type by pulling down the view from the database at vw_top5_skill_sets_by_skill_type and checking the results.

```
scalar_skills <- dbGetQuery(skilldb, sprintf("Select * from %s", Views[9,]))

scalar_skills <- scalar_skills %>%
                  select(skill_type_name, rating_scalar, skill_set_name) %>%
                  arrange(skill_set_name)

ggplot(scalar_skills, aes(y = rating_scalar,
                          x = reorder(skill_set_name, -rating_scalar),  fill = skill_type_name)) +
      geom_bar(stat = "identity")  +
      scale_fill_discrete(name="Skill Type Name") +
      xlab("Skill Set") +
      ylab("Rating") +
      theme(legend.position = "bottom",
            axis.text.x = element_text(angle = 65, hjust = 1)) +
      ggtitle("Top 5 Skills by Skill Type")
```



We continue to see technical skills being the most often found skills in each source. `Big and Distributed Data` is the top skill among skill types which seems appropriate for data science.

ii. Horizontal Bar Graph Showing the Top 20 Data Science Skills from the 3 Sources (Google, Indeed and Kaggle) In an effort to examine specific Data Science Skills across the three sources, the top 20 skills for each are examined independently. At the Skill Type level there is more consistency across the sources compared with the more specific Skill Name level. Because of the variability of these specific skills, it is beneficial to show the relative importance within each source and identify common skills. One challenge is the nomenclature across the sources, if the terms are normalized there is potentially more overlapping skill

requirements, but in the current state there appears to be more consistent nomenclature between Google and Kaggle. The most important skill across the three sources is SQL.

order by source_name (asc) and weighted_rating_overall

```
jobdata <- dbGetQuery(skilldb, "select * from tbl_data")
newjobdata <- jobdata[with(jobdata, order(source_name,-weighted_rating_overall)),]
```

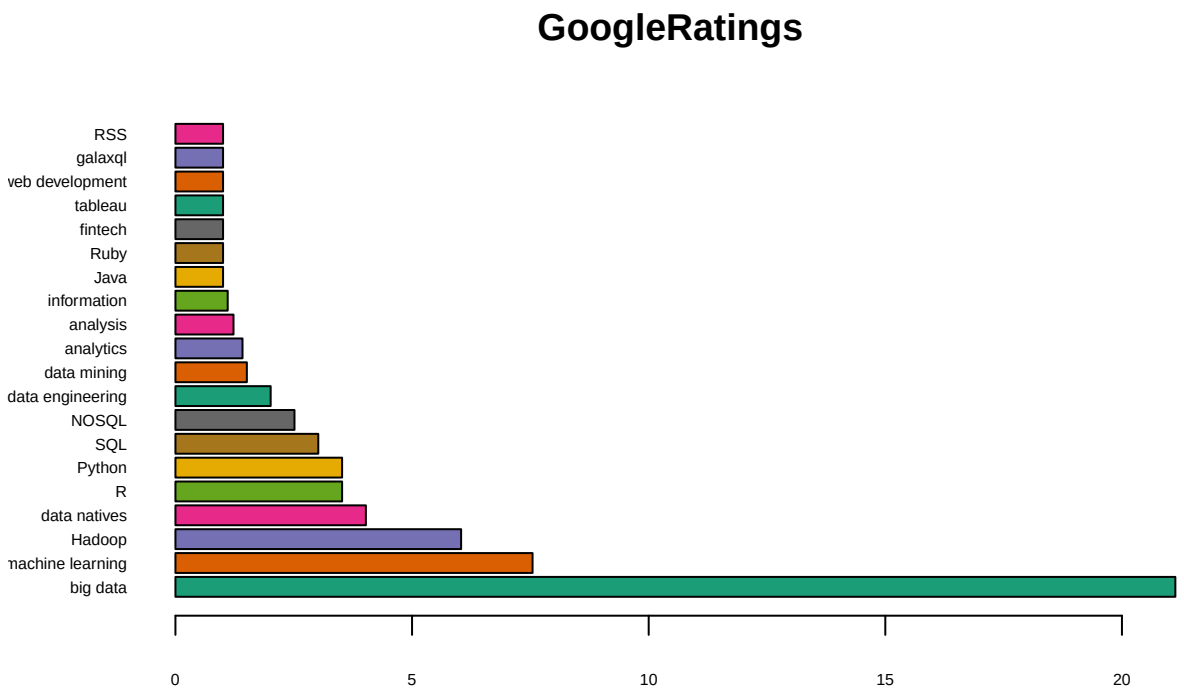
list skill_name for each source_name

```
Google <- subset(newjobdata, source_name == "Google", select=c(source_name, skill_name,weighted_rating_
Google <- Google[c(1:20),]
Indeed <- subset(newjobdata, source_name == "Indeed", select=c(source_name, skill_name, weighted_rating
Indeed <- Indeed[c(1:20),]
Kaggle <- subset(newjobdata, source_name == "Kaggle", select=c(source_name, skill_name, weighted_rating
Kaggle <- Kaggle[c(1:20),]
Combined <- cbind(Google,Indeed,Kaggle)
Combined$source_name <- NULL
Combined$source_name <- NULL
Combined$source_name <- NULL
colnames(Combined)[1] <- "GoogleSkills"
colnames(Combined)[2] <- "GoogleRatings"
colnames(Combined)[3] <- "IndeedSkills"
colnames(Combined)[4] <- "IndeedRatings"
colnames(Combined)[5] <- "KaggleSkills"
colnames(Combined)[6] <- "KaggleRatings"
kable(Combined)
```

