Data Analyst jobs in the Northeast U.S.

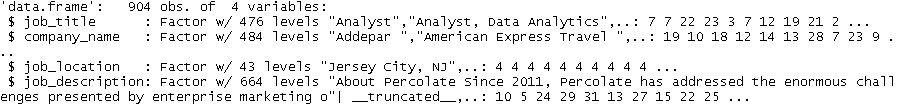
Tanner Sax

Introduction

As companies with technology and data requirements continue to grow, there will be more demand for employees with certain skillsets to navigate this data. There are many exciting opportunities in this field for recent graduates with analytical and problem-solving skills. Harvard Business Review coined the Data Science profession as “The sexiest job of the 21st century.”[[1]](#footnote-1) This is good news for those with an interest in technology, however it is important to take into consideration the experience and skillset required. An article released by Northeastern states that “Data scientists—who have typically earned a graduate degree, boast an advanced skill set, and are often more experienced—are considered more senior than data analysts.”[[2]](#footnote-2) For a recent graduate, it is more likely to land a role as a data analyst before progressing into a data scientist role.

This analysis discovers the most preferred programming language skills and degree field for job seekers in the Northeast searching for a data analyst role. According to Business Insider, among the eleven most high-tech cities in the United States are New York, Boston, Washington, D.C., and Philadelphia.[[3]](#footnote-3) Analytics India Magazine’s list of top programming languages was used to select eight popular languages.[[4]](#footnote-4) The languages analyzed in this report include python, SQL, R, java, matlab, scala, SAS, and Julia.

Glassdoor.com is one of the largest job listing websites in the world, and new jobs are posted daily. Job seekers can view information such as the job title, company, reviews, salary estimates, and job descriptions that contain qualifications and responsibilities. A web scrape was performed on Glassdoor.com to collect information for Data Analyst job postings from the four cities mentioned above. Table 1 contains a summary of the data that was collected for New York City.

**Table 1**

Data

The web scrape collects data from the first thirty pages for each city from Glassdoor.com. This is because only the first thirty pages are viewable on the website. This lessens the sample size; however, 3,544 postings were still able to be collected.

A search for ‘Data Analyst’ was performed for each of four major tech cities in the Northeast – including New York, Boston, Washington, D.C., and Philadelphia. The data consists of the job title, company name, location, and a full job description from the Glassdoor job postings. The web scrape collected the data for each city separately and stored it in a data frame. There were several job description links that were unable to be scraped due to ads. These values were set to return NA in the data frames. The rows containing NA in the job description column were then removed using the complete.case() function, as shown in Figure 1.

*New\_York <- New\_York[complete.cases(New\_York), ]*  **Figure 1**

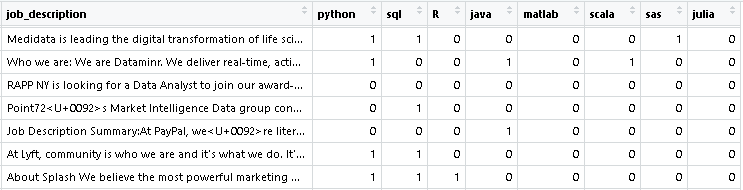
Since there were only 11 NA’s total in 3,555 results, removing them did not have a large impact on the sample.

An important aspect of the data is the job description column. This is because the analysis requires keyword extraction to identify the most common programming language skills and degree fields. Separate columns were created that identify keyword matches through binary variables. This was done through the mutate function for each of the keywords, as shown in Figure 2.

*New\_York <- New\_York %>%* ***Figure 2***

*mutate(python = ifelse(grepl("python", job\_description, ignore.case = TRUE), 1, 0))*

The grepl function searches for a match and returns a logical vector. If it returns TRUE, then according to the ifelse statement, a ‘1’ will be returned. Otherwise, a ‘0’ will be returned. Table 2 shows an example of the binary outputs for each language.

**Table 2**

The means of these columns represent the percentage of job postings that mention the keyword. Then, two separate tidy data frames were created for the language skills and degree fields with these values. To have a tidy data frame, vectors were created for the cities and languages/fields that placed them in the correct order with the means. The vectors were then combined into a tibble data frame, as shown in Figure 3.

*language\_df <- tibble(cities1, language, Percent\_of\_jobs\_language)* ***Figure 3***

*degree\_field\_df <- tibble(cities2, degree\_field, Percent\_of\_jobs\_degree)*

The first data frame includes the city, programming language, and percent of job postings that mention the programming language. The second data frame contains the city, degree field, and percent of job postings that mention the degree field.

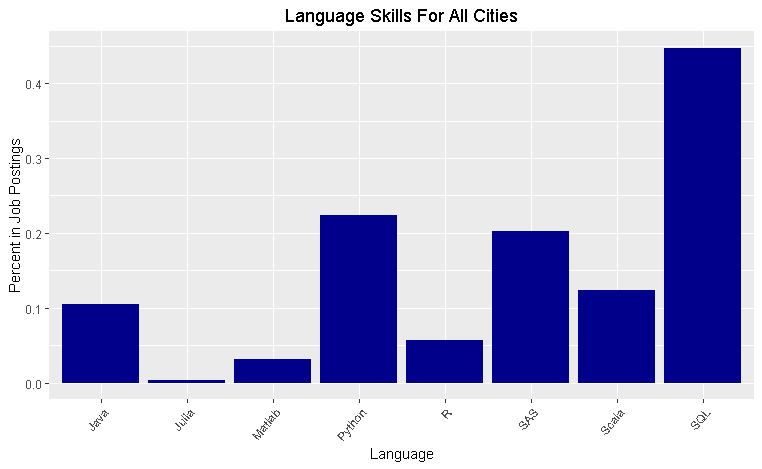
Next, a data frame was created that includes all the data from all four cities. Similar keyword extraction methods were utilized to analyze the interactions of the top three programming languages and the top four degree fields. Each of the top programming languages were paired with one of the top four degree fields. The job description had to contain both the programming language and the degree field to return a ‘1’. Then the means of these combinations were put into a vector. Each combination name was put into a vector as well. Then, a data frame was created of the combination names and the means, as shown in Figure 4.

