Data 607 - Project 3

Cameron Smith

2020-10-18

```
library(tidyverse)
library(RPostgreSQL)
library(DBI)
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.0.3
```

```
## Warning: package 'NLP' was built under R version 4.0.3
```

```
library(SnowballC)
```

```
## Warning: package 'SnowballC' was built under R version 4.0.3
```

```
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 4.0.3
```

```
library(RCurl)
library(RColorBrewer)
```

Introduction and Approach

This project is a group project with the following members:

- Cameron Smith
- Brett D
- Darius Z

The approach used is focused on 3 main steps: (1) loading the data; (2) tidying the data; and (3) analyzing the data. A Google Cloud Postgres database was created, in which the raw data was loaded into tables which are accessed in the code below and then put into data frames. From the data is prepped with key calculations performed by analysis, followed by the actual analysis augmented with visualizations to illustrate the key findings.

Formal citations for sources used:

JobsPikr. (2019; December). 10000 Data Scientist Job Postings from the USA, Version 1. Retrieved 15 October 2020 from https://www.kaggle.com/jobspikr/data-scientist-job-postings-from-the-usa.

Hale, Jeff (2018; October). Data Science Career Terms. Retrieved 15 October 2020 from https://docs.google.com/spreadsheets/d/1lac1H2IgCDCs9LLTQL6yb6MUPN1u4C5fJv_6YjipIaM/edit#gid=1072460513

STHDA. Word cloud generator in R : One killer function to do everything you need. Retrieved 17 October 2020 from http://www.sthda.com/english/wiki/word-cloud-generator-in-r-one-killer-function-to-do-everything-you-need#r-code-of-rquery.wordcloud-function.

Load the data

A Google Cloud Postgres database was created for this project to maximize reproducibility. The below code loads the data from the normalized database and converts it into data frames that can be used for further analysis.

This required (1) setting up a Postgres database; (2) creating a storage bucket; (3) uploading the raw data to the storage bucket; (4) migrating that raw data into the new database, which consists of two tables in the first normal form 1NF; and (5) opening up the firewall to allow public access. Per the rules for 1NF, each table cell should contain a single value and each record needs to be unique. Both of these conditions are satisfied and thus the data can be considered normalized.

The RPostgres package was used to acess and query the database.

```
# Configure database connection
db <- "postgres"
host_db <- "35.243.187.235"
db_port <- "5432"

# Using public account created with read only access
db_user <- "project3public"
db_password <- "cunymsds2020"

con <- dbConnect(RPostgres::Postgres(), dbname = db, host=host_db, port=db_port, user=db_user, password=

# Verify the connection to the database is working
dbListTables(con)
```

```
## [1] "jobs"   "skills"
```

```
# Get job posting data
jobdata_raw <-
  dbGetQuery(con, "SELECT job_title, job_description, category, company, city, state, job_board, post_da

# Get skills data
skills_df <- dbGetQuery(con, "SELECT skill_name FROM skills")

# Disconnect from the database
dbDisconnect(con)
```

Tidy and Transform

With the data now loaded into a data frame, it needs to be prepared for analysis.

Main rules of 'tidyness':

