Job Posting Analysis

Zeng Fung Liew

11/29/2020

General Web-Scraping Approach

All my web-scraping approaches are relatively similar. From the search results webpage, I tried to get as much information from each job as possible, then followed up by extracting data from each of the job's webpage. The commands I have used for web-scraping are:

1) `getForm()`: Used to get the URL connections. This function is used for scraping data from cybercoders.com.
2) `GET()`: Same function as `getForm()`, but is used on the other 3 webpages since `getForm()` did not work on them for some reason.
3) `htmlParse()`: Parse the HTML webpage obtained by `getForm()` and `GET()`.
4) `xmlValue()`: Obtain the value within a HTML node.
5) `xmlGetAttr()`: Obtain the value of a certain attribute of a HTML node.
6) `xpathSApply()`: Used as a function similar to `apply()` or `sapply()` but on nodes with xpaths.

For each web-sites (cybercoders.com, monster.com, etc.), I only wrote and utilize two functions, namely `getJobInfo()` and `getJobPostings()`. `getJobInfo()` takes in the a search result node, passed to it by the `getJobPostings()` function, and extract all the required information, this includes the job position, the job location, the URL link to its full job posting, and all the important information (ie. required skills, preferred skills, etc) from the link. `getJobPostings()` on the other hand, is responsible for passing each node representing a search result to the `getJobInfo()` function, and "flipping" through the pages of the search results. The resulting output is a list of results which is then processed into a data frame, and subsequently saved as a `.RData` file.

CyberCoders

Web-Scraping

Scraping data from cybercoders.com was fairly straight-forward. The great thing about scraping data from this website is that, all the jobs come from the same company, and hence each webpage is formatted very similarly, making it easier to scrape date off it. Also, each of the node's attribute values do not change when scraping compared to what is stated on the Developer Tool's settings on the webpages.

Fortunately with the case of cybercoders.com, the results returned were relatively little, and in most cases I could get all the results in one page. The only exception was when I searched for "data analyst", where I got 26 results. In order to get the results from pages (20 in the first page and 6 in the second), I had to used the following code and then run a loop over it.

```
nextPageLink = xpathSApply(doc, "//ul[@class='pager']/li[@class='lnk-next pager-item ']/a[@rel='next']"
    doc = htmlParse(GET(paste0(url, nextPageLink)))
```

One notable thing that I did when scraping data from cybercoders.com that I did not do when scraping the other websites is organizing all the free form text in the job posting pages into Responsibilities, Requirements, and Benefits. This was possible in cybercoders.com because the headers are the same across all jobs, ie "What You Need for this Position" represents required skills, "What You Will Be Doing" represents responsibilities, and "What's In It for You" represents job benefits.

The searches returned the following:

Data Scientist - 16 results

Data Analyst - 26 results

Statistician - 0 results

Data Cleaning and Analysis

The first step in my data cleaning process is to clean out the data frame, and display a data frame that is significantly easier to read. The data frame below contains the position, company, location, employment type, minimum required education level, salary (if available), and the corresponding job posting URL link.

```
load("cyberCoders.RData")

# minimum education level
minEducationLevel = factor(sapply(cyberCoders$requiredSkills, function(desc){
  if (grepl("high school diploma", desc, ignore.case = TRUE) == TRUE) {
    return ("High School")
  } else if (grepl("(\\<A[.]?A\\>|\\<A[.]?S\\>|Associate.?s degree)", desc) == TRUE){
    return ("Associate's")
  } else if (grepl("(\\<B[.]?A\\>|\\<B[.]?S\\>|Bachelor[']?s?)", desc) == TRUE){
    return ("Bachelor's")
  } else if (grepl("(M[.]?S\\>|Master.?s|Advanced [Dd]egree)", desc) == TRUE) {
    return ("Master's")
  } else if (grepl("(\\<Ph[.]?[Dd]|[Dd]octoral[ [Dd]egree]?)", desc) == TRUE) {
    return ("Ph.D")
  } else {
    return ("None")
  }
}), levels = c("None", "High School", "Associate's", "Bachelor's", "Master's", "Ph.D"))

# location
city = trimws(gsub("([[:alpha:] ]+),([[:space:][:alpha:]]+)", "\\1", cyberCoders$location))
state = trimws(gsub("([[:alpha:] ]+),([[:space:][:alpha:]]+)", "\\2", cyberCoders$location))

# employment type
employmentTypeRegex = "(full[- ]?time|part[- ]time|hourly|contract|short[- ]term|month)"
getEmploymentType = function(employmentType, description){
  if (!is.na(employmentType)){
    return (employmentType)
  } else{
    if (grepl(employmentTypeRegex, description, ignore.case = TRUE) == TRUE){
      m = regexpr(employmentTypeRegex, description, ignore.case = TRUE)
      result = regmatches(description, m)
```

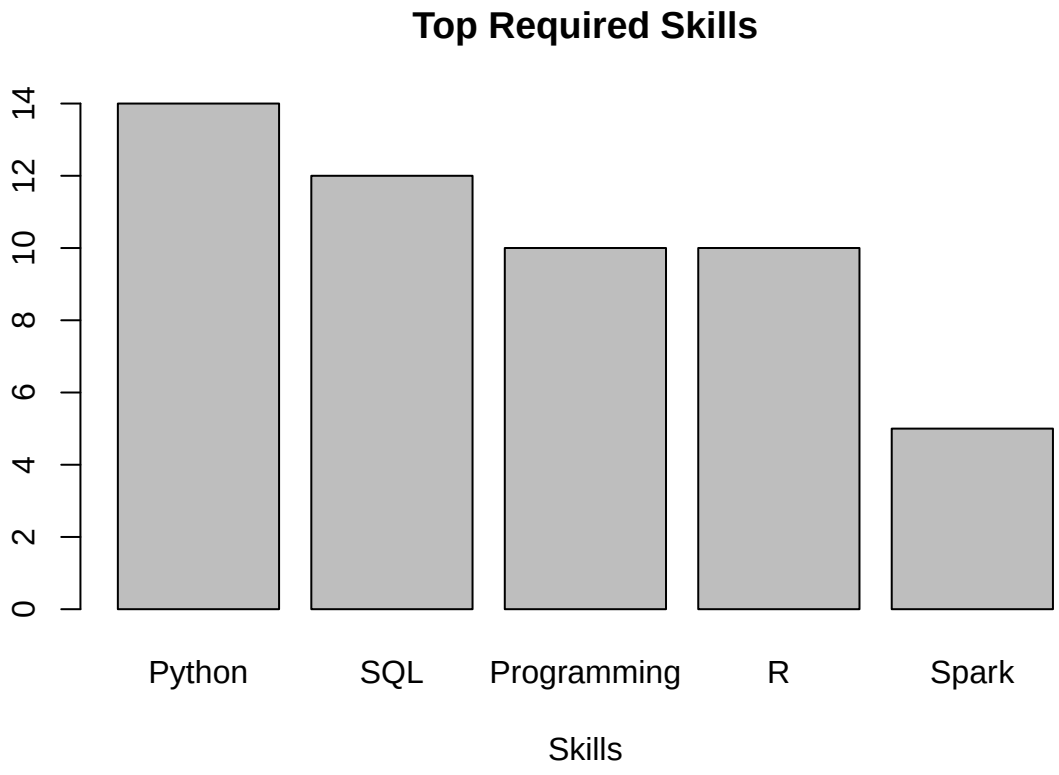
```
      employmentType = switch(tolower(result),
                              "full-time" = "Full-time",
                              "full time" = "Full-time",
                              "part-time" = "Part-time",
                              "part time" = "Part-time",
                              "hourly" = "Hourly",
                              "contract" = "Contract",
                              "short-term" = "Short-term",
                              "short term" = "Short-term",
                              "month" = "Contract")
      return (employmentType)
    } else{
      return (NA)
    }
  }
}
employmentType = mapply(getEmploymentType, cyberCoders$employmentType, cyberCoders$description)

# forming cleaned data frame
cyberCoders_df = data.frame(Position = cyberCoders$title,
                            Company = cyberCoders$company,
                            City = city,
                            State = state,
                            EmploymentType = employmentType,
                            MinEducationLevel = minEducationLevel,
                            Salary = cyberCoders$salary,
                            Link = cyberCoders$link)
head(cyberCoders_df)
```

```
##                                                    Position     Company
## 1        Data Scientist - Machine Learning, Building Information CyberCoders
## 2                                        Senior Data Scientist CyberCoders
## 3 Senior Data Scientist-100% REMOTE (Local candidates preferred) CyberCoders
## 4                                        Senior Data Scientist CyberCoders
## 5 Senior Data Scientist-100% REMOTE (Local candidates preferred) CyberCoders
## 6 Senior Data Scientist-100% REMOTE (Local candidates preferred) CyberCoders
##            City State EmploymentType MinEducationLevel        Salary
## 1    Pittsburgh    PA      Full-time        Bachelor's  $50k - $100k
## 2       Chicago    IL      Full-time             None          <NA>
## 3    Alexandria    VA      Full-time          Master's $150k - $175k
## 4 Buffalo Grove    IL      Full-time             None          <NA>
## 5    Alexandria    VA      Full-time          Master's $150k - $180k
## 6    Alexandria    VA           <NA>          Master's          <NA>
##                                                      Link
## 1        https://www.cybercoders.com/data-scientist-job-552883
## 2 https://www.cybercoders.com/senior-data-scientist-job-559755
## 3 https://www.cybercoders.com/senior-data-scientist-job-565148
## 4 https://www.cybercoders.com/senior-data-scientist-job-559602
## 5 https://www.cybercoders.com/senior-data-scientist-job-566266
## 6 https://www.cybercoders.com/senior-data-scientist-job-564834
```