top\_df <- tibble(top\_names, Percent\_of\_jobs\_top) **Figure 4**

Analysis

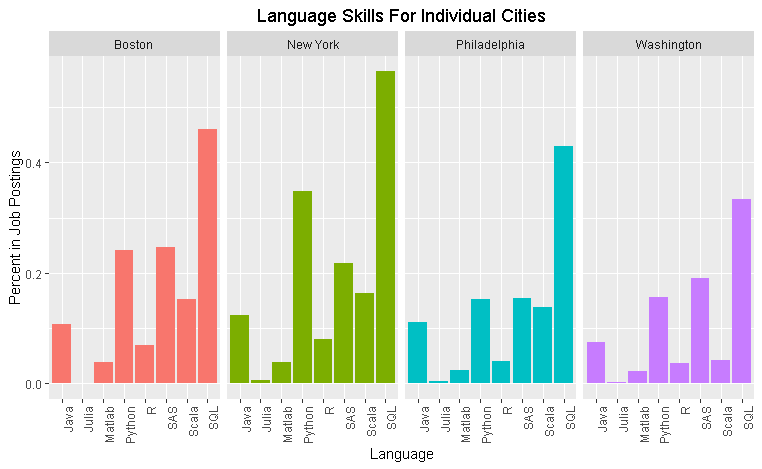
After the data was cleaned and the new appropriate data frames created, several graphs were created with the ggplot2 package. According to the graph below, SQL is the most important programming language for data analysts in all four cities, appearing in 44.9 percent of job postings. This is to be expected for a data analyst job. Julia, a very new programming language, is mentioned the least amount of times at 0.28 percent, as expected. Table 3 shows the percent of language skills in the job postings for all four cities.

**Table 3**



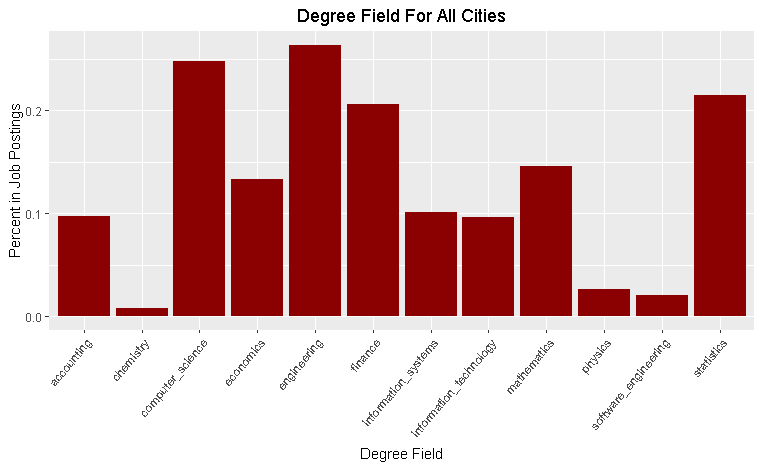
In New York, Boston, Washington, D.C., and Philadelphia, SQL is mentioned in 56.22, 46.07, 33.18, and 42.91 percent of job postings, respectively. In New York, Boston, Washington, D.C., and Philadelphia, Julia is mentioned in .54, 0, .23, and .34 percent of job postings, respectively. Python appears to be more important in New York than in other cities, appearing in 34.66 percent of job postings there. Scala appears to be less important in Washington, D.C. than in other cities, appearing in only 4.16 percent of job postings. Table 4 shows the percent of language skills in the job postings for each city individually.

**Table 4**



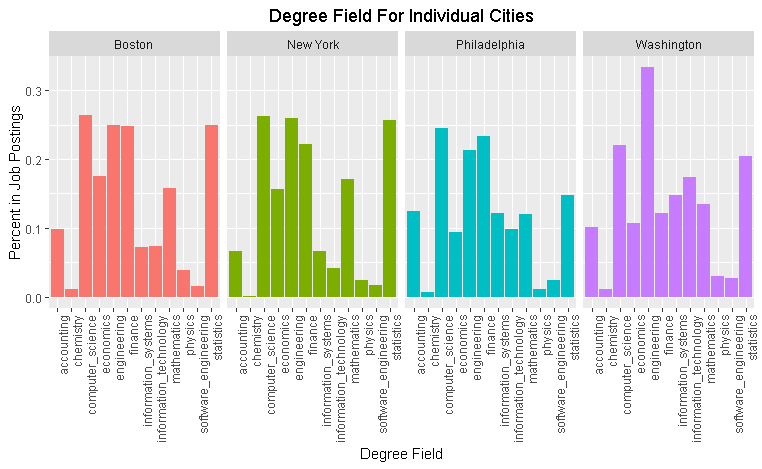
Computer science, engineering, finance, and statistics appear to be among the most popular degree fields. Engineering appears in 26.4 percent of job postings. Table 5 shows the percentage of degree fields in the job postings for all four cities.

