Data 607 - Project 3 - Most Valued Data Science Skills

Team X

2022-10-23

Introduction

The work presented in this analysis is meant to answer the question: "what are the most valued data science skills?"

While this is definitely a broad question to answer, there are a number of preexisting data sets that we can use to help answer this question.

The data used here comes from Kaggle, and comprises the results of a survey of ~16,000 data professionals regarding their work. The dataset includes 4 csv files (pulled directly from the Kaggle documentation):

- `schema.csv`: a CSV file with survey schema. This schema includes the questions that correspond to each column name in both the multipleChoiceResponses.csv and freeformResponses.csv.
- `multipleChoiceResponses.csv`: Respondents' answers to multiple choice and ranking questions. These are non-randomized and thus a single row does correspond to all of a single user's answers.
- `freeformResponses.csv`: Respondents' freeform answers to Kaggle's survey questions. These responses are randomized within a column, so that reading across a single row does not give a single user's answers.

*`conversionRates.csv`: Currency conversion rates (to USD) as accessed from the R package "quantmod" on September 14, 2017

The necessary steps taken to perform this analysis are outlined in the sections below. All files that we created and used can be found on Github.

Loading Data Into a Database

For the purposes of the analysis that is done later in this file, the data used is loaded in directly from the `.csv` files so that it can be viewed by any user. However, if desired the code below outlines the process for loading the data into a MySQL database. It will not be evaluated in the .Rmd file since this file has been uploaded publicly, but if someone wishes to download this file and enter their credentials on their own it should create a MySQL schema called `ds_skills` with the following three tables:

1. `multipleChoiceResponses`: Contains the information in the `multipleChoiceResponses.csv` file.
2. `freeFormResponses`: Contains the information in the `freeFormResponses.csv` file.
3. `conversionRates`: Contains the information in the `converisonRates.csv` file.

If a user desires to do this, simply enter your MySQL credentials below before running the .Rmd file.

```
#user name and password for MySQL data base connection
usr <- 'dummy_username'
pwd <- 'dummy_password'
```

Create the MySQL connection:

```
# Connect to sql server
sql_conn <- dbConnect(MySQL(), user= usr, password = pwd, host='localhost')
# Create schema if does not exist
my_dbname='ds_skills'
MySQLcode <- paste0("CREATE SCHEMA IF NOT EXISTS ",my_dbname,";",sep="")
dbSendQuery(sql_conn, MySQLcode)
# Connect to ds_skills schema
sql_conn <- dbConnect(MySQL(),
                user = usr,
                password = pwd,
                host = 'localhost',
                dbname = my_dbname)
# Enable local data loading
# We may get below error if it is not enabled
# Loading local data is disabled; this must be enabled on both the client and server sides

MySQLcode <- "SET GLOBAL local_infile=1;"
dbSendQuery(sql_conn, MySQLcode)
```

Read the data to load into database from github:

```
github_url <- "https://raw.githubusercontent.com/AlphaCurse/DATA607Project3/main/data/"
```

Read the `multipleChoiceResponses.csv` spreadsheet from github, upload it as a table into `ds_dkills` schema and than read the data from newly created database table and save it in a R dataframe:

```
file_name <-   "multipleChoiceResponses"
file_url <- paste0(github_url,file_name,".csv",sep='')
# read file
mcr_tb = read.csv(file_url, fill = TRUE)
# Drop table if exists
MySQLcode <- paste0("DROP TABLE IF EXISTS ",file_name,";",sep="")
dbSendQuery(sql_conn, MySQLcode)
# load the spreadsheet into database
dbWriteTable(sql_conn,"multipleChoiceResponses",mcr_tb,row.names = FALSE, overwrite = TRUE)
# load the database table into a R dataframe
multipleChoiceResponses <- dbGetQuery(sql_conn, 'SELECT * FROM multipleChoiceResponses')
dim(multipleChoiceResponses)
```

Note that the above code cell may fail above due to the following error: `Row size too large (> 8126).` To fix this error, simply add the following lines to your configuration file (`.my.cnf`) under the `[mysqld]` section:

```
innodb_file_per_table=1
```

```
innodb_file_format = Barracuda
```

Read the `freeformResponses.csv` file from github, upload it as a table into `ds_dkills` schema and than read the data from newly created database table and save it in a R dataframe:

```
file_name <-   "freeformResponses"
file_url <- paste0(github_url,file_name,".csv")
# read file
ffr_tb = read.csv(file_url)
# Drop table if exists
MySQLcode <- paste0("DROP TABLE IF EXISTS ",file_name,";",sep="")
dbSendQuery(sql_conn, MySQLcode)
```

```
# load the spreadsheet into database
dbWriteTable(sql_conn,"freeformResponses",ffr_tb, row.names = FALSE, overwrite = TRUE)
# load the database table into a R dataframe
freeformResponses <- dbGetQuery(sql_conn, 'SELECT * FROM freeformResponses')
dim(freeformResponses)
```

Read the `conversionRates.csv` file from github, upload it as a table into ds_dkills schema and than read the data from newly created database table and save it in a R dataframe:

```
file_name <-   "conversionRates"
file_url <- paste0(github_url,file_name,".csv")
# read file
cr_tb = read.csv(file_url)
# Drop table if exists
MySQLcode <- paste0("DROP TABLE IF EXISTS ",file_name,";",sep="")
dbSendQuery(sql_conn, MySQLcode)
# load the spreadsheet into database
dbWriteTable(sql_conn,"conversionRates",cr_tb, row.names = FALSE, overwrite = TRUE)
# load the database table into a R dataframe
conversionRates <- dbGetQuery(sql_conn, 'SELECT * FROM conversionRates')
dim(conversionRates)
```

At this point, the aforementioned three tables should have been created in the `db_skills` schema in MySQL.

Data Cleaning

Loading Data

The code below chunk below reads in the data from all of our `.csv` files:

```
link <- getURL('https://raw.githubusercontent.com/AlphaCurse/DATA607Project3/main/data/schema.csv')
schema <- read_csv(link, na=c("", "NA"))
```

```
## Rows: 290 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Column, Question, Asked
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
link <- getURL("https://raw.githubusercontent.com/AlphaCurse/DATA607Project3/main/data/multipleChoiceRes
mcq <- read_csv(link, na=c("", "NA"))
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 16716 Columns: 228
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (212): GenderSelect, Country, EmploymentStatus, StudentStatus, LearningD...
## dbl  (13): Age, LearningCategorySelftTaught, LearningCategoryOnlineCourses, ...
## num   (1): CompensationAmount
```

```
## lgl   (2): WorkToolsFrequencyAngoss, WorkToolsFrequencyKNIMECommercial
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
link <- getURL("https://raw.githubusercontent.com/AlphaCurse/DATA607Project3/main/data/freeformResponse
ff <- read_csv(link, na=c("", "NA"))
```

```
## Rows: 16716 Columns: 62
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (51): GenderFreeForm, KaggleMotivationFreeForm, CurrentJobTitleFreeForm,...
## lgl (11): LearningPlatformUsefulnessCommunitiesFreeForm, LearningPlatformUse...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
link <- getURL("https://raw.githubusercontent.com/AlphaCurse/DATA607Project3/main/data/conversionRates.
fx <- read_csv(link, na=c("", "NA"))
```

```
## New names:
## Rows: 86 Columns: 3
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): originCountry dbl (2): ...1, exchangeRate
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

As can be seen in the outupt above, all files were loaded in correctly. The `na` argument was used in the `read_csv` function to turn any empty or "NA" strings into actual `NA` values.

Cleaning the Free Form Response Questions

The following subsection outlines all the steps needed to clean the data from the free form response questions that are now stored in the `ff` dataframe.

Problem 1: Figuring out the questions that correspond to the columns in `ff`

I want to ensure that the number of questions from `mcq` and `ff` add up to all the questions listed in schema.

```
ff_colnames <- colnames(ff)
mcq_colnames <- colnames(mcq)
length(ff_colnames) + length(mcq_colnames) == nrow(schema)
```

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

I have confirmed that all the questions from both `mcq` and `ff` are enumerated in `schema`. However, at first glance, it does not appear to be the case that there is a clear way to identify which questions belong to `mcq` and which ones belong to `ff`. I would very much like to do an eyeball test of all the questions in `ff` to get a sense of what I need to think about in filtering out the questions irrelevant to data science skills.

To isolate the questions that correspond to `ff` columns, I will just use a logical test to subset out those questions in `schema`. I will save those questions to `schema_ff`.