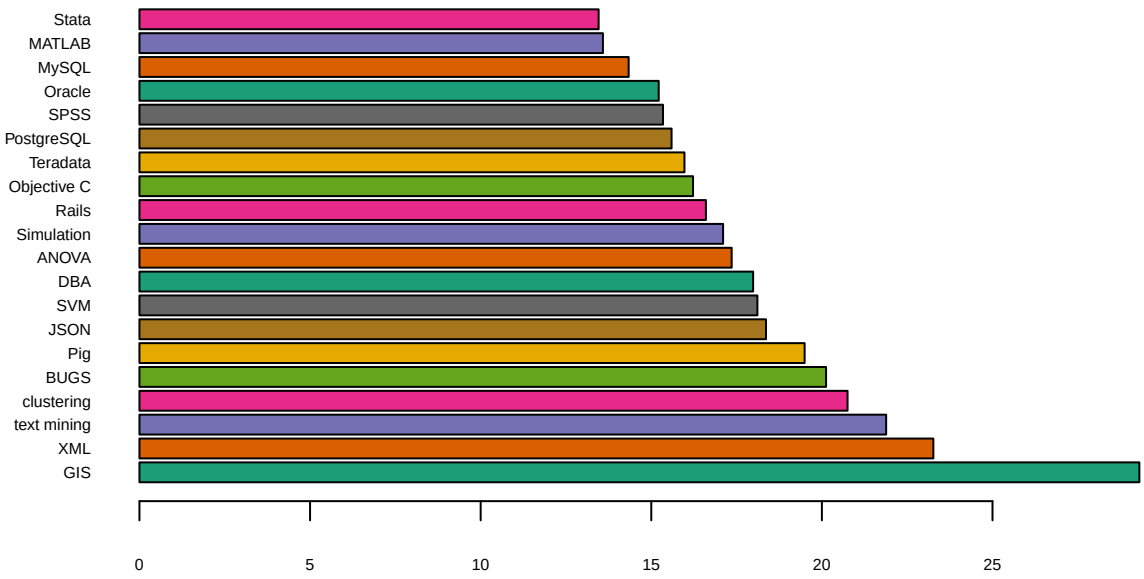
	GoogleSkills	GoogleRatings	IndeedSkills	IndeedRatings	KaggleSkills	KaggleRatings
67	big data	21.12880	GIS	29.3037	Python	14.085900
91	machine learning	7.54601	XML	23.2669	machine learning	9.558280
78	Hadoop	6.03681	text mining	21.8834	programming	7.042940
72	data natives	4.02454	clustering	20.7515	SQL	6.036810
126	R	3.52147	BUGS	20.1227	modeling	4.527610
145	Python	3.52147	Pig	19.4939	big data	3.521470
102	SQL	3.01840	JSON	18.3620	Java	3.018400
113	NOSQL	2.51534	SVM	18.1104	Hadoop	3.018400
69	data engineering	2.01227	DBA	17.9847	analytics	2.963190
71	data mining	1.50920	ANOVA	17.3558	business	1.717790
39	analytics	1.41718	Simulation	17.1043	statistics	1.674850
3	analysis	1.22699	Rails	16.6012	predictive analytics	1.509200
14	information	1.10429	Objective C	16.2239	SAS	1.509200
59	Java	1.00613	Teradata	15.9724	MATLAB	1.509200
63	Ruby	1.00613	PostgreSQL	15.5951	team	0.773006
74	fintech	1.00613	SPSS	15.3436	communication	0.607362
86	tableau	1.00613	Oracle	15.2178	R	0.503067
87	web development	1.00613	MySQL	14.3374	interpersonal	0.496933
97	galaxql	1.00613	MATLAB	13.5828	management	0.496933
99	RSS	1.00613	Stata	13.4571	research	0.490798