**Table 5**



Information technology degrees appear more in the job postings in Washington, D.C. than in other cities. Table 6 shows the percentage of the degree fields in the job postings for each city individually.

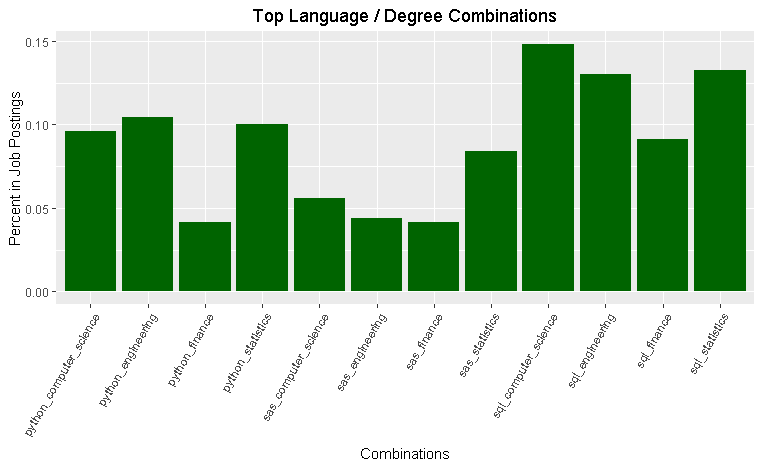
**Table 6**



For data analyst jobs in the Northeast the most preferred programming languages are SQL, Python, and SAS, and the preferred degree fields are computer science, engineering, finance, and statistics. It is more important to know Python for jobs in New York City than for jobs in other cities. It is less important to know Scala for jobs in Washington, D.C. than in other cities.

A further analysis compares each of these top majors with the top programming languages. This suggests which combination of programming language and degree field is the most advantageous for obtaining a data analyst job. The combination of SQL and computer science is the most common. In 14.84 percent of job postings, both the keywords ‘SQL’ and ‘computer science’ appeared. Table 7 shows the percentage of language skill and degree field combinations for all four cities.

**Table 7**



This is an interesting insight for job seekers. The most preferred candidates are computer science majors with knowledge of SQL. This is great; however, many companies don’t rely on only one programming language. Since SQL is used mostly for navigating databases, it is very common for companies to also use other programming languages for different purposes. Wouldn’t it be nice to know which programming languages are usually paired together in the job descriptions? This will inform us of which programming languages typically are sought after simultaneously. Understanding the companies set of programming languages will give an advantage to the applicant. This is possible through a clustering method.

The specific type used will be hierarchical clustering. The process begins by identifying each observation as a separate cluster. The distances between all data points must be computed. The dist() function can be used to compute these distances using the Euclidian method, as shown in Figure 5.

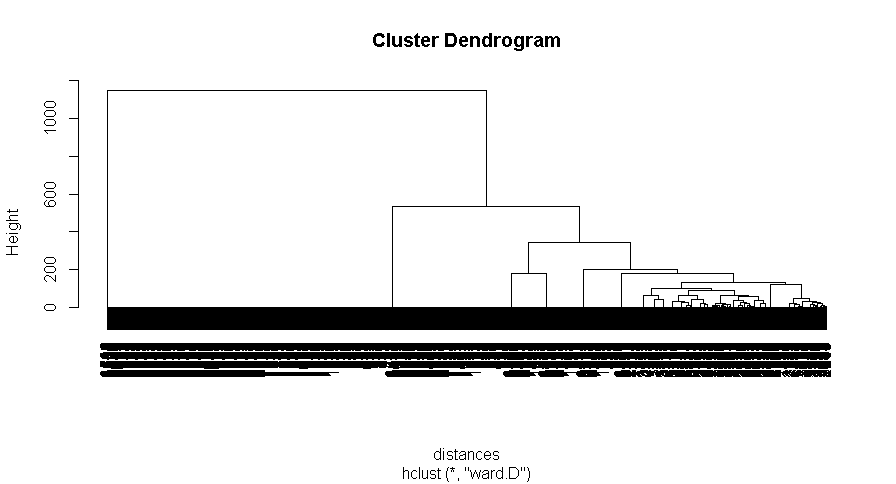
distances = dist(all\_cities[3:10], method = "euclidian") **Figure 5**

Next, the points need to be clustered. This can be done using the hclust() function, for hierarchical clustering, as shown in Figure 6.

clusterLanguages = hclust(distances, method = "ward") **Figure 6**

The ward method measures the distance between clusters using centroid distance and reduces the variance in each of the clusters. The two clusters that are closest are then merged together. This is repeated until there is only one cluster remaining. Since the occurrence of programming languages in this analysis are represented through binary variables, the clustering function will identify which job descriptions have the smallest difference of programming language occurrences. A dendrogram illustrates the relationships between clusters. We can plot the dendrogram of the clustering algorithm, as shown in table 8.