```
schema_ff <- schema[schema$Column %in% ff_colnames, ]
nrow(schema_ff) == ncol(ff)  # Just checking that I got all the columns of ff
```

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

Looking at the questions in `schema_ff`, it turns out there **is** a way to tell if a question belongs in the `ff` dataframe. Column names of `ff` end in the characters "FreeForm", so regex was an alternative option for isolating `ff` questions. Regardless, the task has been accomplished, and now looking at the questions, it seems clear that the free form questions are complementary to the multiple choice questions. In other words, I suspect they were asked to collect follow-up information on certain multiple choice questions. Let's see if this assumption is accurate.

Problem 2: Are free form questions paired always paired with multiple choice questions?

My plan is to strip away the characters 'FreeForm' from `schema_ff$Column` values and convert them to lowercase so that a check can be performed to see how many of those values appear in the lowercase column names of `mcq`.

```
ff_colnames_modified <- tolower(
  str_match(string = schema_ff$Column,
            pattern = '(.*)[Ff]ree[Ff]orm')[, 2]
)
sum(ff_colnames_modified %in% tolower(mcq_colnames))
```

```
## [1] 15
```

Given that only 15 matches are found, perhaps I was too hasty in assuming each free form question corresponds to a multiple choice question. There is only some correspondence.

I feel the next filtering step should be done based on the value in `schema_ff$Asked`. To elaborate, some questions were asked based on the employment status of the respondent. And this information regarding selective questioning is stored in `schema_ff$Asked`. So, I want to select only those questions that were asked to data science professionals and data science learners. It is likely the respondents of these groups will be able to shed some light on the most important data science skills.

Problem 3: Excluding questions asked to people not in the data science professional or learner groups

```
unique(schema_ff$Asked)
```

```
## [1] "All"             "Non-switcher"    "Worker1"         "Learners"
## [5] "OnlineLearners"  "Worker"          "CodingWorker-NC" "CodingWorker"
```

```
mask = schema_ff$Asked %in% c('All', 'Learners', 'OnlineLearners', 'CodingWorker-NC', 'CodingWorker')
schema_ff_filtered <- schema_ff[mask, ]
```

Turns out this process only knocked off 3 columns. Now, I will use regex to sort through the questions further and identify the ones containing certain strings: 'skill', 'abilit', 'competen', 'experience', 'expert', 'job', 'employ', 'proficien', 'aptitude', 'profession', 'business', 'career', 'field', 'work', 'occupation', 'role', 'special', 'position', 'office', 'post', 'service'.

Problem 4: Filtering the questions from `schema_ff_filtered` further

I will save to a vector the values in `schema_ff_filtered$Column` that correspond to the questions containing the strings of interest. Then, I will use that vector to subset `ff` so that a dataframe is obtained that contains only the columns that correspond to relevant "data science skill" related questions.

```
strings_of_interest <- c('.*skill.*', '.*abilit.*', '.*competen.*', '.*experience.*', '.*expert.*', '.*

filtered_questions <- character()

for (i in strings_of_interest) {
  filtered_questions <- append(
    filtered_questions,
    na.omit(str_extract(schema_ff_filtered$Question, pattern = i))
```

```
  )
}

filtered_questions <- unique(filtered_questions)

# Now, I will just find the corresponding column names for each question so that I can subset those col
final_cols <- schema_ff_filtered[
  schema_ff_filtered$Question %in% filtered_questions,
  "Column"] [[1]]
ff_filtered <- ff[, final_cols]

# I am modifying schema_ff_filtered to only contain the final choice of questions
schema_ff_filtered <- schema_ff_filtered[schema_ff_filtered$Question %in% filtered_questions, ]

# Removing the objects that are no longer necessary so that the environment is not so cluttered
rm(list = c('schema_ff', 'ff_colnames', 'ff_colnames_modified', 'final_cols', 'i', 'mask', 'mcq_colname
```

`ff_filtered` is the final dataframe obtained for free-form responses: It contains only 46 of the original 62 columns. Now, it's time to clean the data.

Problem 5: Cleaning `ff_filtered`:

Let's see what proportion of values are missing in each column of `ff_filtered`, and how to deal with the missing values.

```
na_ratio_dtype <- tibble(column_name = colnames(ff_filtered),
                         na_ratio = colMeans(is.na(ff_filtered)),
                        dtype = sapply(ff_filtered, class))

'numeric' %in% na_ratio_dtype$dtype  # No column has numeric data type
```

```
## [1] FALSE
```

Clearly, much of the data is missing, and there are not really any numeric data that allow for imputation. Dropping all rows with missing values is not an option. I think the best option is to first drop columns with all values missing. Then, I can create a list that contains the existent values of each column. Then, those values can be parsed and analyzed for insights.

```
# Getting rid of columns with all values missing
ff_filtered <- ff_filtered %>%
  select(-na_ratio_dtype$column_name[na_ratio_dtype$na_ratio == 1])

# Creating a list containing non-missing values of each of the remaining 35 columns
ff_non_na <- lapply(ff_filtered, na.omit)

# Checking that there really are no missing values
any_missing <- vector()
for (i in 1:length(ff_non_na)) {
  any_missing <- append(any_missing, any(is.na(ff_non_na[[i]])))
}
any(any_missing)  # No missing value in ff_non_na
```

```
## [1] FALSE
```

```
# Cleaning up unnecessary objects
rm(list = c('i', 'any_missing'))
```

```
# It occurs to me that it would be very helpful for analysis if I provide an organized final schema con
schema_ff_filtered <- schema_ff_filtered[schema_ff_filtered$Column %in% names(ff_non_na), ]
all(schema_ff_filtered$Column == names(ff_non_na))  # Schema matches list
```

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

So, the final list containing the non-missing values from the 35 columns pertaining to data science skills is named `ff_non_na`. Information about each column (i.e., the question corresponding to each column) is contained in `schema_ff_filtered`. Hopefully, the list and its corresponding schema can be useful in finding some insights about which skills are valued by data science practitioners and learners.

Cleaning the Multiple Choice Data

Now that the free form response data has been cleaned, the following subsection outlines the steps required to clean the multiple choice response data, now stored in the `mcq` dataframe.

First, filter by necessary columns (determined in associated Google Sheet).

```
mcq <- mcq[, c("GenderSelect","Country","Age","EmploymentStatus","LearningDataScience","CodeWriter","Cu
```

Check for missing data.

```
length(is.na(mcq))
```

```
## [1] 518196
```

Create function to determine most frequent(mode) values.

```
cal_mode = function(x){
  dist_val = unique(x)#lists distinct/unique values
  dist_tab = tabulate(match(x,dist_val)) #count occurrances
  dist_val[which.max(dist_tab)]
}
```

Replace missing data with the mode of the specified column.

```
mcq = mcq %>%
  mutate(GenderSelect = if_else(is.na(GenderSelect),
                                cal_mode(GenderSelect),
                                GenderSelect))

mcq = mcq %>%
  mutate(TitleFit = if_else(is.na(TitleFit),
                                cal_mode(TitleFit),
                                TitleFit))
```

Replace missing data with the median of the specified column.

```
mcq$Age[is.na(mcq$Age)] = median(mcq$Age, na.rm=TRUE)
```

Replace missing data with specified value of the specified column.

```
mcq$Country = mcq$Country %>%
  replace_na('Other')
mcq$CurrentJobTitleSelect = mcq$CurrentJobTitleSelect %>%
  replace_na('Other')

mcq_filtered <- mcq
```

Now that each of the dataframes have been cleaned, we can finally move on to our analysis.

Analysis

All the data (both the multiple choice response data and the free response) have now been cleaned, and stored in the `mcq_filtered` and `ff_filtered` dataframes, respectively. So that we can perform our analysis using only one data set, the following code cell concatenates these two dataframes into one, named `df_clean`:

```
df_clean <- cbind(mcq_filtered, ff_filtered )
```

We can now use this new `df_clean` dataframe to answer our research question: "what are the most valuable data science skills?"

First, let's take a look at the type of job titles included in our data:

```
table(df_clean$CurrentJobTitleSelect)
```

```
##
##                      Business Analyst                   Computer Scientist
##                                   796                                  335
##                         Data Analyst                          Data Miner
##                                  1213                                  118
##                        Data Scientist                DBA/Database Engineer
##                                  2433                                  187
##                              Engineer            Machine Learning Engineer
##                                   552                                  617
##     Operations Research Practitioner                                 Other
##                                    58                                 6119
##                    Predictive Modeler                           Programmer
##                                   181                                  462
##                            Researcher                Scientist/Researcher
##                                   619                                  978
## Software Developer/Software Engineer                         Statistician
##                                  1759                                  289
```

In order to limit the scope of what the most appropriate data science skills are, the following cell limits our data to only include those who consider themselves Data Analysts or Data Scientists.