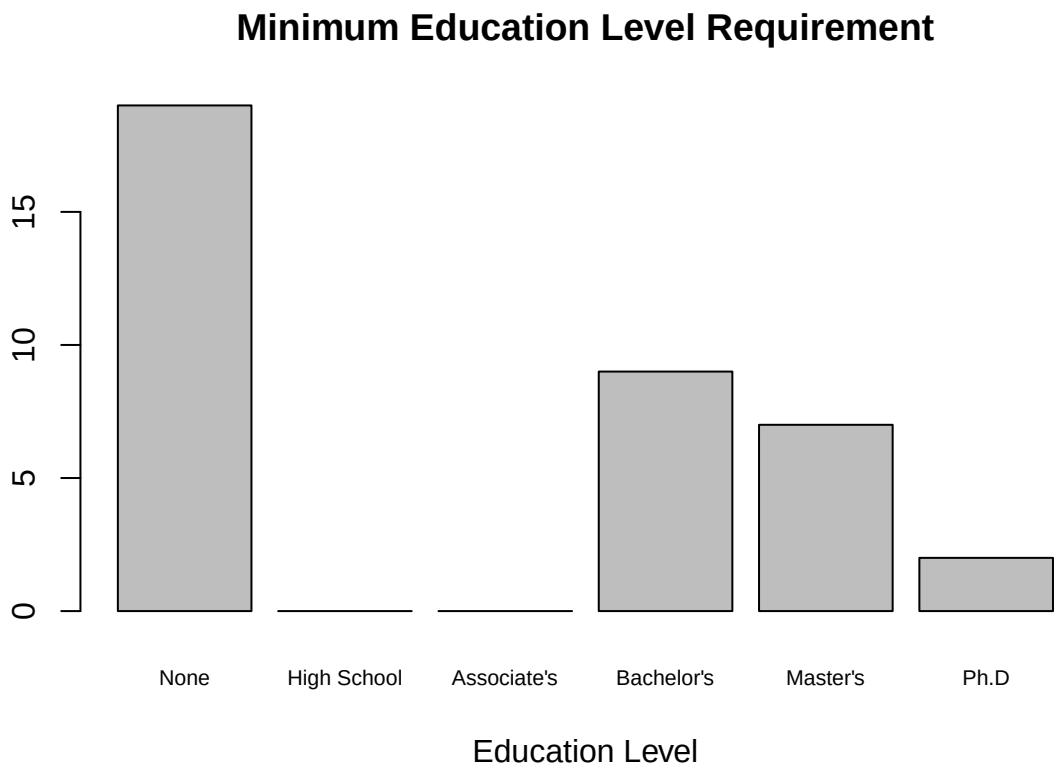
Now, I will move on to conducting analysis across all jobs from cybercoders.com. The first things we will look into are the distributions of required skills and minimum education levels. These data are mostly obtained

by splitting all the text from the required skills section into a large vector of words, and then calculate the occurence of each word. I then selected the top skills based on the calculation. To assist myself in manually doing this task, I have used the `stopwords` function from the `tm` library, to remove stopwords from the list of words.

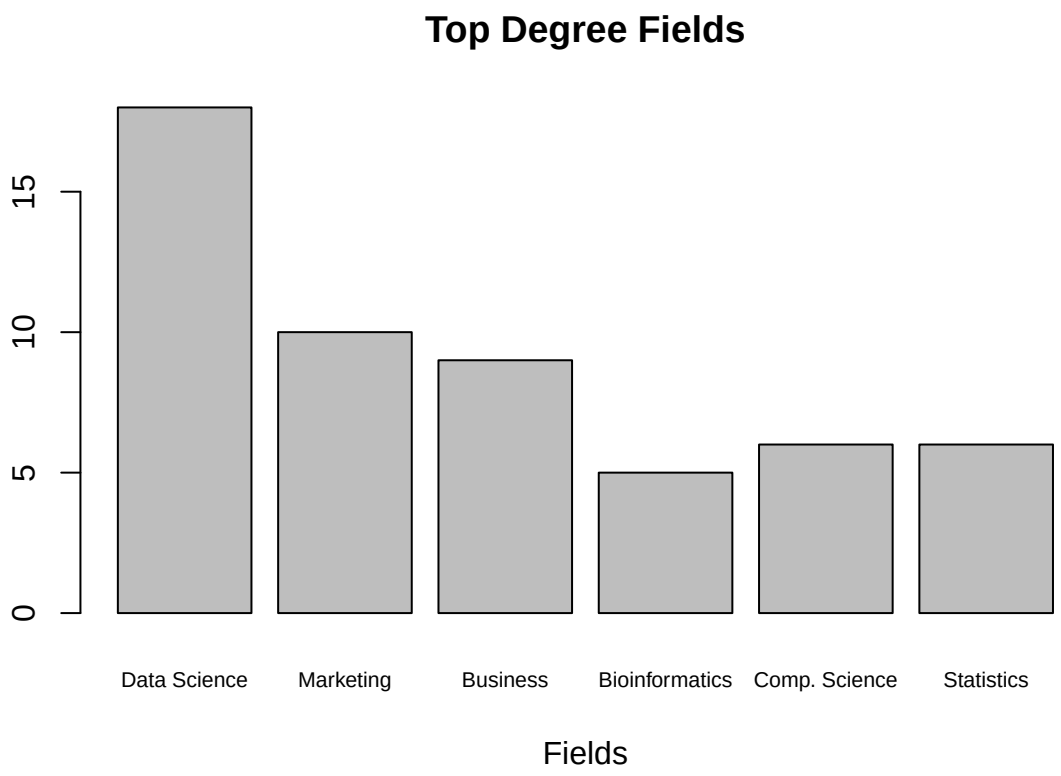
```
stopWords = tm::stopwords("en")
# required skills
reqSkills = tolower(unlist(strsplit(cyberCoders$requiredSkills, "@@")))
reqSkills_keywords = unlist(lapply(reqSkills, function(x) {
  m = gregexpr("([[:alnum:]]+[[:punct:]][[:alnum:]]+|[[:alnum:]]+)", x)
  results = regmatches(x, m)
  return (results)
}))
reqSkills_keywords = reqSkills_keywords[!(reqSkills_keywords %in% stopWords)]
reqSkills_table = sort(table(reqSkills_keywords), decreasing = TRUE)
topReqSkills = reqSkills_table[c("python", "sql", "programming", "r", "spark")]
names(topReqSkills) = c("Python", "SQL", "Programming", "R", "Spark")
barplot(topReqSkills, main = "Top Required Skills", xlab = "Skills")
```




```
# minimum required education level
minEducationLevel_table = table(minEducationLevel)
barplot(minEducationLevel_table, main = "Minimum Education Level Requirement", xlab = "Education Level"
```

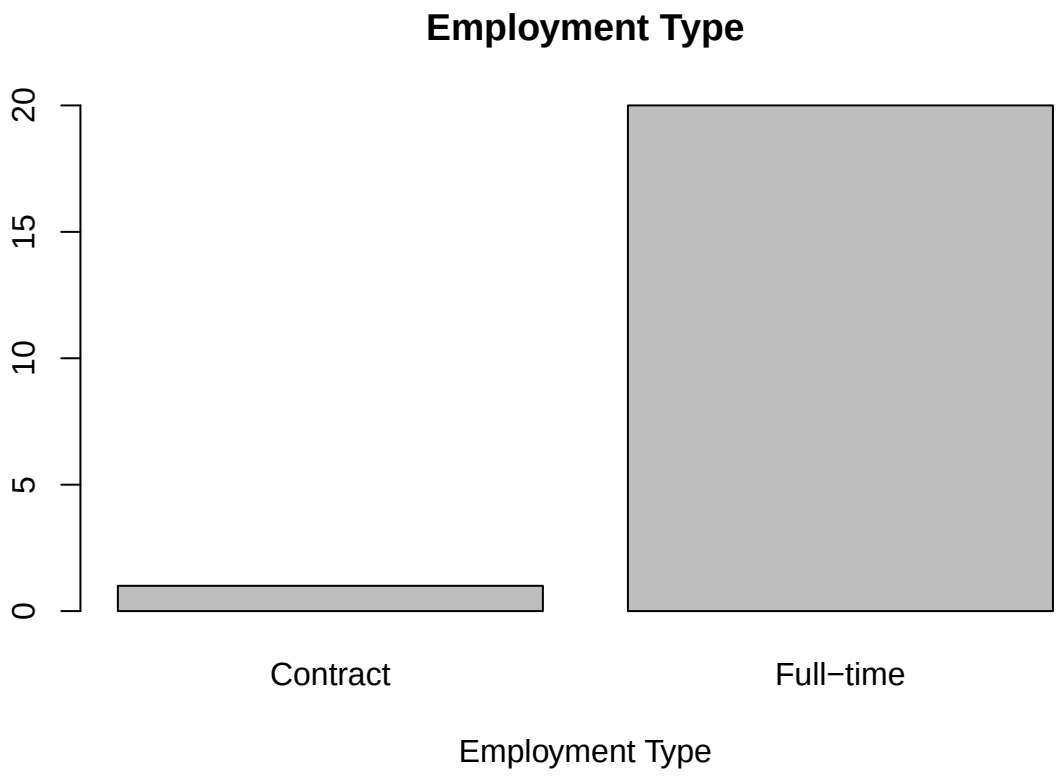



```
# top degree fields
topFields = reqSkills_table[c("science", "marketing", "business", "bioinformatics", "computer", "statis
names(topFields) = c("Data Science", "Marketing", "Business", "Bioinformatics", "Comp. Science", "Stati
barplot(topFields, main = "Top Degree Fields", xlab = "Fields", cex.names = 0.7)
```