**Table 8**

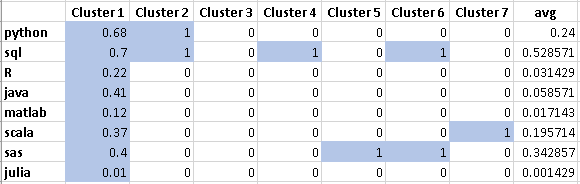


The bottom of the dendrogram contains all the data points. It cannot be viewed because of the large amount of data. The number of clusters can be chosen by drawing a horizontal line on the dendrogram. The amount of vertical lines that are crossed is the number of clusters that would be used. Consideration should be given to the number of clusters that give the most vertical distance between horizontal lines. However, it is also important to consider the dataset being observed, and which number of clusters would be appropriate. The number of clusters selected in hierarchical clustering is not widely known as an objective approach. To select the appropriate amount, several solutions should be observed to compare the best fit. In this example, it appears that a line can be drawn along the seven-cluster range. This will give us enough groups to identify which programming language pairs are commonly viewed together. The cutree() function allows us to group our data into a chosen amount of clusters. This example requires seven clusters, so ‘k’ will be set equal to seven, as shown in Figure 8.

clusterGroups = cutree(clusterLanguages, k = 7) **Figure 8**

The tapply() function used with the mean() function allows for the percentage of programming language in each cluster to be viewed. The average of each programming language is calculated. The values that are greater than the average are highlighted to be marked as significant. Cluster 1 contains job postings with a different number of occurrences of each of the programming languages. Cluster 2 contains only the job postings that contain both python and SQL, and cluster 6 contains postings that only contain both SQL and SAS. The rest of the clusters contain job postings that only contain one of the languages or none. Table 9 shows the percentage of job postings that contain the programming languages in each cluster, when k = 7.

**Table 9**



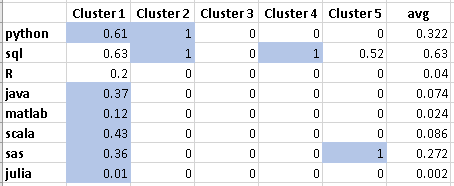
The number of observations in each cluster can be viewed by sub setting the data, as shown in Figure 9.

cluster2 <- subset(all\_cities, clusterGroups == 2) **Figure 9**

Cluster 1 contains 905 observations, with different percentages of programming languages appearing. This cluster does not give any real insight. Cluster 2, which contains only python and SQL, has 187 observations, and cluster 6, which contains only SQL and SAS, has 185 observations. Cluster 3, which contains none of the languages, contains 1,405 observations. It is possible that the number of job postings that contain the pair of programming languages is more than the number in the clusters. The importance of this analysis is that these pairs give us insight into which programming languages appear together in the same job postings most often. Based on this data, applicants for data analyst jobs can potentially improve their prospects by learning one of these programming language pairs.

A similar approach can be used to create a table with five clusters instead of seven. The two tables can be compared to check which number of clusters is a better fit. Table 10 shows the percentage of job postings that contain the programming languages in each cluster, when k = 5.

**Table 10**



The table with five clusters suggests that python and SQL is the only most common programming language pair. This reveals that there is the strongest correlation between these programming languages. The method that involves seven clusters is a better fit for this data because it identifies an additional combination of programming languages.

Recommendations

This data gives many great insights for an entry level job applicant looking to break into technology in the Northeast. Data analyst jobs are a great way to learn while on the path to becoming a data scientist. An important initial finding is that SQL is the most important programming language to learn for these roles. It appeared in more job postings than any of the other languages. Julia, a new programming language appeared in the least amount of job postings and currently is the least important to learn. However, it is important to consider that the preferred programming languages have not always remained the same. Quartz, a digital content producer, stated that “programming languages are rapidly changing” based on an annual survey from Stack Overflow.[[5]](#footnote-5) In Washington, D.C., information technology degrees appeared more than in other cities. An applicant with this type of degree might consider applying for jobs there. It is not as important to learn Scala if an applicant is seeking a position in Washington, D.C., though.

The best combinations for programming languages and degree fields all contained SQL. It is best to pair this skill with a computer science, statistics, or an engineering degree. As companies typically utilize more than one programming language, an applicant can learn multiple programming languages to improve their prospects. The most common combinations are SQL and python, and SQL and SAS. Python appears more in job postings in New York City than other cities. An applicant seeking a data analyst job in New York City can greatly improve their prospects with the SQL and python combination.

A further analysis could predict the emergence of new programming languages for the Data Analyst role. This would allow job applicants to decide if it is beneficial for them to learn a newer language, such as Julia. Currently, it does not seem very important, however if it becomes more popular then there will certainly be more occurrences in the job postings. There are specific languages that are more compatible with each other than others. Companies will often utilize a combination of languages to run their business. A clustering method could group Julia with other programming languages. Then, a model could be created that uses these commonly paired programming languages as predictors of the occurrence of Julia. A model could be created that uses training data over several time periods to predict how often Julia will appear, based on the appearances of the languages that companies are already using.

**Glassdoor Web Scrape**

# Packages

library(tidyverse)

library(dplyr)

library(rvest)

library(stringr)

library(purrr)

library(stringi)

# Set as url\_base to create data frame for desired city

New\_York\_url <- "https://glassdoor.com/Job/new-york-data-analyst-jobs-SRCH\_IL.0,8\_IC1132348\_KO9,21"

Boston\_url <- "https://www.glassdoor.com/Job/boston-data-analyst-jobs-SRCH\_IL.0,6\_IC1154532\_KO7,19"

Washington\_url <- "https://www.glassdoor.com/Job/washington-data-analyst-jobs-SRCH\_IL.0,10\_IC1138213\_KO11,23"