```
df_clean <- df_clean %>%
  filter(CurrentJobTitleSelect %in% c('Data Analyst', 'Data Scientist'))
```

While they are definitely other professions that could be considered under the scope of "data science", they are all a little more specialized. Picking these two titles should give us a more general idea of what skills are important to data science professionals.

Language Reccomendation

One of the fields included in our data, `LanguageRecommendationSelect` contains the survey responses to the question: What programming language would you recommend to a new data scientist learning first? The below shows a summary of how the participants responded:

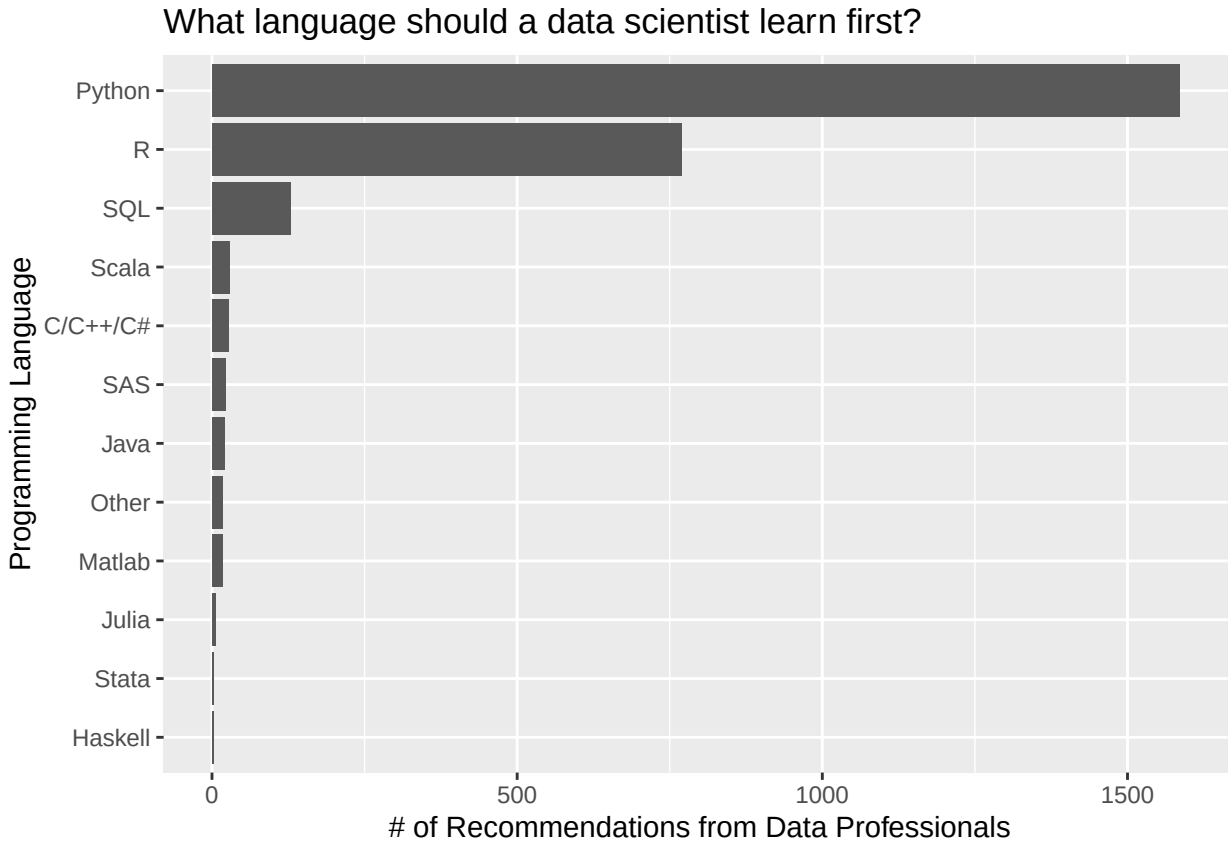
```
lang_df <- df_clean %>%
  filter(!is.na(LanguageRecommendationSelect)) %>%
  group_by(LanguageRecommendationSelect) %>%
    summarise(count_lang = n()) %>%
```

```
      arrange(desc(count_lang))

lang_df
```

```
## # A tibble: 12 x 2
##    LanguageRecommendationSelect count_lang
##    <chr>                             <int>
##  1 Python                             1585
##  2 R                                   769
##  3 SQL                                 128
##  4 Scala                                28
##  5 C/C++/C#                             27
##  6 SAS                                  22
##  7 Java                                 20
##  8 Matlab                               17
##  9 Other                                17
## 10 Julia                                 5
## 11 Stata                                 3
## 12 Haskell                               2
```

As is clear (and probably expected) Python is the language most data scientists and data analysts recommend. It was selected more than twice as much compared to the second most popular language, R. The following code chunk presents this information as a bar chart:

```
ggplot(data=lang_df, aes(x=reorder(LanguageRecommendationSelect, count_lang),
                         y=count_lang)) +
  geom_bar(stat='identity') +
  coord_flip() +
  labs(
    x = 'Programming Language',
    y = ' # of Recommendations from Data Professionals',
    title = 'What language should a data scientist learn first?'
  )
```



Technical Skills Part 1

This section will aim to go a little deeper, and determine how relevant a number of technical skills are for data scientists. Part of the survey asked participants to question the job importance of the following skills: using big data, knowing statistics, using enterprise tools, knowing Python, knowing SQL, knowing R, and how to create visualizations. A field exists for each of these questions asked in the survey, and the following cell groups these fields by their responses:

```
skill_df <- df_clean[, c("JobSkillImportanceBigData",
                    "JobSkillImportanceStats",
                    "JobSkillImportanceEnterpriseTools",
                    "JobSkillImportancePython",
                    "JobSkillImportanceR",
                    "JobSkillImportanceSQL",
                    "JobSkillImportanceVisualizations")]

skill_counts <- data.frame(matrix(ncol = 1, nrow = 3))
colnames(skill_counts) <- c('response')
skill_counts$response[1] <- 'Necessary'
skill_counts$response[2] <- 'Nice to have'
skill_counts$response[3] <- 'Unnecessary'

for(i in 1:ncol(skill_df)) {
  tmp <- skill_df[, i]
  tmp <- as.data.frame(table(tmp))
```

```
  colnames(tmp) <- c("response",
                     str_extract(colnames(skill_df)[i],
                                 "(?<=JobSkillImportance).*"))
  skill_counts <- skill_counts %>%
    left_join(tmp, by='response')
}