- Each variable must have its own column

- Each observation must have its own row
- Each value must have its own cell

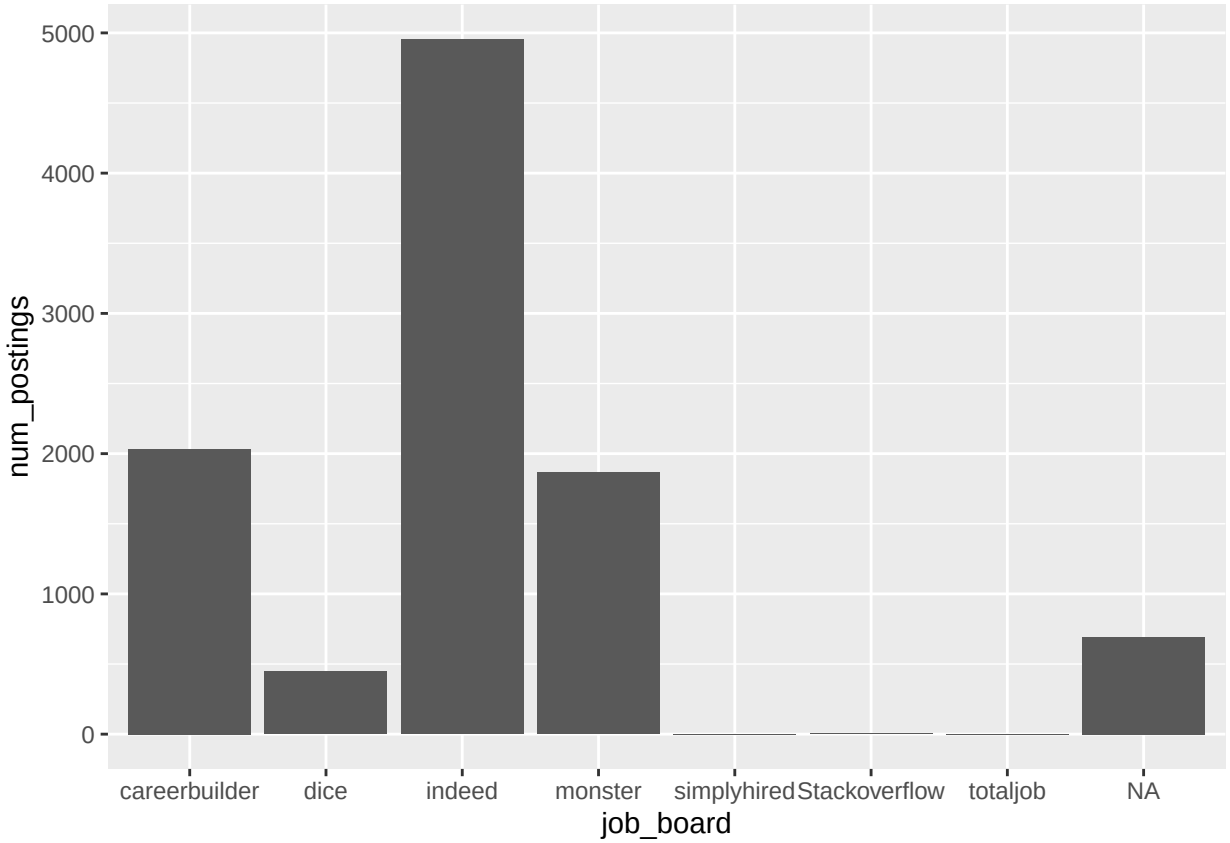
Following these rules we end up with a very wide data set in this instance.

```
# Start with a bit of exploratory data analysis
glimpse(jobdata_raw)
```

```
## Rows: 10,002
## Columns: 8
## $ job_title       <chr> "Enterprise Data Scientist I", "Data Scientist", "D...
## $ job_description <chr> "Read what people are saying about working here. \n...
## $ category        <chr> "Accounting/Finance", NA, NA, "Accounting/Finance",...
## $ company         <chr> "Farmers Insurance Group", "Luxoft USA Inc", "Cinci...
## $ city            <chr> "Woodland hills", "Middletown", "New york", "New yo...
## $ state           <chr> "California", "New jersey", "New york", "New york",...
## $ job_board       <chr> "indeed", "dice", "dice", "indeed", "monster", "ind...
## $ post_date       <date> 2019-02-06, 2019-02-05, 2019-02-05, 2019-02-06, 20...
```

```
# Summary of job postings by site
jobdata_raw %>%
  select(job_board) %>%
  group_by(job_board) %>%
  summarise(num_postings = n()) %>%
  ggplot(aes(x = job_board, y = num_postings)) +
    geom_bar(stat = 'identity')
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Prepare the data frame to capture the data needed. For this step a list of data science skills was used to essentially serve as a dictionary. A for loop goes through each element of the dictionary and creates a corresponding column based on the skill name with blank values.

```
# Copy the raw data into a new data frame to preserve the original
jobdata_tidy <- jobdata_raw

# Create wide data frame with new columns and blank based on each skill name
for (i in 1:nrow(skills_df)){
  tempvar = skills_df$skill_name[i]
  jobdata_tidy[[tempvar]] <- ""
}
```

Now that the data frame is in the intended format, with the key columns created, we will search through each job listing to identify whether each skill is listed, then add it to a count in the columns created. This is done through a loop that iterates through a list of each skill and then sets the corresponding column name to true or false depending on whether the string is found.

Note: This operation can take up to 1 minute to complete

```
for(s in 1:nrow(skills_df)) {
  skill = skills_df[s, 1]
  jobdata_tidy[skill] = grepl(tolower(paste("", skill, "", sep=" ")), tolower(jobdata_tidy$job_descripti
}
```

With the data calculated and populated, a new data frame will be created in a longer format for easier analysis.