Philadelphia\_url <- "https://www.glassdoor.com/Job/philadelphia-data-analyst-jobs-SRCH\_IL.0,12\_IC1152672\_KO13,25"

page\_result\_start <- 1 # starting page

page\_result\_end <- 30 # last page results

full\_df <- data.frame()

for(i in page\_result\_start:page\_result\_end) {

url\_base <- New\_York\_url

url <- paste0(url\_base, "\_IP", i, ".htm")

page <- xml2::read\_html(url)

# Sys.sleep pauses R for two seconds before it resumes

# Putting it there avoids error messages such as "Error in open.connection(con, "rb") : Timeout was reached"

Sys.sleep(2)

# get the job title

job\_title <- page %>%

html\_nodes(".jobTitle") %>%

html\_text() %>%

subset(. != '')

# get the company name

company\_name <- page %>%

html\_nodes("span") %>%

html\_nodes(xpath = '//\*[@id="MainCol"]/div/ul/li/div/div/div/text()') %>%

html\_text() %>%

subset(. != 'New') %>%

subset(. != 'Hot') %>%

subset(. != "We're Hiring") %>%

subset(. != "Top Company") %>%

stri\_trim\_both()

# remove hyphens at end of company name

company\_name <- gsub('[^[:alnum:][:blank:]?&/\\-]', '', company\_name)

# get job location

job\_location <- page %>%

html\_nodes(".loc") %>%

html\_text() %>%

subset(. != '')

# get links

links <- page %>%

html\_nodes("div") %>%

html\_nodes(xpath = '//\*[@id="MainCol"]/div/ul/li/div[2]/div[1]/div[1]/a') %>%

html\_attr("href")

# get job description

job\_description <- c()

for(i in seq\_along(links)) {

url <- paste0("https://www.glassdoor.com", links[i])

result <- try({

page <- xml2::read\_html(url)

job\_description[[i]] <- page %>%

html\_nodes("div") %>%

html\_nodes('.jobDesc') %>%

html\_text() %>%

stri\_trim\_both()

} ,silent = TRUE)

if(!inherits(result, "try-error")) result

}

df <- data.frame(job\_title, company\_name, job\_location, job\_description)

full\_df <- rbind(full\_df, df)

}

**Analysis Code**

library(stringr)

library(dplyr)

library(ggplot2)

# Load data frame for each city

# New\_York <- read.csv('')

# Boston <- read.csv('')

# Washington <- read.csv('')

# Philadelphia <- read.csv('')

# create dataframe with all cities

all\_cities <- rbind(New\_York, Boston, Washington, Philadelphia)

# remove NA's from dataframes (10 total)

all\_cities <- all\_cities[complete.cases(all\_cities), ]

# remove NA's from dataframes (10 total)

New\_York <- New\_York[complete.cases(New\_York), ]

Boston <- Boston[complete.cases(Boston), ]

Washington <- Washington[complete.cases(Washington), ]

Philadelphia <- Philadelphia[complete.cases(Philadelphia), ]

# Create columns with binary variabes for programming languages

New\_York <- New\_York %>%

mutate(python = ifelse(grepl("python", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sql = ifelse(grepl("sql", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(R = ifelse(grepl("\\s[R]\\s", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(java = ifelse(grepl("java", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(matlab = ifelse(grepl("matlab", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(scala = ifelse(grepl("scala", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sas = ifelse(grepl("sas", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(julia = ifelse(grepl("julia", job\_description, ignore.case = TRUE), 1, 0))

Boston <- Boston %>%

mutate(python = ifelse(grepl("python", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sql = ifelse(grepl("sql", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(R = ifelse(grepl("\\s[R]\\s", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(java = ifelse(grepl("java", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(matlab = ifelse(grepl("matlab", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(scala = ifelse(grepl("scala", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sas = ifelse(grepl("sas", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(julia = ifelse(grepl("julia", job\_description, ignore.case = TRUE), 1, 0))

Washington <- Washington %>%

mutate(python = ifelse(grepl("python", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sql = ifelse(grepl("sql", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(R = ifelse(grepl("\\s[R]\\s", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(java = ifelse(grepl("java", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(matlab = ifelse(grepl("matlab", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(scala = ifelse(grepl("scala", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sas = ifelse(grepl("sas", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(julia = ifelse(grepl("julia", job\_description, ignore.case = TRUE), 1, 0))

Philadelphia <- Philadelphia %>%

mutate(python = ifelse(grepl("python", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sql = ifelse(grepl("sql", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(R = ifelse(grepl("\\s[R]\\s", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(java = ifelse(grepl("java", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(matlab = ifelse(grepl("matlab", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(scala = ifelse(grepl("scala", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sas = ifelse(grepl("sas", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(julia = ifelse(grepl("julia", job\_description, ignore.case = TRUE), 1, 0))

# Find percent of programming languages in job postings per city

Percent\_of\_jobs\_language <- c(mean(New\_York$python), mean(New\_York$sql), mean(New\_York$R),

mean(New\_York$java), mean(New\_York$matlab), mean(New\_York$scala),

mean(New\_York$sas), mean(New\_York$julia),

mean(Boston$python), mean(Boston$sql), mean(Boston$R), mean(Boston$java),

mean(Boston$matlab), mean(Boston$scala), mean(Boston$sas), mean(Boston$julia),

mean(Washington$python), mean(Washington$sql),

mean(Washington$R), mean(Washington$java), mean(Washington$matlab),

mean(Washington$scala), mean(Washington$sas), mean(Washington$julia),

mean(Philadelphia$python), mean(Philadelphia$sql), mean(Philadelphia$R),

mean(Philadelphia$java), mean(Philadelphia$matlab), mean(Philadelphia$scala),

mean(Philadelphia$sas), mean(Philadelphia$julia))