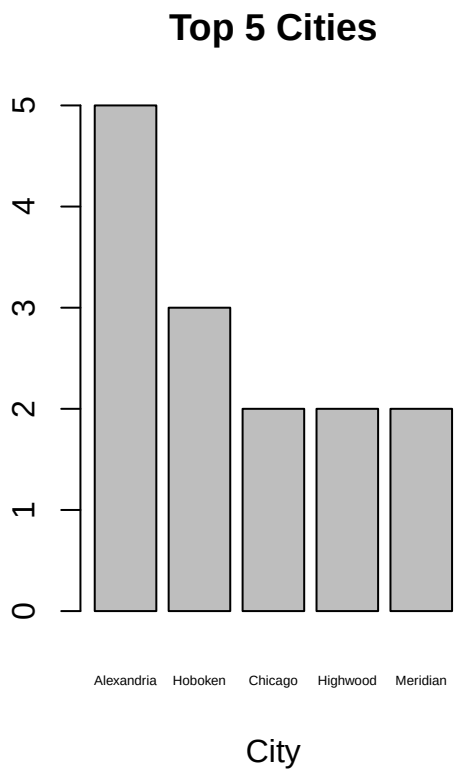


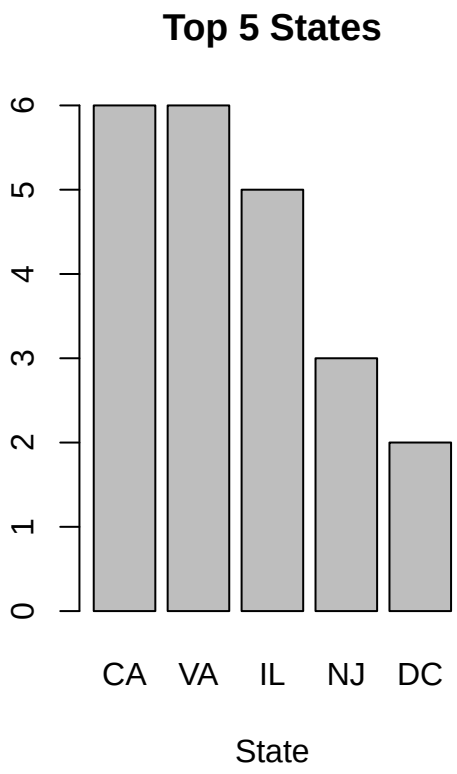
Some other plots are shown below:

```
# employment types
barplot(table(employmentType), main = "Employment Type", xlab = "Employment Type")
```

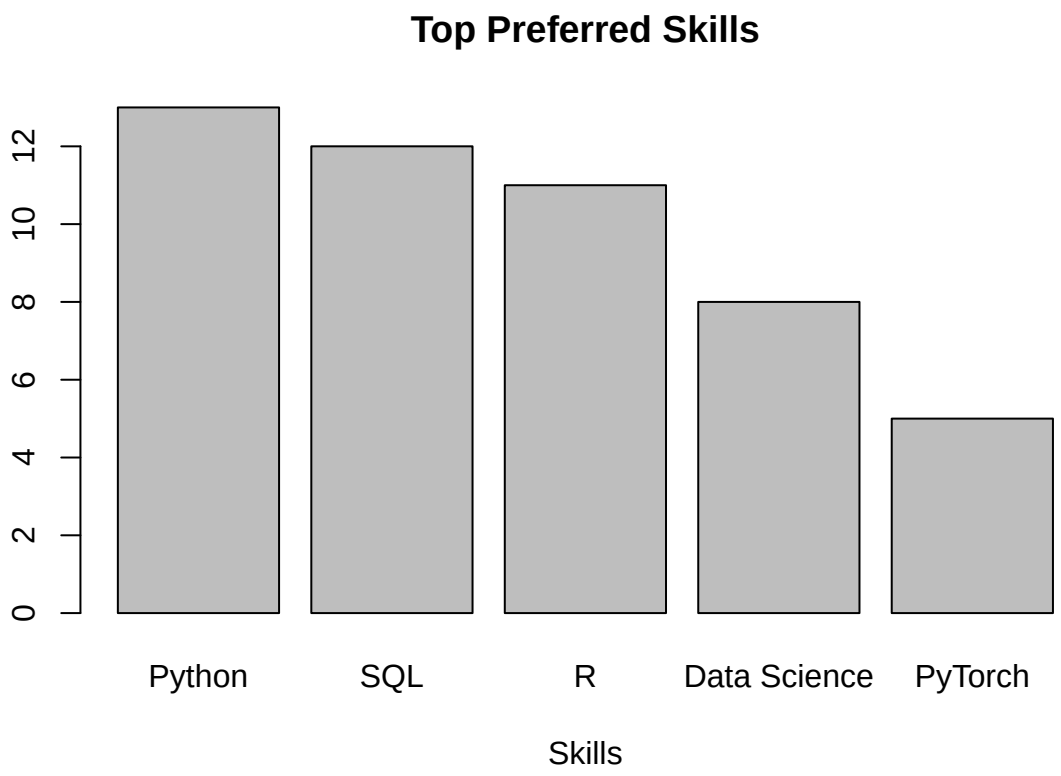



```
# location
par(mfrow = c(1,2))
barplot(sort(table(city), decreasing = TRUE)[1:5], main = "Top 5 Cities", xlab = "City", cex.names = 0.
barplot(sort(table(state), decreasing = TRUE)[1:5], main = "Top 5 States", xlab = "State")
```






```r
# top preferred skills
par(mfrow = c(1,1))
prefSkills = unlist(strsplit(cyberCoders$preferredSkills, "@@"))
prefSkills_table = sort(table(prefSkills), decreasing = TRUE)
topPrefSkills = prefSkills_table[1:5]
barplot(topPrefSkills, main = "Top Preferred Skills", xlab = "Skills")
```



Based on the plots above, here are a few things I have noticed:

- Python, SQL, and R are very important tools to get a data science/statistics related job at CyberCoders. These skills are some of the top required and preferred skills.
- A lot of jobs do not specify the education level required to get into these jobs, but those that did mostly required at least a Bachelor's or Master's degree.
- While it is obvious that Data Science , Computer Science and Statistics are the most sought after degree fields here, it is interesting to find Marketing, Business, and Bioinformatics to be up there as well.
- Almost all jobs are looking for full-time employment.
- What's interesting about the location bar charts is that California, being the city with the joint highest number of job postings, does not have any of its city among the top 2 of that category. This suggests that CyberCoder jobs are fairly spread out in California, with job postings coming from Los Angeles, Mountain View, South San Francisco etc.

Career Builder

Web-Scraping

Scraping this website is also relatively straightforward, the only difference here compared to CyberCoders is that Career Builder preoduces significantly more search results with the same search queries. Due to the time constraints (as my computer is taking too long to scrape all jobs across the United States), I have only scraped jobs from the state of California.

With multiple pages of results being output from the search queries, I had to find a way to programmatically "click" and browse the next page of search results. The solution that I came up with was to use a `while` loop to browse through all pages. How this works is that as long as I am able to find the button that leads me to the next page, ie. the node `<div id="load_more_jobs">`, I extract the URL link to the next page using `xmlGetAttr()` on the attribute `data-load-more` and repeat the same processes in the next page.

The searches returned the following:

Data Scientist - 623 results

Data Analyst - 1332 results

Statistician - 35 results

Data Cleaning/Analysis

Cleaning this data is also relatively simple just like in CyberCoders, since most of the required fields are already sorted and only require at most some trimming of data. The cleaned version of the data frame is shown below:

```
load("careerBuilder.RData")

# minimum education level
minEducationLevel = factor(sapply(careerBuilder$description, function(desc){
  if (grepl("high school diploma", desc, ignore.case = TRUE) == TRUE) {
    return ("High School")
  } else if (grepl("(\\<A[.]?A\\>|\\<A[.]?S\\>|Associate.?s degree)", desc) == TRUE){
    return ("Associate's")
  } else if (grepl("(\\<B[.]?A\\>|\\<B[.]?S\\>|Bachelor[']?s?)", desc) == TRUE){
    return ("Bachelor's")
  } else if (grepl("(M[.]?S\\>|Master.?s|Advanced [Dd]egree)", desc) == TRUE) {
    return ("Master's")
  } else if (grepl("(\\<Ph[.]?[Dd]|[Dd]octoral[ [Dd]egree]?)", desc) == TRUE) {
    return ("Ph.D")
  } else {
    return ("None")
  }
}), levels = c("None", "High School", "Associate's", "Bachelor's", "Master's", "Ph.D"))

# location
city = trimws(gsub("([[:alpha:] ]+)-([[:space:][:alpha:]]+)", "\\2", careerBuilder$location))
state = trimws(gsub("([[:alpha:] ]+)-([[:space:][:alpha:]]+)", "\\1", careerBuilder$location))

# employment type
employmentType = sapply(careerBuilder$employmentType, function(x) switch(x,
                        "Contract to Hire" = "Contract",
                        "Contractor" = "Contract",
                        "Full-Time" = "Full-time",
                        "Full-Time/Part-Time" = "Full/Part-time",
                        "Intern" = "Intern",
                        "Not Specified" = "NA",
                        "Part-Time" = "Part-time",
                        "Per Diem" = "Hourly",
                        "Seasonal/Temp" = "Short-term"))
```

```r
# cleaned data frame
careerBuilder_df = data.frame(Title = careerBuilder$title,
                              Company = careerBuilder$company,
                              City = city,
                              State = state,
                              EmploymentType = employmentType,
                              MinEducationLevel = minEducationLevel,
                              Link = careerBuilder$link)
head(careerBuilder_df)
```