```
# Create long data frame with key data for easier analysis
skillsdata_long <- jobdata_tidy %>%
  select(9:54) %>%
  gather ("skill_name", "exists", 1:46) %>%
  filter(exists == TRUE)
```

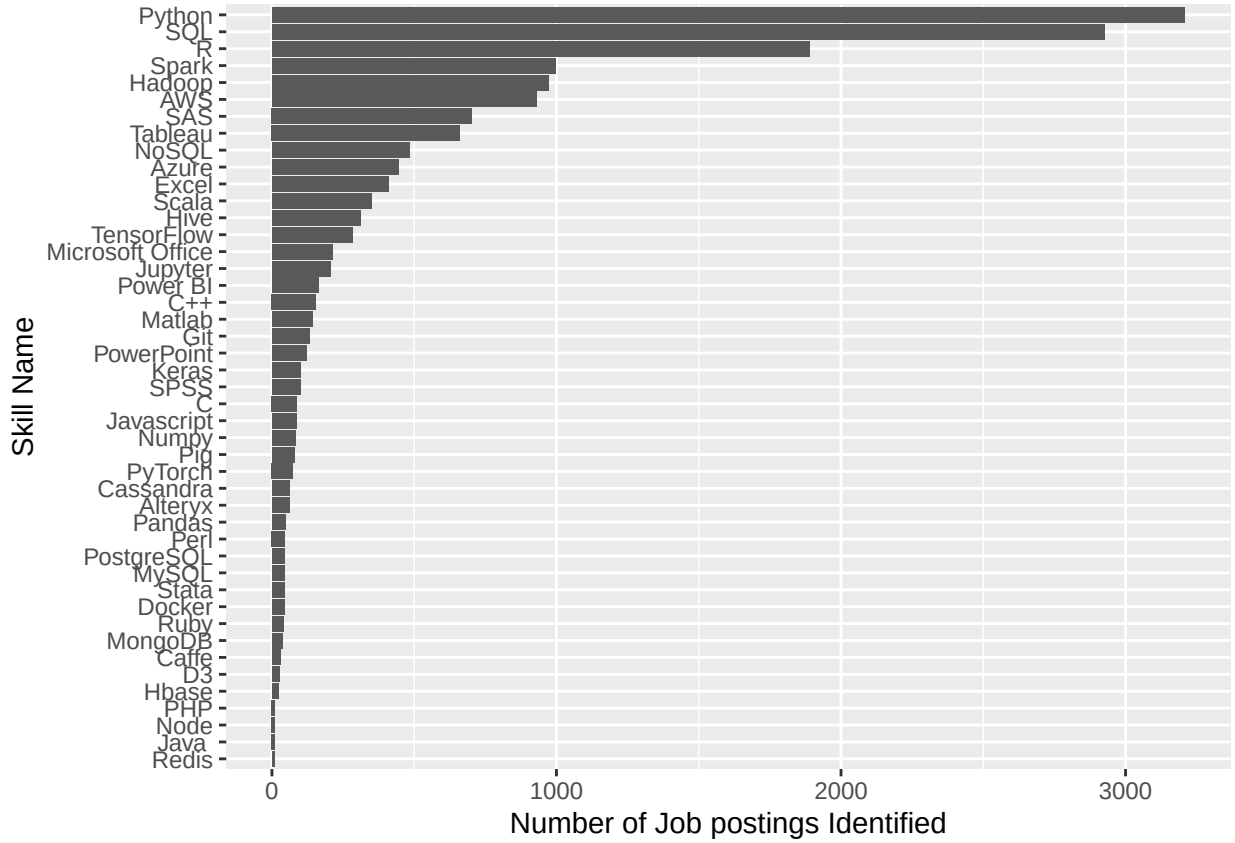
Analysis

The most sought after skills for a data scientist according are Python, SQL, and R (in that order), with everything else at least 50 percent behind. I think that it is interesting that Python has become so pivotal in this field, surpassing even R.