plot results

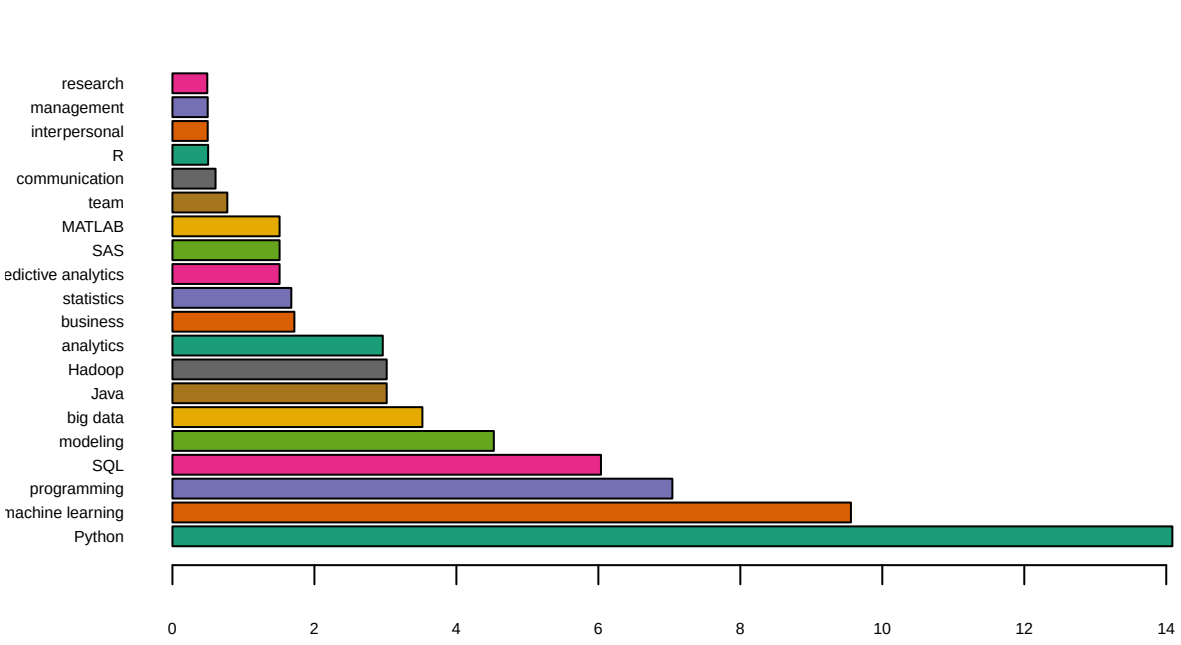
```
darkcols <- brewer.pal(8,"Dark2")
names <- Combined$GoogleSkills
barplot(Combined$GoogleRatings,main="GoogleRatings", horiz=TRUE, names.arg=names, las=1, col=darkcols, c
```




```
names <- Combined$IndeedSkills
barplot(Combined$IndeedRatings,main="IndeedRatings", horiz=TRUE, names.arg=names, las=1, col=darkcols, c
```




```
names <- Combined$KaggleSkills
barplot(Combined$KaggleRatings,main="KaggleRatings", horiz=TRUE, names.arg=names, las=1, col=darkcols, c
```