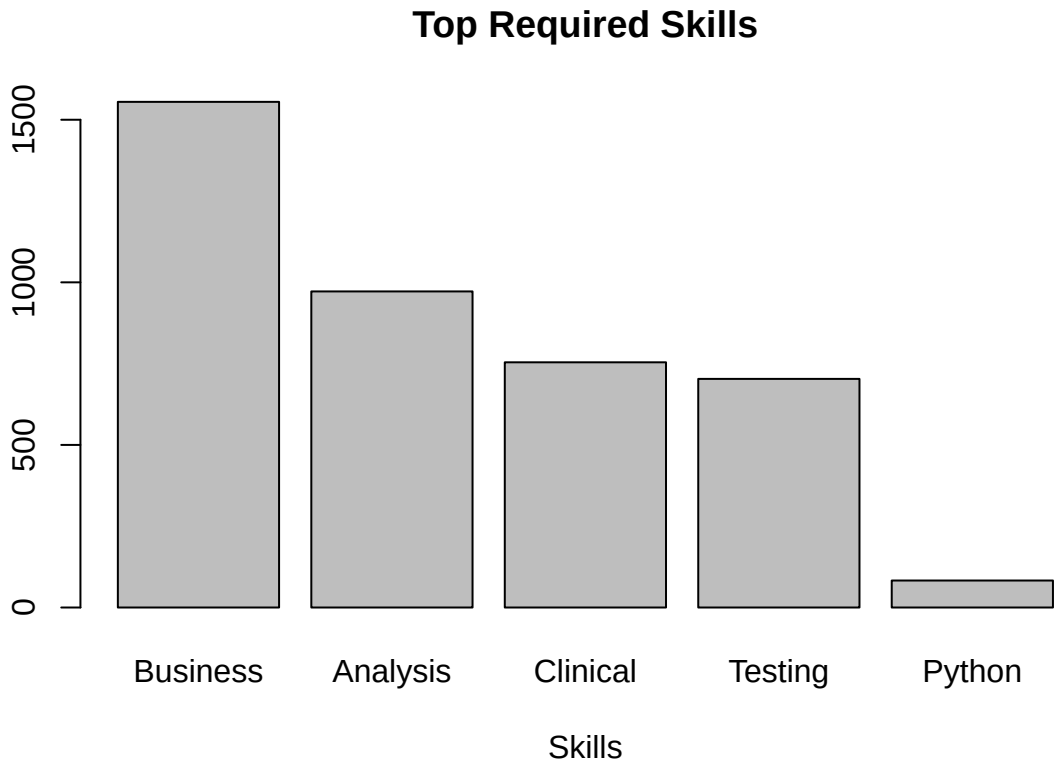
```
##                                                      Title
## 1         IT - Data Scientist : On-Site after SIP : W2 contract
## 2 Systems Verification Scientist II (Must Have Flow Cytometry))
## 3              Senior Data Scientist (Cardio Signal Processing)
## 4                                                Data Scientist
## 5                     Data Scientist/Machine Learning Engineer
## 6                                      OneSource Data Analyst
##                  Company           City State EmploymentType MinEducationLevel
## 1            Finezi Inc.  San Francisco    CA       Contract        Bachelor's
## 2         Millenniumsoft       San Jose    CA       Contract        Bachelor's
## 3        Mastech Digital    Los Angeles    CA      Full-time        Bachelor's
## 4                Experis Rancho Cordova    CA      Full-time              None
## 5               Analysts      Palo Alto    CA       Contract              None
## 6 OneSource Distributors      Oceanside    CA      Full-time        Bachelor's
##                                                          Link
## 1 https://www.careerbuilder.com/job/J2W0YW67P8DBWN8LPFC?ipath=CRJR1
## 2 https://www.careerbuilder.com/job/J2P7HH6KJXYQ9V1NCR2?ipath=CRJR2
## 3 https://www.careerbuilder.com/job/J3M2SK6KXGY0RQQYY4T?ipath=CRJR3
## 4 https://www.careerbuilder.com/job/J3R8J362FMRYRHZ40LC?ipath=CRJR4
## 5 https://www.careerbuilder.com/job/J3Q5P076KBGO95JYHFQ?ipath=CRJR5
## 6 https://www.careerbuilder.com/job/J3S15W6L3N91BTL4TSZ?ipath=CRJR6
```

The challenge in analyzing this data, however, is finding out the degree fields and the required skills required by companies in those job postings. This is due to the fact that all of these data are available as free form text in the job description page.

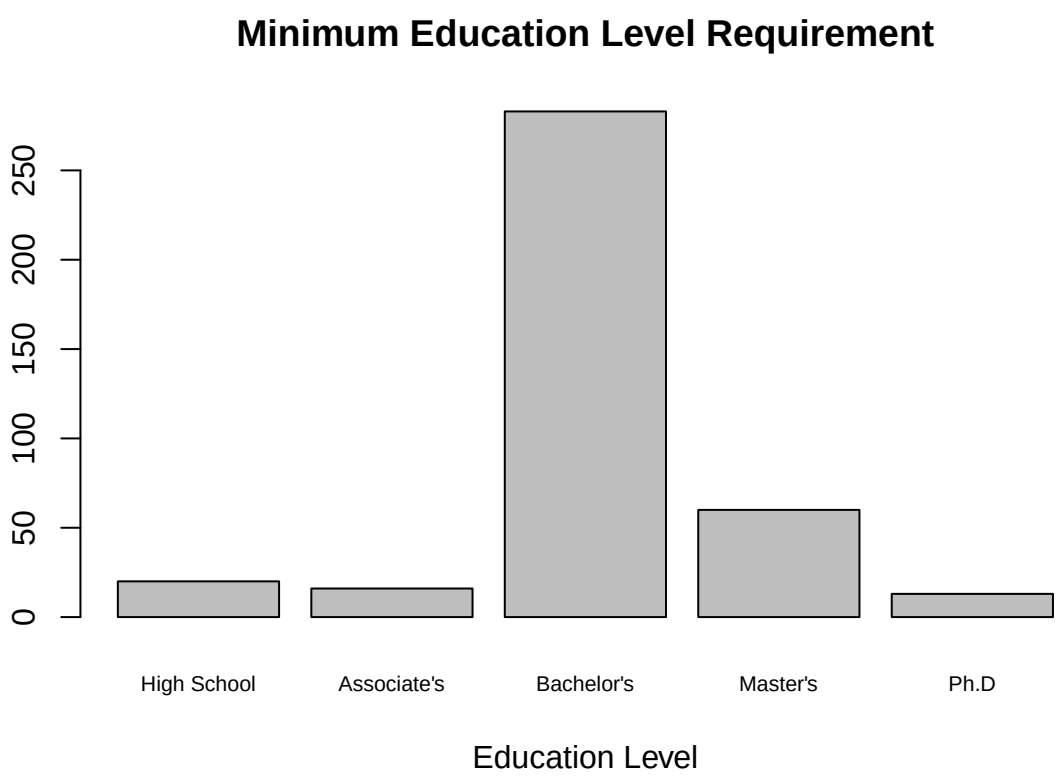
Once again, I combined all the free form texts into a huge vector of words, then calculate the recurrence of each word in this huge vector, before finding the important keywords relating to skills and degree fields. The problem here is, however, that the vector contains too many "connecting" words that are useless for my analysis (eg. "and", "a", "the"), which makes it harder for me to look for the keywords I need.

Overall, the results are shown below:

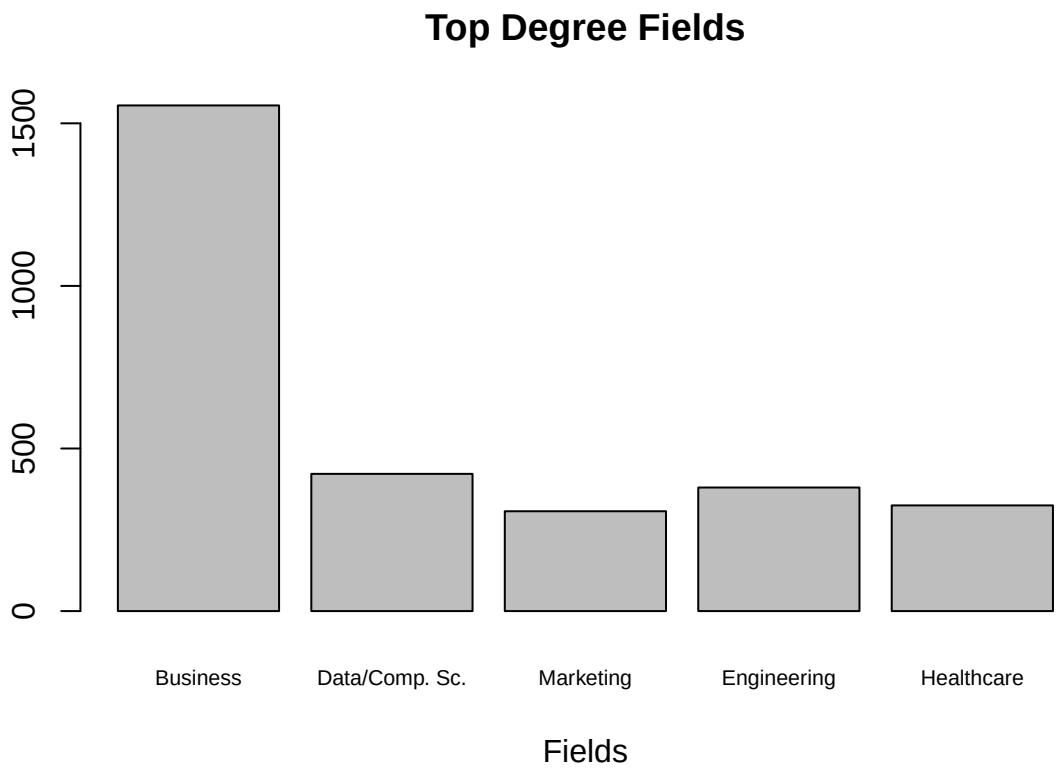
```r
# required skills
reqSkills = tolower(unlist(strsplit(careerBuilder$description, "@@")))
reqSkills_keywords = unlist(lapply(reqSkills, function(x) {
  m = gregexpr("([[:alnum:]]+[[:punct:]][[:alnum:]]+|[[:alnum:]]+)", x)
  results = regmatches(x, m)
  return (results)
}))
reqSkills_keywords = reqSkills_keywords[!(reqSkills_keywords %in% stopWords)]
reqSkills_table = sort(table(reqSkills_keywords), decreasing = TRUE)
topReqSkills = reqSkills_table[c("business", "analysis", "clinical", "testing", "python")]
names(topReqSkills) = c("Business", "Analysis", "Clinical", "Testing", "Python")
barplot(topReqSkills, main = "Top Required Skills", xlab = "Skills")
```