# Create programming language names for language data frame

language <- c('Python', 'SQL', 'R', 'Java', 'Matlab', 'Scala', 'SAS', 'Julia',

'Python', 'SQL', 'R', 'Java', 'Matlab', 'Scala', 'SAS', 'Julia',

'Python', 'SQL', 'R', 'Java', 'Matlab', 'Scala', 'SAS', 'Julia',

'Python', 'SQL', 'R', 'Java', 'Matlab', 'Scala', 'SAS', 'Julia')

# Create city names for language data frame

cities1 <- c('New York', 'New York', 'New York', 'New York',

'New York', 'New York', 'New York', 'New York',

'Boston', 'Boston', 'Boston', 'Boston',

'Boston', 'Boston', 'Boston', 'Boston',

'Washington', 'Washington', 'Washington', 'Washington',

'Washington', 'Washington', 'Washington', 'Washington',

'Philadelphia', 'Philadelphia', 'Philadelphia', 'Philadelphia',

'Philadelphia', 'Philadelphia', 'Philadelphia', 'Philadelphia')

# Create language data frame

language\_df <- tibble(cities1, language, Percent\_of\_jobs\_language)

rename(language\_df, City = cities1, Language = language, Percent\_of\_jobs = Percent\_of\_jobs\_language)

# Create graph of percent of languages for all cities combined

Percent\_of\_jobs\_language\_all <- Percent\_of\_jobs\_language/4

graph\_lang\_all <- ggplot(language\_df, aes(x = language, y = Percent\_of\_jobs\_language\_all))

graph\_lang\_all + geom\_bar(stat = 'identity') +

theme(axis.text.x = element\_text(angle = 50, hjust = 1))+

ggtitle("Language Skills For All Cities")

# Create graph of percent of languages for each city

graph\_lang\_each <- ggplot(language\_df, aes(x = language, y = Percent\_of\_jobs\_language))

graph\_lang\_each + geom\_bar(stat = 'identity') +

facet\_grid(. ~ cities1) +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

ggtitle("Language Skills For Individual Cities")

# Find percent of sql in all job postings

sql\_all <- sum(New\_York$sql, Boston$sql,

Washington$sql, Philadelphia$sql)/3544

# Find percent of julia in all job postings

julia\_all <- sum(New\_York$julia, Boston$julia,

Washington$julia, Philadelphia$julia)/3544

# Create binary columns for degree fields

New\_York <- New\_York %>%

mutate(accounting = ifelse(grepl("accounting", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(computer\_science = ifelse(grepl("computer science", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(finance = ifelse(grepl("finance", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(economics = ifelse(grepl("economics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(engineering = ifelse(grepl("engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(statistics = ifelse(grepl("statistics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(mathematics = ifelse(grepl("mathematics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(information\_technology = ifelse(grepl("information technology", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(physics = ifelse(grepl("physics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(software\_engineering = ifelse(grepl("software engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(chemistry = ifelse(grepl("chemistry", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(information\_systems = ifelse(grepl("information systems", job\_description, ignore.case = TRUE), 1, 0))

Boston <- Boston %>%

mutate(accounting = ifelse(grepl("accounting", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(computer\_science = ifelse(grepl("computer science", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(finance = ifelse(grepl("finance", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(economics = ifelse(grepl("economics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(engineering = ifelse(grepl("engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(statistics = ifelse(grepl("statistics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(mathematics = ifelse(grepl("mathematics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(information\_technology = ifelse(grepl("information technology", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(physics = ifelse(grepl("physics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(software\_engineering = ifelse(grepl("software engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(chemistry = ifelse(grepl("chemistry", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(information\_systems = ifelse(grepl("information systems", job\_description, ignore.case = TRUE), 1, 0))

Washington <- Washington %>%

mutate(accounting = ifelse(grepl("accounting", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(computer\_science = ifelse(grepl("computer science", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(finance = ifelse(grepl("finance", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(economics = ifelse(grepl("economics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(engineering = ifelse(grepl("engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(statistics = ifelse(grepl("statistics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(mathematics = ifelse(grepl("mathematics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(information\_technology = ifelse(grepl("information technology", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(physics = ifelse(grepl("physics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(software\_engineering = ifelse(grepl("software engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(chemistry = ifelse(grepl("chemistry", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(information\_systems = ifelse(grepl("information systems", job\_description, ignore.case = TRUE), 1, 0))

Philadelphia <- Philadelphia %>%

mutate(accounting = ifelse(grepl("accounting", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(computer\_science = ifelse(grepl("computer science", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(finance = ifelse(grepl("finance", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(economics = ifelse(grepl("economics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(engineering = ifelse(grepl("engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(statistics = ifelse(grepl("statistics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(mathematics = ifelse(grepl("mathematics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(information\_technology = ifelse(grepl("information technology", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(physics = ifelse(grepl("physics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(software\_engineering = ifelse(grepl("software engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(chemistry = ifelse(grepl("chemistry", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(information\_systems = ifelse(grepl("information systems", job\_description, ignore.case = TRUE), 1, 0))