iii. Bubble Graph Showing Weighted Rank of Skills by Skill Type and Source (Google, Indeed and Kaggle) We have a wealth of information within our Data Science Skills database, we need to be able to quickly understand where the majority of highly ranked skills are located. The below graph provides a high level overview of the skills and how they relate to each other across the skill types by each source.

```
data_science <- jobdata

data_science <- jobdata

data_science$skill_name <- factor(data_science$skill_name)
data_science$skill_name <- factor(data_science$skill_name, levels = rev(levels(data_science$skill_name))

ggplot(data_science[(data_science$source_id != 2),],
       aes( source_name, skill_name, label = skill_name,
           size = rating_scalar, fill = skill_type_name)) +
        geom_point(pch = 21) +
        scale_fill_manual(values =  brewer.pal(9, "Set1")) +
        scale_size_continuous(range =c(1,10)) +
        facet_grid(~skill_type_name) +
        theme_light() +
        xlab("Source") +
        ylab("Skill") +
        theme(legend.position = "none" ,
              axis.text.y = element_text(size=5),
              axis.text.x = element_text(angle = 45, hjust = 1)) +
```

```
        ggtitle("Weighted Rank of Skill by Skill Type and Source")
```

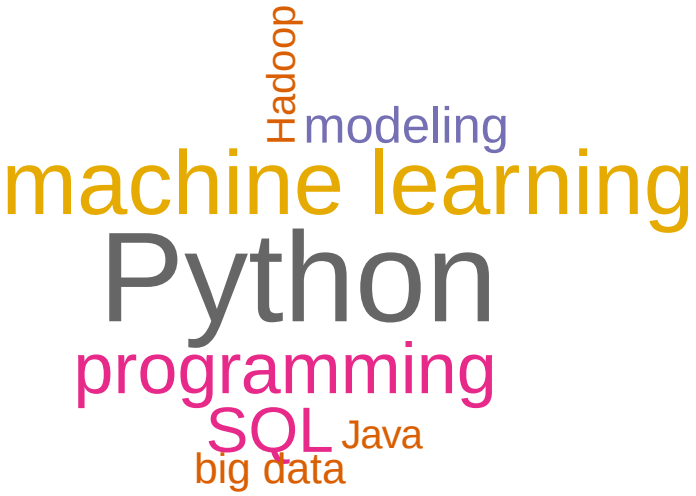
```
stab <- data %>%
        group_by(source_name,skill_type_name) %>%
        summarise(ave_wgt =mean(weighted_rating_overall))

ggplot(stab, aes(x =source_name, y=round(ave_wgt,2), fill = skill_type_name)) +
       geom_bar(stat="identity",position="dodge") +
       xlab("Source") +
       ylab("Average Weighted Rating Overall") +
       ggtitle("Average Weighted Overall Rating by Source and Skill Type")
```

```
require(wordcloud)

google <- subset(jobdata, source_name=="Google")
indeed <- subset(jobdata, source_name=="Indeed")
kaggle <- subset(jobdata, source_name=="Kaggle")

wordcloud(google$skill_name, google$weighted_rating_overall, random.order = FALSE, colors=brewer.pal(8,
```

iv. Graph Showing Average Weighted Overall Rating by Source and Skill Type

```
wordcloud(indeed$skill_name, indeed$weighted_rating_overall, random.order = FALSE, colors=brewer.pal(8,
```


```
wordcloud(kaggle$skill_name, kaggle$weighted_rating_overall, random.order = FALSE, colors=brewer.pal(8,
```



b. Conclusion

The data from all 3 sources show that programming is the primary and predominant set of skill needed in data science. The average weighted overall rating for programming, which included such skills such as GIS, Machine Learning and Python, exceeded the average weighted overall rating for all the other skill sets combined.

For all 3 sources, math skills came in second, followed by business skills. For Google and Indeed, visualization skills were fourth followed by communication skills last. For our Kaggle source, communication skills came in fourth followed by visualization skills last.