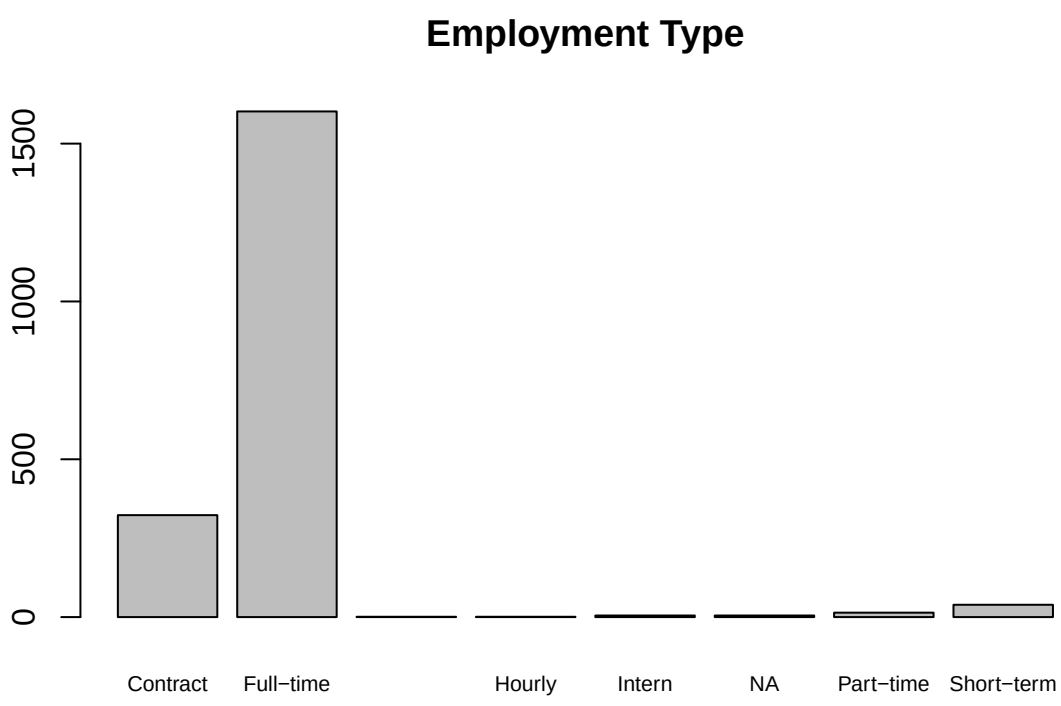

```
# minimum required education level
minEducationLevel_table = table(minEducationLevel)
barplot(minEducationLevel_table[2:6], main = "Minimum Education Level Requirement", xlab = "Education L
```



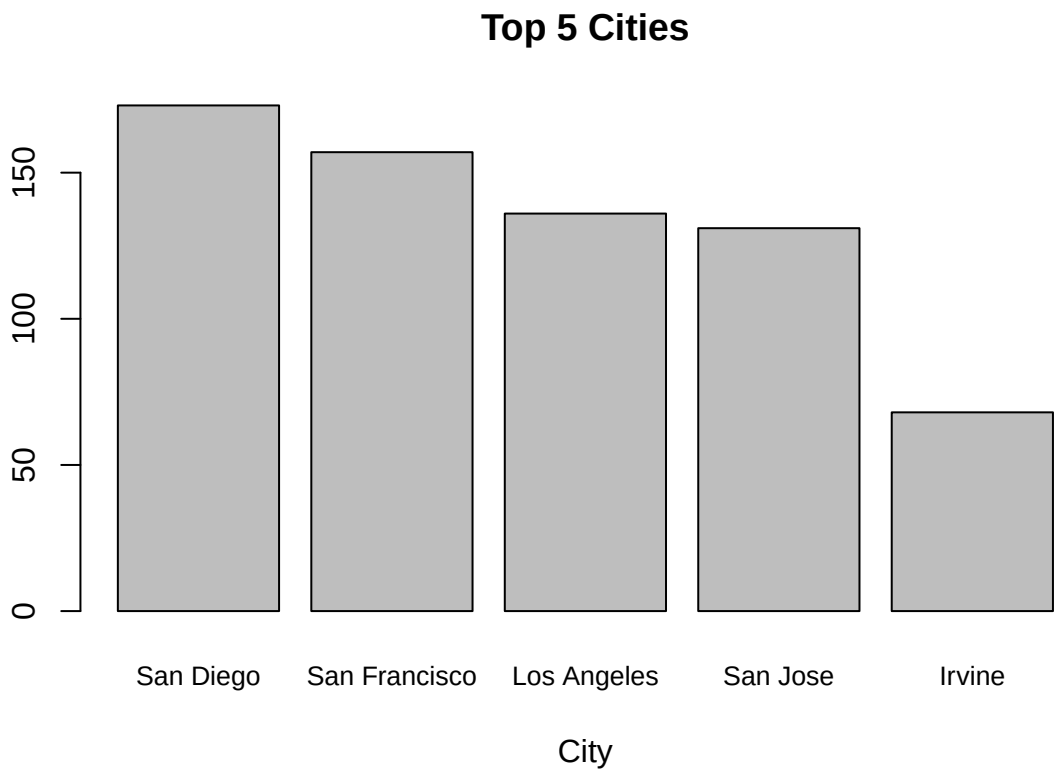

```
# top degree fields
topFields = reqSkills_table[c("business", "science", "marketing", "engineering", "healthcare")]
names(topFields) = c("Business", "Data/Comp. Sc.", "Marketing", "Engineering", "Healthcare")
barplot(topFields, main = "Top Degree Fields", xlab = "Fields", cex.names = 0.7)
```



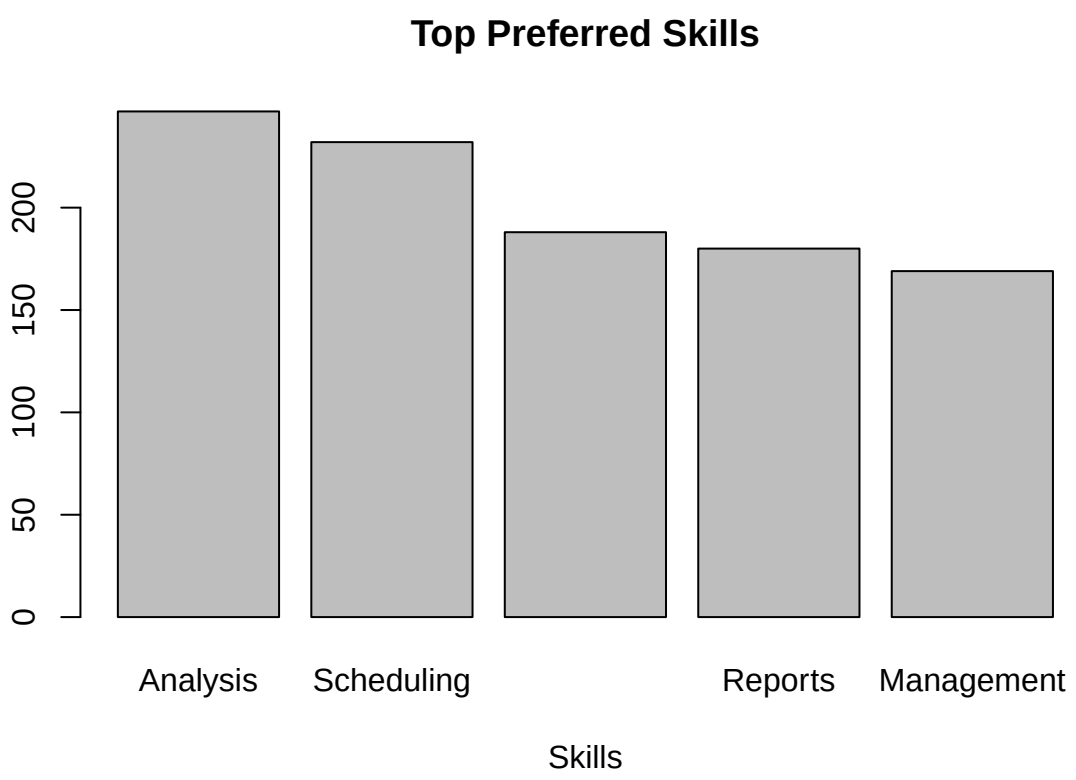

```
# employment types
barplot(table(employmentType), main = "Employment Type", xlab = "Employment Type", cex.names = 0.7)
```




```
# location
barplot(sort(table(city), decreasing = TRUE)[1:5], main = "Top 5 Cities", xlab = "City", cex.names = 0.8
```




```r
# top preferred skills
prefSkills = unlist(strsplit(careerBuilder$preferredSkills, "@@"))
prefSkills_table = sort(table(prefSkills), decreasing = TRUE)
topPrefSkills = prefSkills_table[1:5]
barplot(topPrefSkills, main = "Top Preferred Skills", xlab = "Skills")
```



From the plots above, here are some (interesting) findings:

- Majority of the employment types are once again full-time and contract work.
- As expected, San Diego, San Francisco, Los Angeles, and San Jose are the top cities with the highest amount of job postings, seeing as they are the major cities in California.
- Analysis, Scheduling, Reports, and Management are some highly sought after preferred skills, which is significantly different from that of CyberCoders, but this could be due to the fact that majority of the search results are for Data Analyst related positions.
- A huge amount of job postings did not specify a required degree, but for those that did, it once again mainly just requires at least a Bachelor's degree.
- Some of the main degree fields for these positions are Business, Data/Comp. Science, Marketing, Engineering, and Healthcare.