# Find percent of degree fields in job postings per city

Percent\_of\_jobs\_degree <- c(mean(New\_York$accounting), mean(New\_York$computer\_science), mean(New\_York$finance),

mean(New\_York$economics), mean(New\_York$engineering), mean(New\_York$statistics),

mean(New\_York$mathematics), mean(New\_York$information\_technology),

mean(New\_York$physics), mean(New\_York$software\_engineering), mean(New\_York$chemistry),

mean(New\_York$information\_systems),

mean(Boston$accounting), mean(Boston$computer\_science), mean(Boston$finance),

mean(Boston$economics), mean(Boston$engineering), mean(Boston$statistics),

mean(Boston$mathematics), mean(Boston$information\_technology),

mean(Boston$physics), mean(Boston$software\_engineering), mean(Boston$chemistry),

mean(Boston$information\_systems),

mean(Washington$accounting), mean(Washington$computer\_science), mean(Washington$finance),

mean(Washington$economics), mean(Washington$engineering), mean(Washington$statistics),

mean(Washington$mathematics), mean(Washington$information\_technology),

mean(Washington$physics), mean(Washington$software\_engineering), mean(Washington$chemistry),

mean(Washington$information\_systems),

mean(Philadelphia$accounting), mean(Philadelphia$computer\_science), mean(Philadelphia$finance),

mean(Philadelphia$economics), mean(Philadelphia$engineering), mean(Philadelphia$statistics),

mean(Philadelphia$mathematics), mean(Philadelphia$information\_technology),

mean(Philadelphia$physics), mean(Philadelphia$software\_engineering), mean(Philadelphia$chemistry),

mean(Philadelphia$information\_systems))

# Create names for degree field data frame

degree\_field <- c('accounting', 'computer\_science', 'finance', 'economics', 'engineering', 'statistics', 'mathematics',

'information\_technology', 'physics', 'software\_engineering', 'chemistry', 'information\_systems',

'accounting', 'computer\_science', 'finance', 'economics', 'engineering', 'statistics', 'mathematics',

'information\_technology', 'physics', 'software\_engineering', 'chemistry', 'information\_systems',

'accounting', 'computer\_science', 'finance', 'economics', 'engineering', 'statistics', 'mathematics',

'information\_technology', 'physics', 'software\_engineering', 'chemistry', 'information\_systems',

'accounting', 'computer\_science', 'finance', 'economics', 'engineering', 'statistics', 'mathematics',

'information\_technology', 'physics', 'software\_engineering', 'chemistry', 'information\_systems')

# Create city names for degreee field data frame

cities2 <- c('New York', 'New York', 'New York', 'New York',

'New York', 'New York', 'New York', 'New York',

'New York', 'New York', 'New York', 'New York',

'Boston', 'Boston', 'Boston', 'Boston',

'Boston', 'Boston', 'Boston', 'Boston',

'Boston', 'Boston', 'Boston', 'Boston',

'Washington', 'Washington', 'Washington', 'Washington',

'Washington', 'Washington', 'Washington', 'Washington',

'Washington', 'Washington', 'Washington', 'Washington',

'Philadelphia', 'Philadelphia', 'Philadelphia', 'Philadelphia',

'Philadelphia', 'Philadelphia', 'Philadelphia', 'Philadelphia',

'Philadelphia', 'Philadelphia', 'Philadelphia', 'Philadelphia')

# Create data frame for degree field

degree\_field\_df <- tibble(cities2, degree\_field, Percent\_of\_jobs\_degree)

rename(degree\_field\_df, City = cities2, Degree.Field = degree\_field, Percent\_of\_jobs = Percent\_of\_jobs\_degree)

# Create graph for percentage of degree field for all cities combined

Percent\_of\_jobs\_degree\_all <- Percent\_of\_jobs\_degree/4

graph\_degree\_all <- ggplot(degree\_field\_df, aes(x = degree\_field, y = Percent\_of\_jobs\_degree\_all))

graph\_degree\_all + geom\_bar(stat = 'identity') +

theme(axis.text.x = element\_text(angle = 50, hjust = 1)) +

ggtitle("Degree Field For All Cities")

# Create graph for percentage of degree field for cities individually

graph\_degree\_each <- ggplot(degree\_field\_df, aes(x = degree\_field, y = Percent\_of\_jobs\_degree))

graph\_degree\_each + geom\_bar(stat = 'identity') +

facet\_grid(. ~ cities2) +

theme(axis.text.x = element\_text(angle = 90, hjust = 1))+

ggtitle("Degree Field For Individual Cities")

# Find percent of engineering degree in all job postings

engineering\_all <- sum(New\_York$engineering, Boston$engineering,

Washington$engineering, Philadelphia$engineering)/3544

# Create binary columns for Top Language / Degree Combinations

all\_cities <- all\_cities %>%

mutate(sql\_computer\_science = ifelse(grepl("sql", job\_description, ignore.case = TRUE) &

grepl("computer science", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sql\_engineering = ifelse(grepl("sql", job\_description, ignore.case = TRUE) &

grepl("engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sql\_finance = ifelse(grepl("sql", job\_description, ignore.case = TRUE) &

grepl("finance", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sql\_statistics = ifelse(grepl("sql", job\_description, ignore.case = TRUE) &

grepl("statistics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(python\_computer\_science = ifelse(grepl("python", job\_description, ignore.case = TRUE) &

grepl("computer science", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(python\_engineering = ifelse(grepl("python", job\_description, ignore.case = TRUE) &

grepl("engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(python\_finance = ifelse(grepl("python", job\_description, ignore.case = TRUE) &

grepl("finance", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(python\_statistics = ifelse(grepl("python", job\_description, ignore.case = TRUE) &

grepl("statistics", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sas\_computer\_science = ifelse(grepl("sas", job\_description, ignore.case = TRUE) &

grepl("computer science", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sas\_engineering = ifelse(grepl("sas", job\_description, ignore.case = TRUE) &

grepl("engineering", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sas\_finance = ifelse(grepl("sas", job\_description, ignore.case = TRUE) &

grepl("finance", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sas\_statistics = ifelse(grepl("sas", job\_description, ignore.case = TRUE) &

grepl("statistics", job\_description, ignore.case = TRUE), 1, 0))