There may be several reasons for these results. First, our group's classification of data science skill set types (programming, math, business, communication and visualization) may need revision (is GIS programing or math). Should we have had another skill type such as Analytics? There may be also be implied overlaps between programming, math and visualization skills. It seems that when employers post skills on the job boards or when bloggers write articles on data science, they assume that when a person has the skill to program in Python, Hadoop, Machine Language or R, the math and visualization skills are already part of it. Employers and writers assume that if a person is proficient in a data science programming language, he also has the math and visualization skills that come with knowing the programming language. Second, programming is the predominant skill needed in the early stages of the data science process such as data collection, data cleaning and building algorithms and model. It is only when we get to the visualization and data analysis stage where math, communication and visualization skills become as significant as programming skills. Third, domain knowledge and expertise (business skills), although as important as technical and match skills (see data science Venn diagram), are not emphasized on job sites. Most of the jobs for data scientist are entry or mid-level jobs that do not require domain expertise. These qualifications are assumed to come later as the employee gains more experience with the company and learns its business processes.

So what is the most valued skill type in data science? Not surprisingly, technical skills such as programming and math skills are the most valued. You need to be technically savvy to have a career in data science.

6. Lessons Learned

This was a very interesting project for multiple reasons and it fair to say that we learned quite a bit as we strove to accomplish our goals in the time allotted. We encountered some difficulties along the way as expected; some we could resolve and some we could not and had to make quick decision and move on. Everyone was eager participate and ready to pinch in.

The following are some key points that we learned along the way:

1. Communication:
   For most of us, this project was the first time that we worked collaboratively with each other and even though some of us reside in the NY area, all communications were done remotely (no In-person communication). The technology stack that we selected for this project yielded mixed results.

- Webex for online meeting: Good
- Email for basic exchange of ideas/files: Not Good, became very unruly towards end of project
- Github / Rpubs for handoff of project pieces: Good, probably could make better use of Github capability regarding forked repositories.
- Phone: OK, limited use and not always had numbers. We could have done better with same technology
- FTP: for file exchange: Not Good, we underutilize this option, should have done better job

In future projects we would definitely look to improve communication probably via using a tool like "slack" and agree on a protocol to exchange material and files much earlier in the project.

2. Knowledge Transfer between teams The structure of the project was vertical and each stage could have benefitted from a formalized meeting plan where the information was handed off to each downstream group. For instance, the "Transformers" team had created Views in the skills database that the "Presenters" may have never used had there not been a quick Webex meeting to discuss the "Transformers" section of the presentation.

The more structure agreed upon in earlier stages, the better. The original scraped files did not all conform to the same format and required cleaning. This processing step could have been avoided with a stricter template.

3. Time-Management: The team was fairly successful in managing time. Although some groups "Transformers" got squeazed and had to work hard to get results out. The documenting proved to be a problem as well.

4. Limited scope and more focus approach
   The team was quite ambitious in what we tried to accomplish. We were quite successful but a more focus approach with a more time spent in planning would have been beneficial. For example, having a team using a SQL server can be challenging. The handoff of CSV files between teams may be an easier approach. In future project, we should keep our eyes on the goals and requirements.

5. Documenation: It is not unusal for documentation to be an afterthought in a project and this one was no different. In future project, we would probably appoint a record-keeper much earlier and have this person in charge of the overall documentation of the project.

6. Scrapping Process: This project demonstrated that scraping relevant information from the web can be a tedious and difficult process. Scraping can collect a lot of meaningless data. More fine-tuning or downstream processing is sometimes required to get a useful output. We understood this going in and attempting to adopt a "Agile" methodology that would allow for several iterations to refine the process. We were not quite successful as each group got buried in their own process and we ran out of time. As mentioned earlier, it would have been beneficial to have more team to team communications. For example, the mapping of sills to skill set could have been refined iteratively.

Final note, we learned as much about the process and methodology as we did about scraping text from the web. Our team demonstrated inquisitiveness, perseverance, team work and collaboration, high level of technical skills and dived in whole heartedly in visualization of the data. We demonstrated a "can do" attitude that serves us well as we pushed toward the finish line.
We are on our way of becoming Data Scientists.