Monster

Web-Scraping

This website is slightly more problematic than the previous two. This is due to the "load more" or "next page" issue when browsing through results. Unlike Indeed and Career Builder, where the "next page" button brings me to the next page of the search results, the "load more" button was designed to extend the amount of search results, ie. the web page does not reload or redirect to a different page, but merely just adds more content. The URL, however, does change from

https://www.monster.com/jobs/search/?q=Data-Scientist&where=california&intcid=skr_navigation_nhpso_searchMain

to

https://www.monster.com/jobs/search/?q=Data-Scientist&where=california&intcid=skr_navigation_nhpso_searchMain&stpage=1&page=3.

Notice that the latter URL contains additional queries "stpage" and "page". This was a good sign, but it only works up until "page=10", even if there exists more than 10 pages of search results.

According to Prof. Lang's explanation in his Office Hours, there is supposedly a different way of excessing the search results. This is found via the Network section in the Developer Tools settings. There, we can find a URL with the form

https://www.monster.com/jobs/search/pagination/?q=Data-Scientist&where=california&intcid=skr_navigation_nhpso_searchMain&isDynamicPage=true&isMKPagination=true&page=4&total=78.

The main differences between the URL here and the ones before are the addition of the "isDynamicPage", "isMKPagination" and "total" queries. Once I have added these queries, I will then be directed to a webpage written in JSON. Each of these JSON pages contains 29 objects, of which 26 are job postings and 3 are ads.

With this in mind, scraping data becomes significantly easier. A lot of the data from a job posting can now be extracted directly without worrying about xpaths, eg. `jobTitle = jsonResult[[1]]$title`. The only HTML pages that we have to parse are the individual job description pages.

The searches returned the following:

Data Scientist - 821 results

Data Analyst - 417 results

Statistician - 33 results

Data Cleaning/Analysis

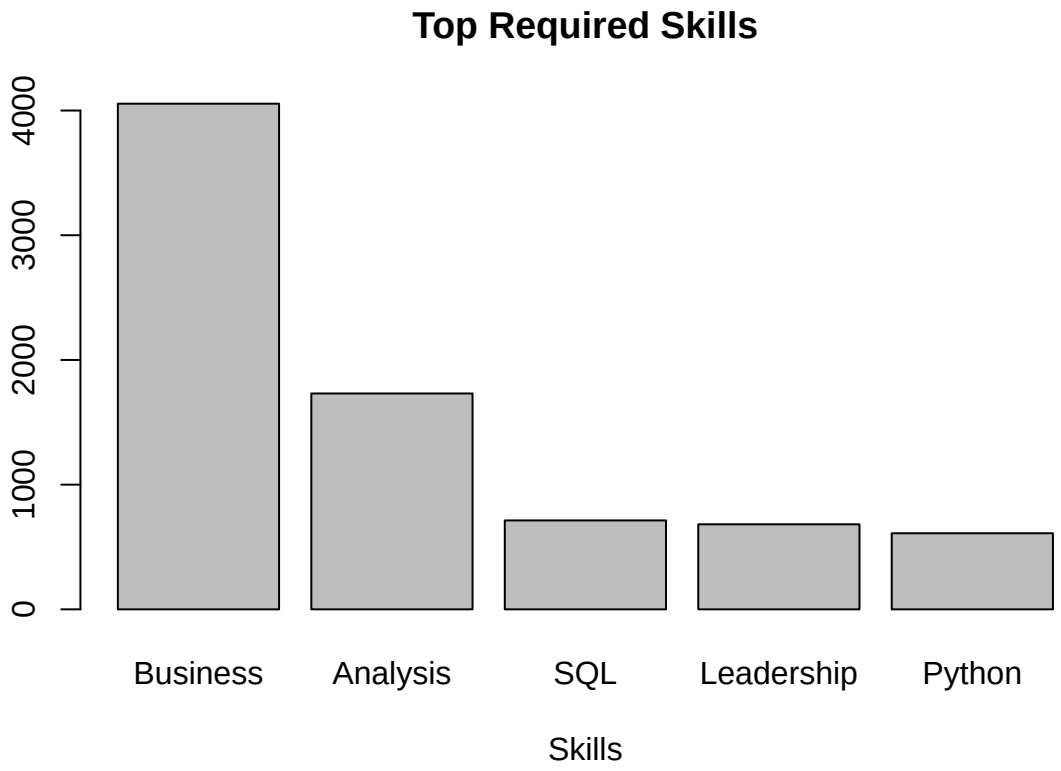
```
load("monster.RData")

# minimum education level
minEducationLevel = factor(sapply(monster$description, function(desc){
  if (grepl("high school diploma", desc, ignore.case = TRUE) == TRUE) {
    return ("High School")
  } else if (grepl("(\\<A[.]?A\\>|\\<A[.]?S\\>|Associate.?s degree)", desc) == TRUE){
    return ("Associate's")
  } else if (grepl("(\\<B[.]?A\\>|\\<B[.]?S\\>|Bachelor[']?s?)", desc) == TRUE){
    return ("Bachelor's")
  } else if (grepl("(M[.]?S\\>|Master.?s|Advanced [Dd]egree)", desc) == TRUE) {
    return ("Master's")
  } else if (grepl("(\\<Ph[.]?[Dd]|[Dd]octoral[ [Dd]egree]?)", desc) == TRUE) {
    return ("Ph.D")
  } else {
    return ("None")
  }
}), levels = c("None", "High School", "Associate's", "Bachelor's", "Master's", "Ph.D"))

# location
city = trimws(gsub("([[:alpha:] ]+),([[:space:][:alpha:]]+).*", "\\1", monster$location))
state = trimws(gsub("([[:alpha:] ]+),([[:space:][:alpha:]]+).*", "\\2", monster$location))
```

```r
# employment type
employmentTypeRegex = "(full[- ]?time|part[- ]time|hourly|contract|short[- ]term|month)"
getEmploymentType = function(description){
  if (grepl(employmentTypeRegex, description, ignore.case = TRUE) == TRUE){
    m = regexpr(employmentTypeRegex, description, ignore.case = TRUE)
    result = regmatches(description, m)
    employmentType = switch(tolower(result),
                            "full-time" = "Full-time",
                            "full time" = "Full-time",
                            "part-time" = "Part-time",
                            "part time" = "Part-time",
                            "hourly" = "Hourly",
                            "contract" = "Contract",
                            "short-term" = "Short-term",
                            "short term" = "Short-term",
                            "month" = "Contract",
                            result)
    return (employmentType)
  } else{
    return (NA)
  }
}
employmentType = unlist(lapply(monster$description, getEmploymentType))

# forming cleaned data frame
monster_df = data.frame(Position = monster$title,
                        Company = monster$company,
                        City = city,
                        State = state,
                        EmploymentType = employmentType,
                        MinEducationLevel = minEducationLevel,
                        Link = monster$link)
head(monster_df)
```