# Create combination names

top\_names <- c('sql\_computer\_science', 'sql\_engineering', 'sql\_finance', 'sql\_statistics',

'python\_computer\_science', 'python\_engineering', 'python\_finance', 'python\_statistics',

'sas\_computer\_science', 'sas\_engineering', 'sas\_finance', 'sas\_statistics')

# Find percent of each combination in all job postings

Percent\_of\_jobs\_top <- c(mean(all\_cities$sql\_computer\_science), mean(all\_cities$sql\_engineering), mean(all\_cities$sql\_finance),

mean(all\_cities$sql\_statistics), mean(all\_cities$python\_computer\_science), mean(all\_cities$python\_engineering),

mean(all\_cities$python\_finance), mean(all\_cities$python\_statistics),

mean(all\_cities$sas\_computer\_science), mean(all\_cities$sas\_engineering), mean(all\_cities$sas\_finance),

mean(all\_cities$sas\_statistics))

# Create data frame of top programming language / degree field

top\_df <- tibble(top\_names, Percent\_of\_jobs\_top)

# Create graph for top combinations

graph\_top <- ggplot(top\_df, aes(x = top\_names, y = Percent\_of\_jobs\_top))

graph\_top + geom\_bar(stat = 'identity') +

theme(axis.text.x = element\_text(angle = 60, hjust = 1)) +

ggtitle("Top language/degree combinations")

## Clustering

# Remove unnecessary columns

all\_cities$job\_title = NULL

all\_cities$job\_location = NULL

all\_cities$company\_name = NULL

# Create data frame for clustering

all\_cities <- all\_cities %>%

mutate(python = ifelse(grepl("python", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sql = ifelse(grepl("sql", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(R = ifelse(grepl("\\s[R]\\s", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(java = ifelse(grepl("java", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(matlab = ifelse(grepl("matlab", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(scala = ifelse(grepl("scala", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(sas = ifelse(grepl("sas", job\_description, ignore.case = TRUE), 1, 0)) %>%

mutate(julia = ifelse(grepl("julia", job\_description, ignore.case = TRUE), 1, 0))

# Create euclidian distances between data points

distances = dist(all\_cities[3:10], method = "euclidian")

# Create clustering algorithm

clusterLanguages = hclust(distances, method = "ward")

# Plot dendrogram

plot(clusterLanguages)

# Cluster data into 7 groups

clusterGroups = cutree(clusterLanguages, k = 7)

# Find means of programming languages in each cluster

tapply(all\_cities$python, clusterGroups, mean)

tapply(all\_cities$sql, clusterGroups, mean)

tapply(all\_cities$R, clusterGroups, mean)

tapply(all\_cities$java, clusterGroups, mean)

tapply(all\_cities$matlab, clusterGroups, mean)

tapply(all\_cities$scala, clusterGroups, mean)

tapply(all\_cities$sas, clusterGroups, mean)

tapply(all\_cities$julia, clusterGroups, mean)

# Create subsets to tally

cluster1 <- subset(all\_cities, clusterGroups == 1)

cluster2 <- subset(all\_cities, clusterGroups == 2)

cluster3 <- subset(all\_cities, clusterGroups == 3)

cluster4 <- subset(all\_cities, clusterGroups == 4)

cluster5 <- subset(all\_cities, clusterGroups == 5)

cluster6 <- subset(all\_cities, clusterGroups == 6)

cluster7 <- subset(all\_cities, clusterGroups == 7)

# Count occurances in clusters

tally(cluster1)

tally(cluster2)

tally(cluster3)

tally(cluster4)

tally(cluster5)

tally(cluster6)

tally(cluster7)

1. DavenportD.J, Thomas H., et al. “Data Scientist: The Sexiest Job of the 21st Century.” *Harvard Business Review*, 26 May 2017, hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century. [↑](#footnote-ref-1)
2. “Data Scientist vs. Data Analyst: What's the Difference?” *Cyberbullying in the Workplace*, Kevin Carvalho Https://Www.northeastern.edu/Iuhrp/Wp-Content/Uploads/2016/09/Northeastern-Iuhrp.png, 27 Nov. 2018, www.northeastern.edu/graduate/blog/data-scientist-vs-data-analyst/. [↑](#footnote-ref-2)
3. Weller, Chris. “The 11 Most High-Tech Cities in the US.” *Business Insider*, Business Insider, 22 Aug. 2017, www.businessinsider.com/most-high-tech-cities-in-the-us-2017-8. [↑](#footnote-ref-3)
4. Deoras, Srishti. “Top 10 Programming Languages For Data Scientists to Learn In 2018.” *Analytics India Magazine*, 1 Jan. 2019, www.analyticsindiamag.com/top-10-programming-languages-data-scientists-learn-2018/. [↑](#footnote-ref-4)
5. Nisen, Max. “The Most Popular Programming Languages Are Rapidly Changing.” *Quartz*, Quartz, 8 Apr. 2015, qz.com/378939/the-most-popular-programming-languages-are-rapidly-changing/. [↑](#footnote-ref-5)