skill_counts <- as.data.frame(t(skill_counts))
colnames(skill_counts) <- as.character(skill_counts[1,])
skill_counts <- skill_counts[-1,]
skill_counts <- rownames_to_column(skill_counts, 'skill')
colnames(skill_counts)[3] <- 'NiceToHave'
skill_counts <- skill_counts %>%
  mutate(
    Necessary = as.integer(Necessary),
    NiceToHave = as.integer(NiceToHave),
    Unnecessary = as.integer(Unnecessary)
  )
skill_counts
```

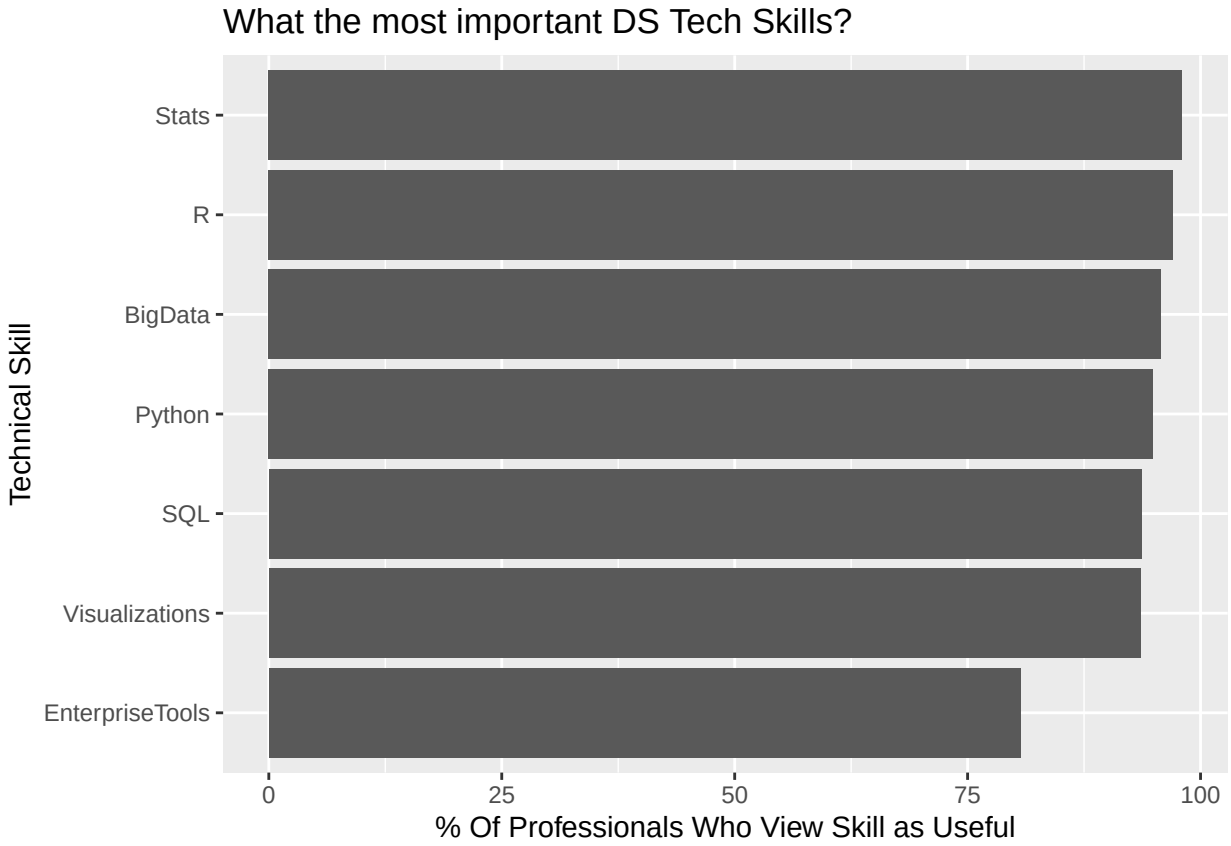
```
##             skill Necessary NiceToHave Unnecessary
## 1         BigData        44         46           4
## 2           Stats        59         40           2
## 3 EnterpriseTools        28         43          17
## 4          Python        65         28           5
## 5               R        59         39           3
## 6             SQL        55         34           6
## 7  Visualizations        54         33           6
```

Now that the data has been grouped, we can analyze which of these skills were rated as most important. We'll define a skill as useful if it was categorized as "Necessary" or "Nice to have" in the results above. The cell below calculates the percentage of participants who viewed each of these skills as useful using our definition:

```
skill_counts <- skill_counts %>%
  mutate(useful = Necessary + NiceToHave,
         useful_pct = (Necessary + NiceToHave) /
           (Necessary + NiceToHave + Unnecessary))
```

The results are summarized in the bar chart below:

```
ggplot(data=skill_counts, aes(x=reorder(skill, useful_pct),y=useful_pct*100)) +
  geom_bar(stat='identity') +
  coord_flip() +
  labs(
    x = 'Technical Skill',
    y = '% Of Professionals Who View Skill as Useful',
    title = 'What the most important DS Tech Skills?'
  )
```



As is clear in the chart above, almost every skill was rated highly, though Enterprise Tools had the largest deviance from the rest of the group. Since each of the above skills seems important on some level to Data Scientists, it appears as though we have to dig a little deeper to figure out the relevant technical chops that they most value.

Technical Skills Pt. 2

Two of the survey questions might be able to add a little more detail to the above analysis by asking the following two questions:

1. For work, which data science/analytics tools, technologies, and languages have you used in the past year?
2. At work, which algorithms/analytic methods do you typically use?

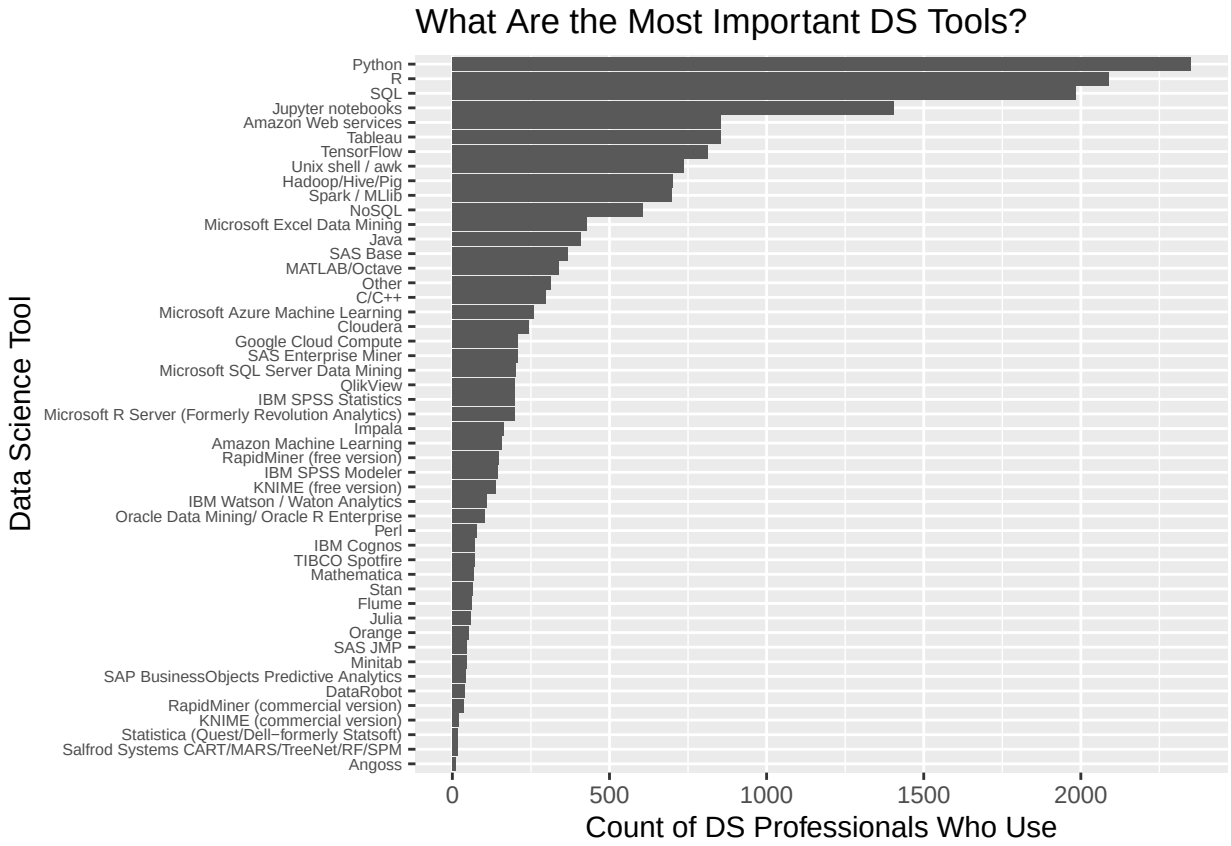
The responses to these questions are contained within the `WorkToolsSelect` and the `WorkAlgorithmSelect` fields. The following cell parses the information contained within each of them:

```
tools_used <- df_clean$WorkToolsSelect

tools_list <- list()
for(i in 1:length(tools_used)) {
  tmp <- tools_used[i]
  tmp <- strsplit(tmp, ",", fixed = TRUE)[[1]]
  tools_list <- c(tools_list, tmp)