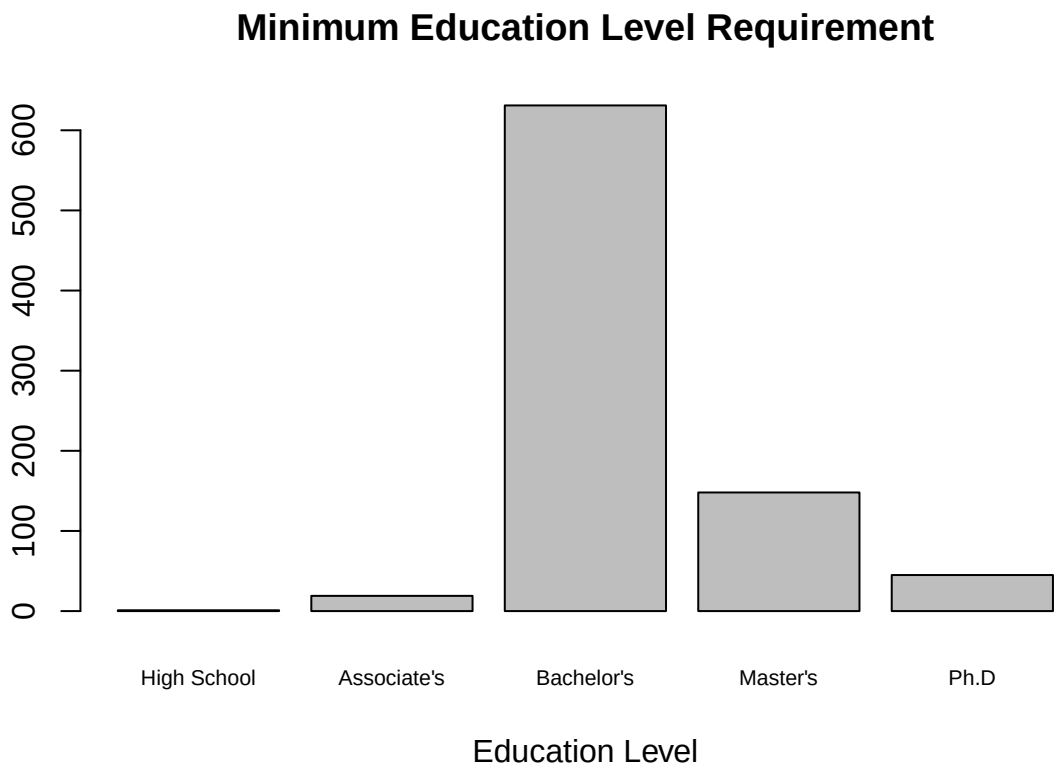
```
##                                     Position        Company
## 1                              Data Scientist APR Consulting
## 2 Senior Data Scientist - Geneticist (REMOTE)    CyberCoders
## 3                        Senior Data Scientist           Loom
## 4        Data Scientist needed at Media Giant       Ingenium
## 5               Senior Data Scientist - Homes       Opendoor
## 6                              Data Scientist           Visa
##                  City State EmploymentType MinEducationLevel
## 1             Oakland    CA       Contract          Master's
## 2 South San Francisco    CA           <NA>              Ph.D
## 3       San Francisco    CA           <NA>              None
## 4       San Francisco    CA           <NA>        Bachelor's
## 5       San Francisco    CA           <NA>              None
## 6         Foster City    CA           <NA>        Bachelor's
##
## 1                      https://job-openings.monster.com/data-scientist-oakland-ca-us-apr-consulting,
## 2                https://job-openings.monster.com/senior-data-scientist-geneticist-remote-south-san-
## 3                    https://job-openings.monster.com/senior-data-scientist-san-francisco-ca-us-loom,
## 4 https://job-openings.monster.com/data-scientist-needed-at-media-giant-san-francisco-ca-us-ingenium,
```

```
## 5          https://job-openings.monster.com/senior-data-scientist-homes-san-francisco-ca-us-opendoor/
## 6                              https://job-openings.monster.com/data-scientist-foster-city-ca-us-visa/
```

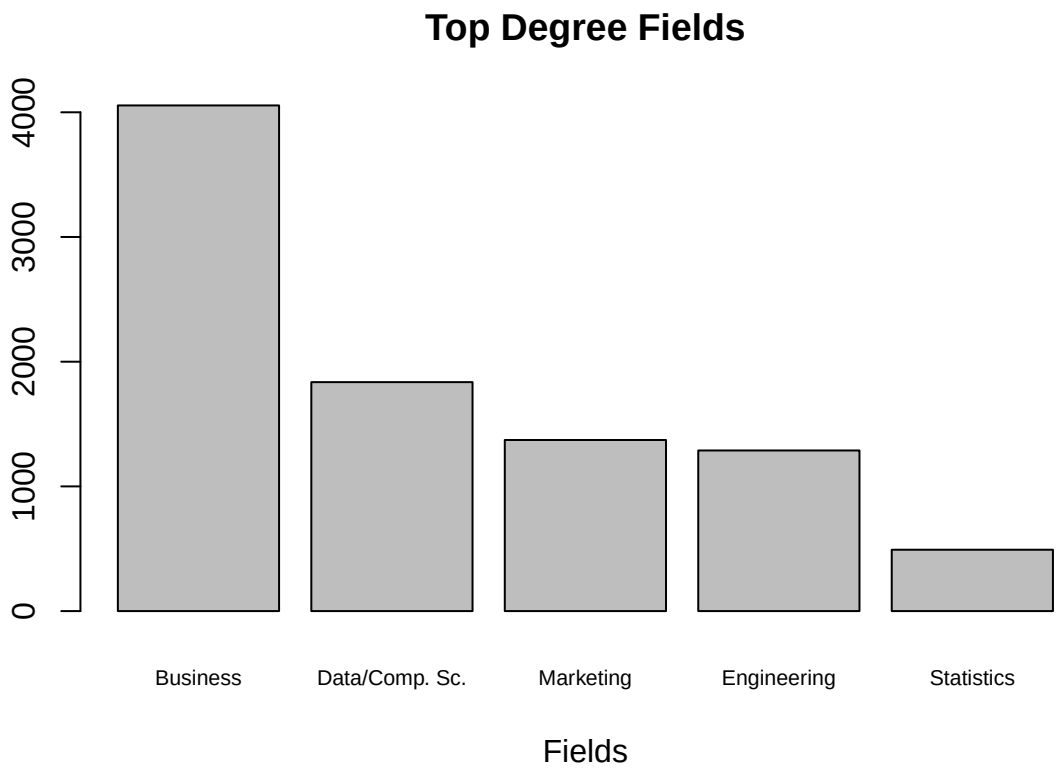
```r
# required skills
reqSkills = tolower(unlist(strsplit(monster$description, "@@")))
reqSkills_keywords = unlist(lapply(reqSkills, function(x) {
  m = gregexpr("([[:alnum:]]+[[:punct:]][[:alnum:]]+|[[:alnum:]]+)", x)
  results = regmatches(x, m)
  return (results)
}))
reqSkills_keywords = reqSkills_keywords[!(reqSkills_keywords %in% stopWords)]
reqSkills_table = sort(table(reqSkills_keywords), decreasing = TRUE)
topReqSkills = reqSkills_table[c("business", "analysis", "sql", "leadership", "python")]
names(topReqSkills) = c("Business", "Analysis", "SQL", "Leadership", "Python")
barplot(topReqSkills, main = "Top Required Skills", xlab = "Skills")
```



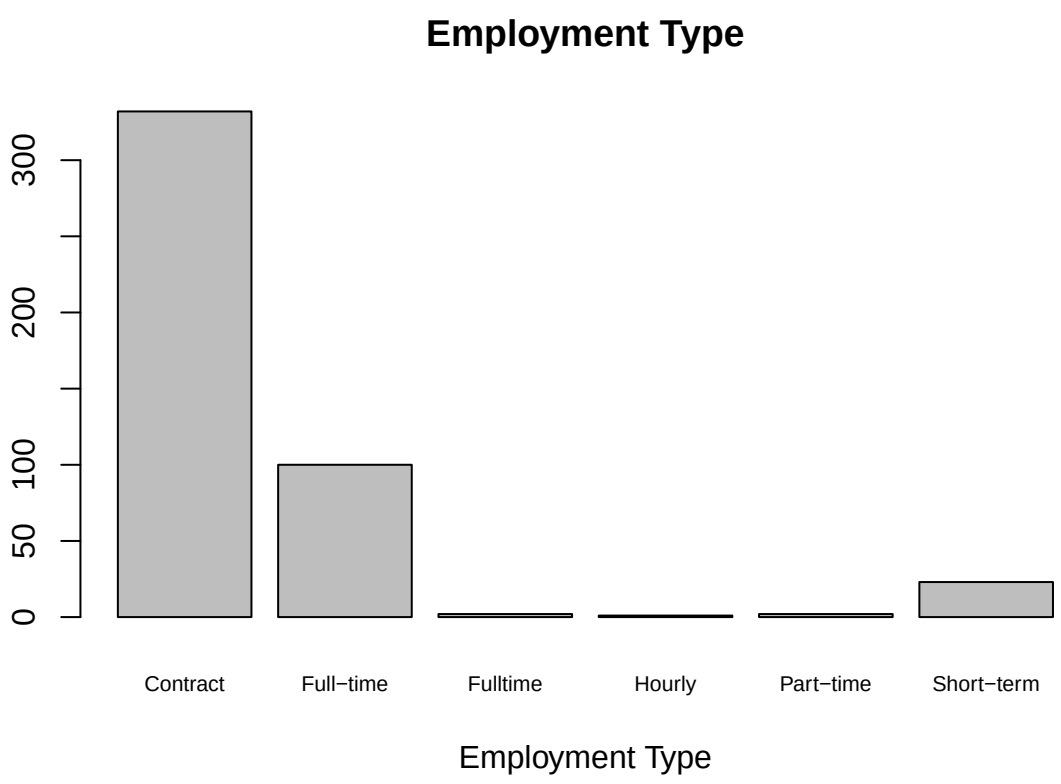

```r
# minimum required education level
minEducationLevel_table = table(minEducationLevel)
barplot(minEducationLevel_table[2:6], main = "Minimum Education Level Requirement", xlab = "Education L
```