```
skillssummary <- skillsdata_long %>%
  group_by(skill_name) %>%
  summarise(num_postings = n())

# Visualize in descending order the popularity of all skills identified

skillssummary %>%
  ggplot(aes(x = reorder(skill_name, num_postings), y = num_postings)) +
  geom_bar(stat = 'identity') +
  xlab("Skill Name") +
  ylab("Number of Job postings Identified") +
  coord_flip()
```



A word cloud was also created based on the text of the job descriptions to provide a different view of the

data from which perhaps new insights can be gleaned. An advantage of this approach is that it does not rely on pre-defined terms as did the previous analysis.

To facilitate simple creation of the wordcloud the rquery.wordcloud function was used, which is not part of an existing R package so the code has been pasted into the below code block.

The original source for the code is:

http://www.sthda.com/english/wiki/word-cloud-generator-in-r-one-killer-function-to-do-everything-you-need#r-code-of-rquery.wordcloud-function

```
rquery.wordcloud <- function(x, type=c("text", "url", "file"),
                             lang="english", excludeWords=NULL,
                             textStemming=FALSE,  colorPalette="Dark2",
                             min.freq=3, max.words=200)
{
  library("tm")
  library("SnowballC")
  library("wordcloud")
  library("RColorBrewer")

  if(type[1]=="file") text <- readLines(x)
  else if(type[1]=="url") text <- html_to_text(x)
  else if(type[1]=="text") text <- x

  # Load the text as a corpus
  docs <- Corpus(VectorSource(text))
  # Convert the text to lower case
  docs <- tm_map(docs, content_transformer(tolower))
  # Remove numbers
  docs <- tm_map(docs, removeNumbers)
  # Remove stopwords for the language
  docs <- tm_map(docs, removeWords, stopwords(lang))
  # Remove punctuations
  docs <- tm_map(docs, removePunctuation)
  # Eliminate extra white spaces
  docs <- tm_map(docs, stripWhitespace)
  # Remove your own stopwords
  if(!is.null(excludeWords))
    docs <- tm_map(docs, removeWords, excludeWords)
  # Text stemming
  if(textStemming) docs <- tm_map(docs, stemDocument)
  # Create term-document matrix
  tdm <- TermDocumentMatrix(docs)
  m <- as.matrix(tdm)
  v <- sort(rowSums(m),decreasing=TRUE)
  d <- data.frame(word = names(v),freq=v)
  # check the color palette name
  if(!colorPalette %in% rownames(brewer.pal.info)) colors = colorPalette
  else colors = brewer.pal(8, colorPalette)
  # Plot the word cloud
  set.seed(1234)
  wordcloud(d$word,d$freq, min.freq=min.freq, max.words=max.words,
            random.order=FALSE, rot.per=0.35,
            use.r.layout=FALSE, colors=colors)
```

```
  invisible(list(tdm=tdm, freqTable = d))
}

#++++++++++++++++++++++
# Helper function
#++++++++++++++++++++++
# Download and parse webpage
html_to_text<-function(url){
  library(RCurl)
  library(XML)
  # download html
  html.doc <- getURL(url)
  #convert to plain text
  doc = htmlParse(html.doc, asText=TRUE)
 # "//text()" returns all text outside of HTML tags.
 # We also don't want text such as style and script codes
  text <- xpathSApply(doc, "//text()[not(ancestor::script)][not(ancestor::style)][not(ancestor::noscrip
  # Format text vector into one character string
  return(paste(text, collapse = " "))
}
```

With the function now defined we can create the wordcloud which gleans new insights from our data. Previous analysis had been focused on skills from a traditional technical, software-focused perspective. The wordcloud indicates that there are also more general concepts and terms found within the data ,with the words "data", "analytics", "business", "learning", "analysis", and "machine" some of the most common, which can reasonably be distilled into: analysis, business, and machine learning.

```
write(jobdata_tidy[1:1250,]$job_description, "temp_r_wordcloud.txt")
filePath <- "temp_r_wordcloud.txt"
res <- rquery.wordcloud(filePath, type ="file", lang = "english", max.words = 50)
```

Conclusion

In conclusion, based on a consolidated list of 10,000 data science job postings from a variety of websites, the top 10 data science skills from a software perspective are as follows:

```
top_n(arrange(skillssummary, desc(num_postings)), 10)
```

```
## # A tibble: 10 x 2
##    skill_name num_postings
##    <chr>             <int>
##  1 Python             3208
##  2 SQL                2925
##  3 R                  1889
##  4 Spark               998
##  5 Hadoop              971
##  6 AWS                 931
##  7 SAS                 703
##  8 Tableau             661
##  9 NoSQL               483
## 10 Azure               445
```

In addition, it is also clear that analysis, business, and machine learning skills are also highly sought after.