}
```

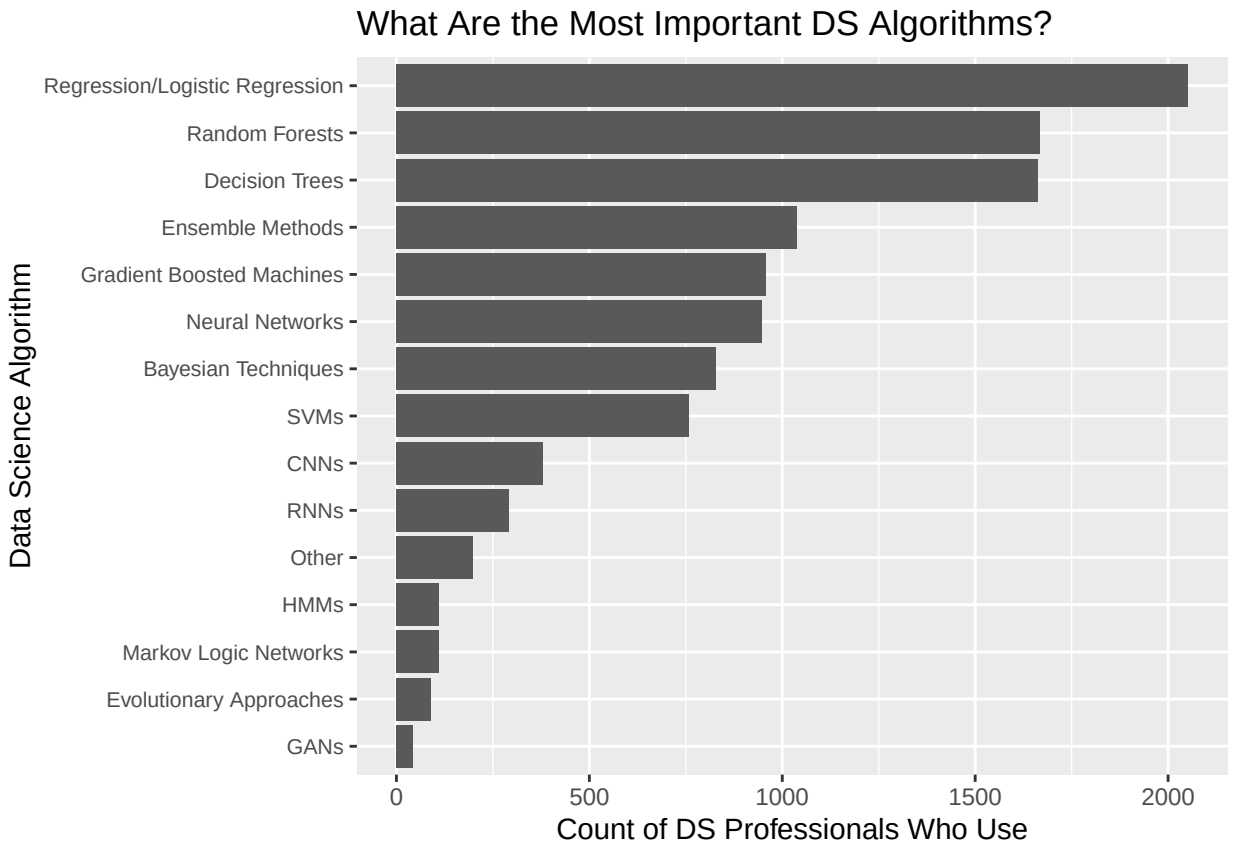
```
tools_list <- unlist(tools_list)
tools_df <- as.data.frame(table(tools_list))
colnames(tools_df) <- c('tool_used', 'count')

#-----------------------------------------

algorithms_used <- df_clean$WorkAlgorithmsSelect

algorithms_list <- list()
for(i in 1:length(algorithms_used)) {
  tmp <- algorithms_used[i]
  tmp <- strsplit(tmp, ",", fixed = TRUE)[[1]]
  algorithms_list <- c(algorithms_list, tmp)

}

algorithms_list <- unlist(algorithms_list)
algorithms_df <- as.data.frame(table(algorithms_list))
colnames(algorithms_df) <- c('algorithm_used', 'count')
```

The following code chunk summarizes the results produced above by creating bar charts of both the algorithms and tools most used by data scientists:

```
ggplot(data=tools_df, aes(x=reorder(tool_used, count),y=count)) +
  geom_bar(stat='identity') +
  coord_flip() +
  labs(
    x = 'Data Science Tool',
    y = 'Count of DS Professionals Who Use',
    title = 'What Are the Most Important DS Tools?'
  ) +
  theme(axis.text.y = element_text(size = 6))
```




```
ggplot(data=algorithms_df, aes(x=reorder(algorithm_used, count),y=count)) +
  geom_bar(stat='identity') +
  coord_flip() +
  labs(
    x = 'Data Science Algorithm',
    y = 'Count of DS Professionals Who Use',
    title = 'What Are the Most Important DS Algorithms?'
  ) +
  theme(axis.text.y = element_text(size = 8))
```



As a benchmark, we can define the most useful tools and algorithms as those that fall above the 75% quantile of count values. These quantiles are determined in the cell below and then used to filter out those tools and algorithms that are most useful:

```
tool_q <- unname(quantile(tools_df$count)[4])
algo_q <- unname(quantile(algorithms_df$count)[4])

tools_df %>%
  filter(count > tool_q) %>%
    arrange(desc(count)) %>%
      print()
```

```
##                       tool_used count
## 1                        Python  2350
## 2                             R  2088
## 3                           SQL  1983
## 4             Jupyter notebooks  1406
## 5           Amazon Web services   854
## 6                       Tableau   853
## 7                    TensorFlow   813
## 8              Unix shell / awk   738
## 9               Hadoop/Hive/Pig   702
## 10                Spark / MLlib   699
## 11                        NoSQL   605
## 12 Microsoft Excel Data Mining   429
```

```
algorithms_df %>%
  filter(count > algo_q) %>%
```

```
    arrange(desc(count)) %>%
      print()
```

```
##                   algorithm_used count
## 1 Regression/Logistic Regression  2052
## 2                 Random Forests  1667
## 3                 Decision Trees  1662
## 4               Ensemble Methods  1037
```

Being able to use tools and algorithms listed above are likely the skills that are most sought after when industries are looking to hire data scientists.

Monetary Analysis

The following analysis will attempt to answer our same research question ("what are the most valued DS skills?") by using the respondent's reported compensation as a way of defining a quantitative measure of "value". The reported salaries of each respondent is stored in the `CompensationAmount` column. However, when trying to determine the data science workers who have the skills that get them higher salaries, we encounter two complicating factors:

1. Location: Average salaries for data science professionals likely differs depending on the country they live in.
2. Experience: The longer someone has been working, the more likely they are to receive higher compensation.

The first complicating factor is relatively easy to deal with, as we can just filter out all those participants who live outside of the US:

```
comp_df <- df_clean %>%
  filter(!is.na(CompensationAmount) &
         !is.na(Tenure) &
         Country == 'United States')
```

However, despite that our data now only includes those working in the U.S., there are a number of participants that did not use the USD currency when reporting their compensation. As such, we will have to use the `conversionRates.csv` file to convert them. This is done below:

```
colnames(fx) <- c('', 'CompensationCurrency', 'exchangeRate')

comp_df <- comp_df %>%
  left_join(fx, by='CompensationCurrency')

comp_df$CompensationAmountUSD <- comp_df$CompensationAmount * comp_df$exchangeRate
```

The new field `CompensationAmountUSD` now contains all participant's compensation in United States Dollars.

Next is to tackle the experience issue. The `Tenure` field in our data frame contains info as to the number of years each participant has been working as a data science professional. The possible responses are listed below:

```
unique(comp_df$Tenure)
```

```
## [1] "1 to 2 years"       "6 to 10 years"      "3 to 5 years"
## [4] "More than 10 years" "Less than a year"
```

So that we can understand the relationship between tenure and compensation amount, the following cell coverts the tenure field into a numerical field, `experienceLevel`:

```
comp_df <- comp_df[, -67]
comp_df <- comp_df %>%
  mutate(experienceLevel = case_when(
    Tenure == 'Less than a year' ~ 1,
    Tenure == '1 to 2 years' ~ 2,
    Tenure == '3 to 5 years' ~ 3,
    Tenure == '6 to 10 years' ~ 4,
    Tenure == 'More than 10 years' ~ 5
    )
  )
```

The `experienceLevel` now contains a numerical value from 1-5, with higher values indicating higher levels of experience.

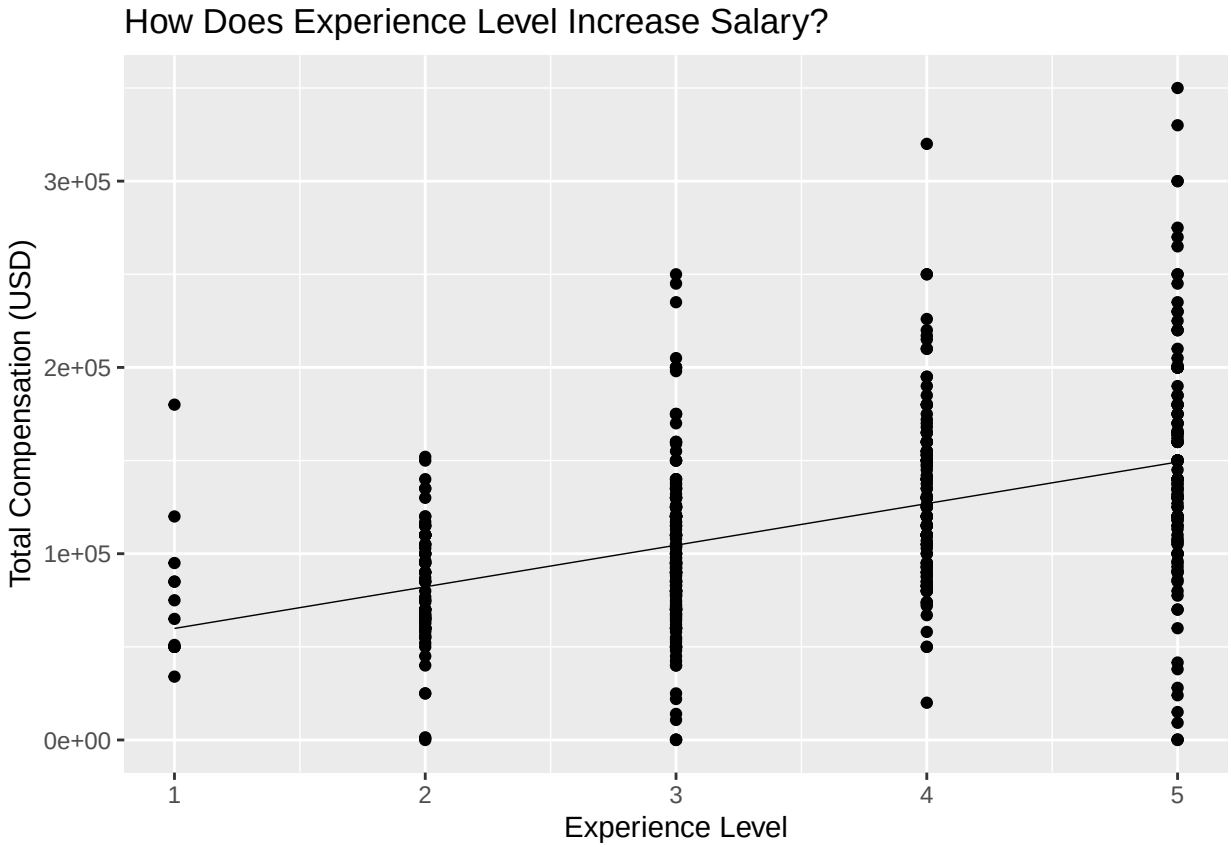
Now that both `experienceLevel` and `CompensationAmountUSD` have been created, and that they are both numerical fields, we can relate the two using a linear regression model. This is done in following cell, which creates a plot to show the relationship between `experienceLevel` and compensation amount.

```
ggplot(comp_df, aes(x=experienceLevel, y=CompensationAmountUSD)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, size=0.25, color='black') +
  labs(
    x = "Experience Level",
    y = "Total Compensation (USD)",
    title = "How Does Experience Level Increase Salary?",
  )
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning: Removed 160 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 160 rows containing missing values (geom_point).
```



As expected we see some level of correlation between the two. The exact numbers and statistics regarding the linear fit are reported below:

```
lin_mod = lm(comp_df$CompensationAmountUSD~comp_df$experienceLevel)
summary(lin_mod)
```

```
##
## Call:
## lm(formula = comp_df$CompensationAmountUSD ~ comp_df$experienceLevel)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -149213  -27041   -4541   25459  200787
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 37534       7030   5.339 1.39e-07 ***
## comp_df$experienceLevel     22336       1888  11.833  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 48070 on 535 degrees of freedom
##   (160 observations deleted due to missingness)
## Multiple R-squared:  0.2074, Adjusted R-squared:  0.2059
## F-statistic:   140 on 1 and 535 DF,  p-value: < 2.2e-16
```

The p value corresponding to the relationship between our two fields is very much below 0.05, statistically proving the correlation between the two. Thus, to try and remove the impact of experience on compensation

level, we will use the slope of the best fit line shown above to estimate the compensation of every participant if they had `experienceLevel = 1`. Given an experience level e and compensation amount C_i, the adjusted compensation amount C_f can be determined via the formula below:

$$C_f = \begin{cases} C_i & e = 1 \\ C_i - (m(e-1)) & e > 1 \end{cases}$$

where e is the slope of the best fit line shown in the plot above. The following code cells makes this adjustment in R by creating a new column `adjCompUSD`:

```
m <- unname(lin_mod[[1]][2]) # slope of the fit

comp_df <- comp_df %>%
  mutate(adjCompUSD =
         ifelse(
          experienceLevel > 1,
          CompensationAmountUSD - (m * (experienceLevel - 1)),
          CompensationAmount)
  )
```

Taking a look at the linear regression fit between `experienceLevel` and the new `adjCompUSD` reveals that there is no longer any correlation (the new p value is close to 1):

```
summary(lm(comp_df$adjCompUSD~comp_df$experienceLevel))
```

```
##
## Call:
## lm(formula = comp_df$adjCompUSD ~ comp_df$experienceLevel)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -149108  -27221   -4653   25347  200892
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)               60305.8     6909.9   8.727   <2e-16 ***
## comp_df$experienceLevel    -108.2     1860.1  -0.058    0.954
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 48070 on 538 degrees of freedom
##   (157 observations deleted due to missingness)
## Multiple R-squared:  6.284e-06,  Adjusted R-squared:  -0.001852
## F-statistic: 0.003381 on 1 and 538 DF,  p-value: 0.9537
```

Now that we have removed the possible complicating skew we might have received from the location and experience level, we can evaluate which skills correlate to the highest levels of compensation. To do this, the below cell completes a similar process that was used to determine the most commonly used tools and algorithms, but now takes compensation into account:

```
tools_used <- select(comp_df, WorkToolsSelect, adjCompUSD)

tools_df <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(tools_df) <- c('tool', 'comp')

for(i in 1:nrow(tools_used)) {
```

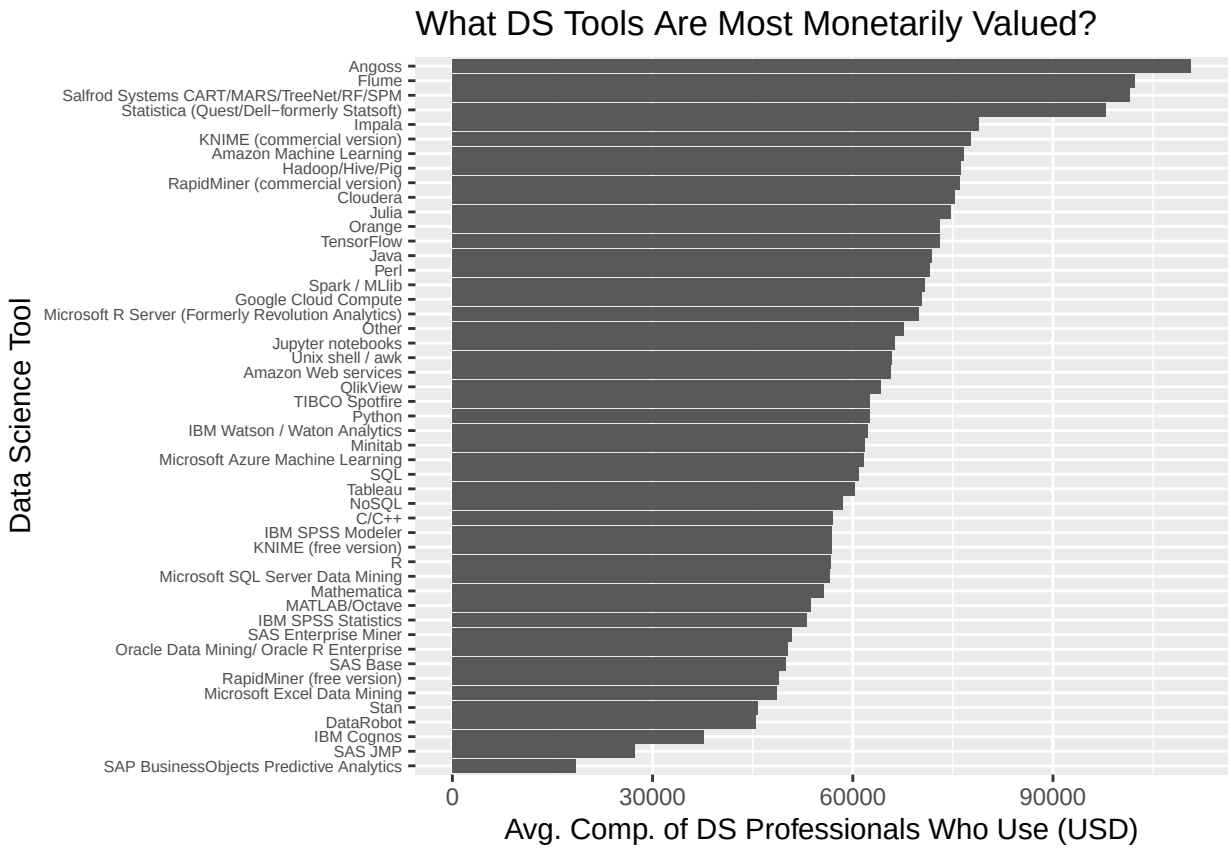
```
  tmp1 <- tools_used$WorkToolsSelect[i]
  tmp1 <- strsplit(tmp1, ",", fixed = TRUE)[[1]]
  tmp2 <- tools_used$adjCompUSD[i]
  tmp2 <- rep(tmp2, length(tmp1))
  tmp <- do.call(rbind, Map(data.frame, tool=tmp1, comp=tmp2))
  rownames(tmp) <- NULL
  tools_df <- rbind(tools_df, tmp)
}



#---------------------------------

algorithms_used <- select(comp_df, WorkAlgorithmsSelect, adjCompUSD)

algorithms_df <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(tools_df) <- c('tool', 'comp')

for(i in 1:nrow(algorithms_used)) {
  tmp1 <- algorithms_used$WorkAlgorithmsSelect[i]
  tmp1 <- strsplit(tmp1, ",", fixed = TRUE)[[1]]
  tmp2 <- algorithms_used$adjCompUSD[i]
  tmp2 <- rep(tmp2, length(tmp1))
  tmp <- do.call(rbind, Map(data.frame, algorithm=tmp1, comp=tmp2))
  rownames(tmp) <- NULL
  algorithms_df <- rbind(algorithms_df, tmp)
}
```

The cell below aggregates the results by tool or algorithm:

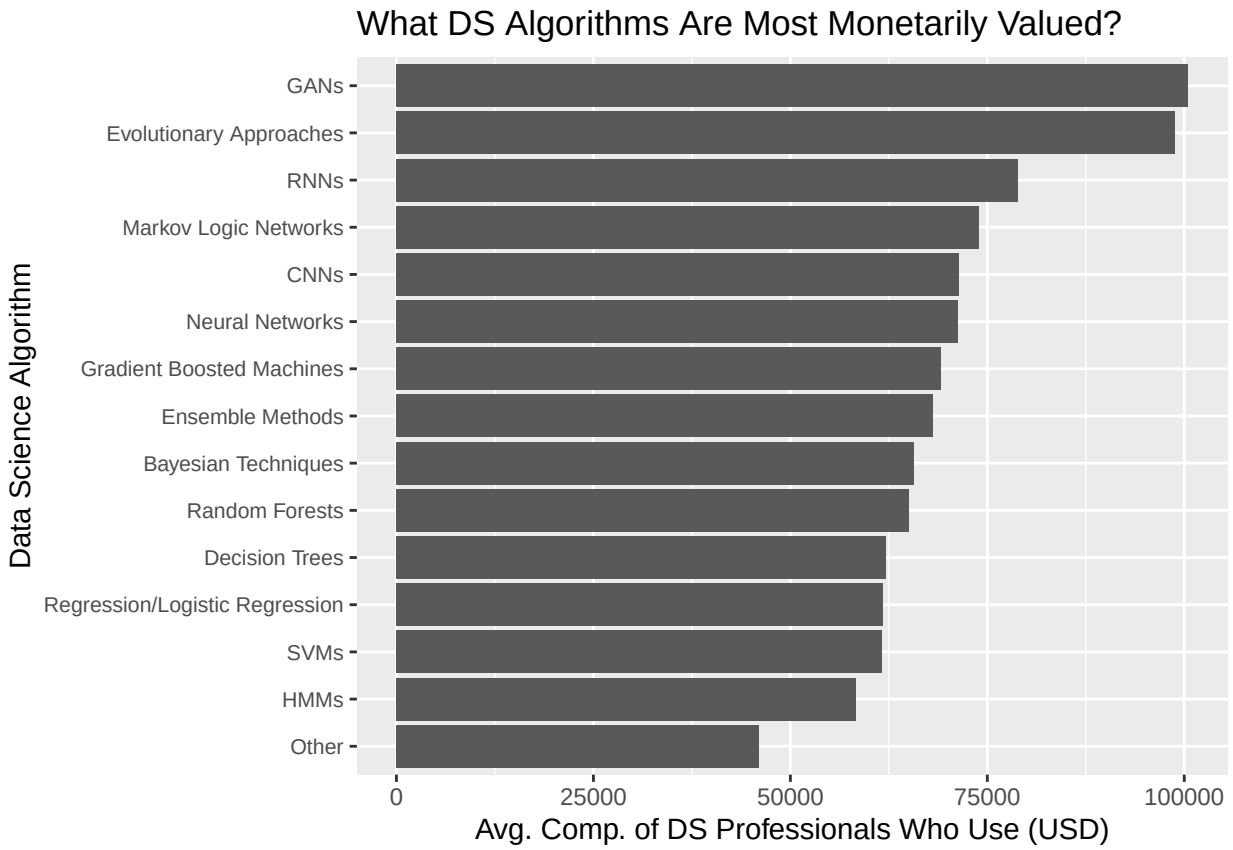
```
tools_comp <- tools_df %>%
  filter(!is.na(tool) & !is.na(comp)) %>%
    group_by(tool) %>%
      summarise(avg_comp = mean(comp), count = n()) %>%
        arrange(desc(avg_comp))

alg_comp <- algorithms_df %>%
  filter(!is.na(algorithm) & !is.na(comp)) %>%
    group_by(algorithm) %>%
      summarise(avg_comp = mean(comp), count = n()) %>%
        arrange(desc(avg_comp))
```

Finally, the cell below shows bar charts of the average compensation for each data science professional who chose either each algorithm or tool as crucial to their work performance:

```
ggplot(data=tools_comp, aes(x=reorder(tool, avg_comp),y=avg_comp)) +
  geom_bar(stat='identity') +
  coord_flip() +
  labs(
    x = 'Data Science Tool',
    y = 'Avg. Comp. of DS Professionals Who Use (USD)',
    title = 'What DS Tools Are Most Monetarily Valued?'
  ) +
  theme(axis.text.y = element_text(size = 6))
```

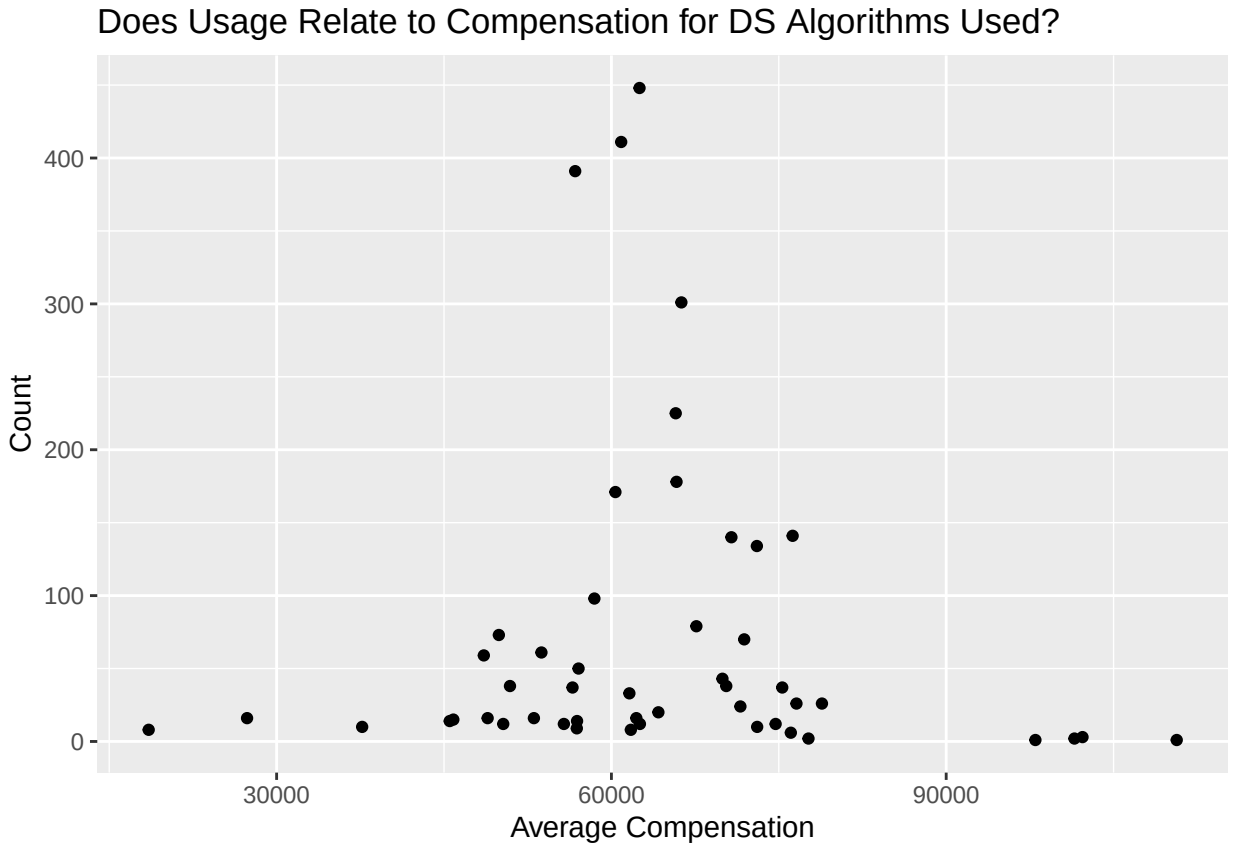



```
ggplot(data=alg_comp, aes(x=reorder(algorithm, avg_comp),y=avg_comp)) +
  geom_bar(stat='identity') +
  coord_flip() +
  labs(
    x = 'Data Science Algorithm',
    y = 'Avg. Comp. of DS Professionals Who Use (USD)',
    title = 'What DS Algorithms Are Most Monetarily Valued?'
  ) +
  theme(axis.text.y = element_text(size = 8))
```

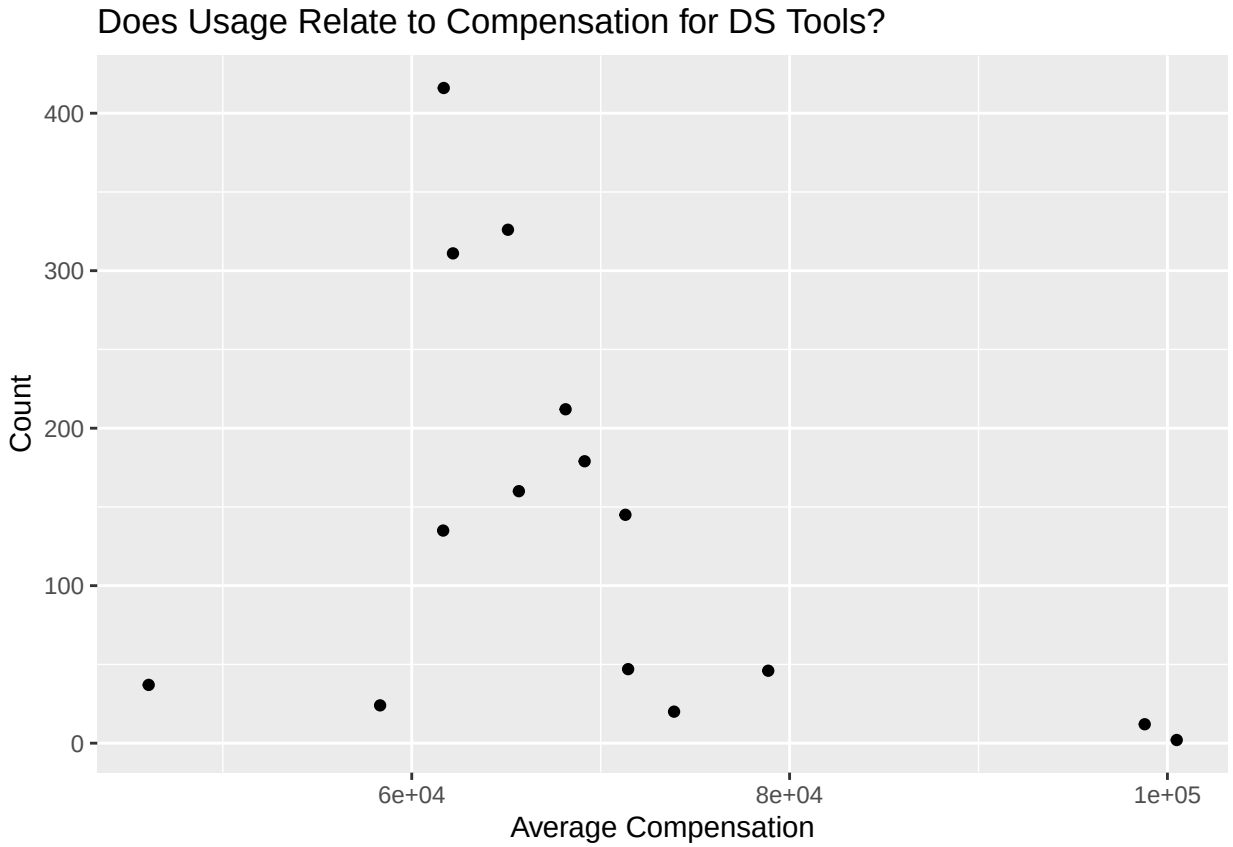


The barcharts above reveal some interesting results: a good portion of the tools/algorithms that we deemed as non-important in the earlier section actually seem to have higher total compensation (for example, GANs for algorithms, and Angoss for tools). We can evaluate this a little bit further by making some scatter plots:

```
ggplot(data=tools_comp, aes(x=avg_comp, y=count)) +
  geom_point() +
    labs(
      x = 'Average Compensation',
      y = 'Count',
      title = 'Does Usage Relate to Compensation for DS Algorithms Used?'
    )
```




```
ggplot(data=alg_comp, aes(x=avg_comp, y=count)) +
  geom_point() +
    labs(
      x = 'Average Compensation',
      y = 'Count',
      title = 'Does Usage Relate to Compensation for DS Tools?'
    )
```



When looking at the plot of the algorithms, there are some not so commonly used ones in which the average compensation of those who use are either very high or very large. The majority fo the more commonly used algorithms seem to have average compensation values that fall closer to the middle of the plot. For the tools plot, we see that there might be a slight negative correlation between the two, indicating that those who use more obscure DS tools might have higher average compensations. We can analyze this a little further by taking a look at the correlation coefficients between average compensation and usage for both tools and algorithms:

```
cor(alg_comp$count, alg_comp$avg_comp)
```

```
## [1] -0.3959509
```

```
cor(tools_comp$count, tools_comp$avg_comp)
```

```
## [1] -0.04471097
```

As we guessed, there is a negative correlation for both. However, the correlation coefficient using the algorithms is definitely more significant.

Conclusion

The above results provide us with some interesting conclusions: while there is a clear group of tools and methods that are most used by data science professionals (i.e. Python, SQL, R, etc.), there are some more highly specific skills (i.e. using Angoss, being able to implement GANs) that are less used but might provide a data scientist with a higher compensation. This makes sense, seeing as the skills that might land someone their first DS job are likely to be those that are more generalized. Once spending some time in the industry, more specialized tools will likely provide a more experienced data science professional with higher compensation.

While these results are interesting, the data used is nowhere near from being fully analyzed. The whole data set comprises the responses to about 230 data science related questions meaning that there are likely many more insights that could be drawn. In terms of this analysis, a next step might be to further inspect this negative correlation between compensation and usage via more advanced modelling techniques.