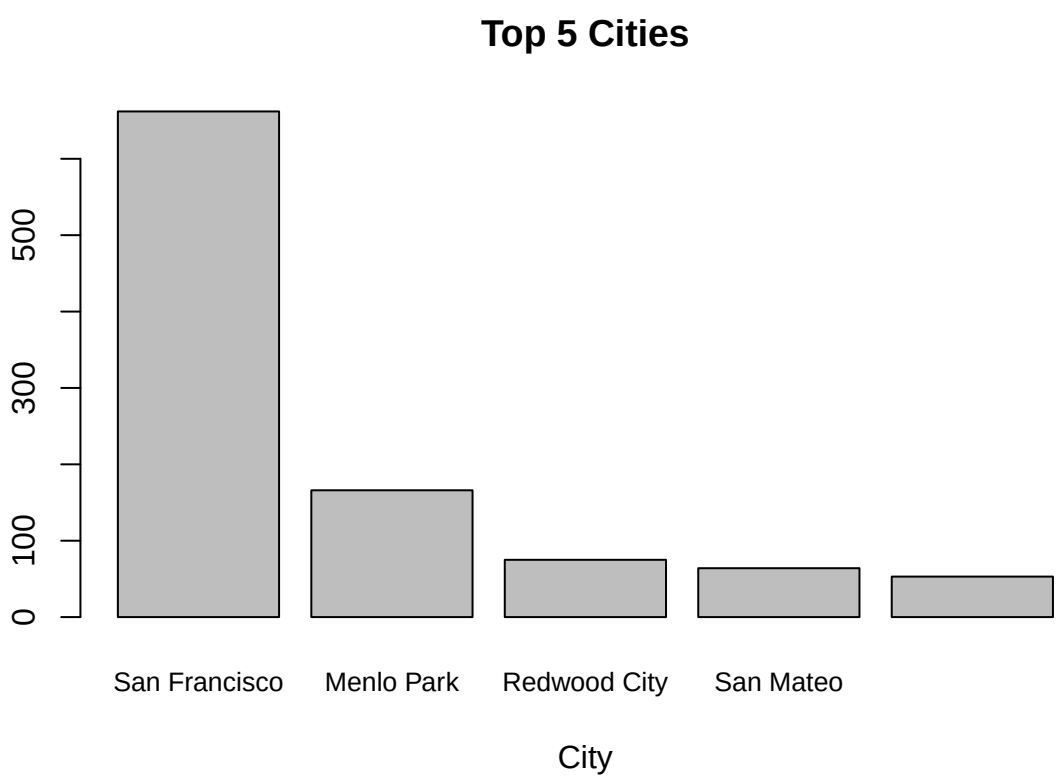

```r
# top degree fields
topFields = reqSkills_table[c("business", "science", "marketing", "engineering", "statistics")]
names(topFields) = c("Business", "Data/Comp. Sc.", "Marketing", "Engineering", "Statistics")
barplot(topFields, main = "Top Degree Fields", xlab = "Fields", cex.names = 0.7)
```




```
# employment types
barplot(table(employmentType), main = "Employment Type", xlab = "Employment Type", cex.names = 0.7)
```




```
# location
barplot(sort(table(city), decreasing = TRUE)[1:5], main = "Top 5 Cities", xlab = "City", cex.names = 0.8
```



From the plots above, here are some findings:

- In contrast to the previous websites, the majority of the employment type here are contract based.
- A huge amount of jobs requires at least a Bachelor's degree. This is also a huge contrast to previous findings where many job postings did not specify the minimum required education level.
- SQL, Python, and leadership skills are some of the most sought after skills here. While business sits up there with numbers significantly larger than the rest of the skills, it is hard to tell if the word "business" here comes in the context of skills, or degree field, or something else.

Indeed

Web-Scraping

Scraping this website was significantly tougher, not because parsing the HTML was tough, but mainly I kept encountering the reCAPTCHA issue when scraping this website. This always happens after I scraped around 100 job postings, and as a result I get temporarily banned from the website. I tried various workarounds, including adding `Sys.sleep(n)`, changing the cookie configuration, and simply just waiting it out until I get unbanned from the site. After numerous tries and failures that took up more than 2 days of my time, I decided to limit the location of my search from "California" to "Sacramento" in order to complete this assignment.

Moving on, an issue that arised with this website is the different attribute names stated on the webpage and when I parse it using R. For example, all the search results are located inside the `div` nodes with `class = "jobsearch-SerpJobCard unifiedRow row result clickcard"]`. However, the `class` value for those `div` nodes when parsing through R was instead `jobsearch-SerpJobCard unifiedRow row result`.

I found this by looking for its parent or an ancestor where the attribute names on the webpage and when parsing through R is the same, ie. using the code `xpathSApply(doc, "//td[@id='resultsCol']//*", xmlGetAttr, "class")`. Once this issue is solved, the rest of the web-scraping procedure was relatively standard compared to the previous websites.

The searches returned the following:

Data Scientist - 44 results

Data Analyst - 108 results

Statistician - 6 results

Data Cleaning/Analysis

The data cleaning steps are similar to the previous sections.

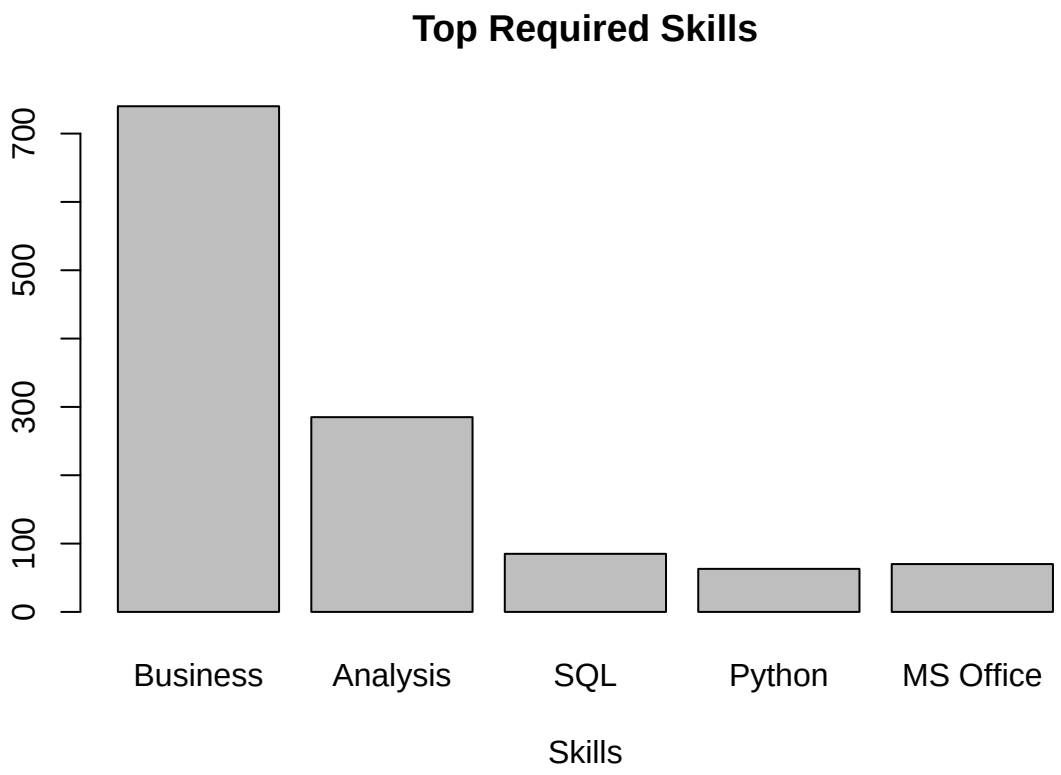
```
load("indeed.RData")

# minimum education level
minEducationLevel = factor(sapply(indeed$description, function(desc){
  if (grepl("high school diploma", desc, ignore.case = TRUE) == TRUE) {
    return ("High School")
  } else if (grepl("(\\<A[.]?A\\>|\\<A[.]?S\\>|Associate.?s degree)", desc) == TRUE){
    return ("Associate's")
  } else if (grepl("(\\<B[.]?A\\>|\\<B[.]?S\\>|Bachelor[']?s?)", desc) == TRUE){
    return ("Bachelor's")
  } else if (grepl("(M[.]?S\\>|Master.?s|Advanced [Dd]egree)", desc) == TRUE) {
    return ("Master's")
  } else if (grepl("(\\<Ph[.]?[Dd]|[Dd]octoral[ [Dd]egree]?)", desc) == TRUE) {
    return ("Ph.D")
  } else {
    return ("None")
  }
}), levels = c("None", "High School", "Associate's", "Bachelor's", "Master's", "Ph.D"))

# location
city = trimws(gsub("(^[a-zA-Z ]+$|([[:alpha:] ]+),( ?[[:alpha:]]+).*)", "\\2", indeed$location))
city = ifelse(city == "", NA, city) # change "" to NA
state = trimws(gsub("(^[a-zA-Z ]+$|([[:alpha:] ]+),( ?[[:alpha:]]+).*)", "\\3", indeed$location))
state = ifelse(state == "", NA, state) # change "" to NA

# employment type
employmentTypeRegex = "(full[- ]?time|part[- ]time|hourly|contract|short[- ]term|month)"
getEmploymentType = function(description){
  if (grepl(employmentTypeRegex, description, ignore.case = TRUE) == TRUE){
    m = regexpr(employmentTypeRegex, description, ignore.case = TRUE)
    result = regmatches(description, m)
    employmentType = switch(tolower(result),
                            "full-time" = "Full-time",
                            "full time" = "Full-time",
                            "part-time" = "Part-time",
                            "part time" = "Part-time",
                            "hourly" = "Hourly",
                            "contract" = "Contract",
```

```
                            "short-term" = "Short-term",
                            "short term" = "Short-term",
                            "month" = "Contract",
                            result)
    return (employmentType)
  } else{
    return (NA)
  }
}
employmentType = unlist(lapply(indeed$description, getEmploymentType))

# forming cleaned data frame
indeed_df = data.frame(Position = indeed$title,
                       Company = indeed$company,
                       City = city,
                       State = state,
                       EmploymentType = employmentType,
                       MinEducationLevel = minEducationLevel,
                       Link = indeed$link)
head(indeed_df)
```

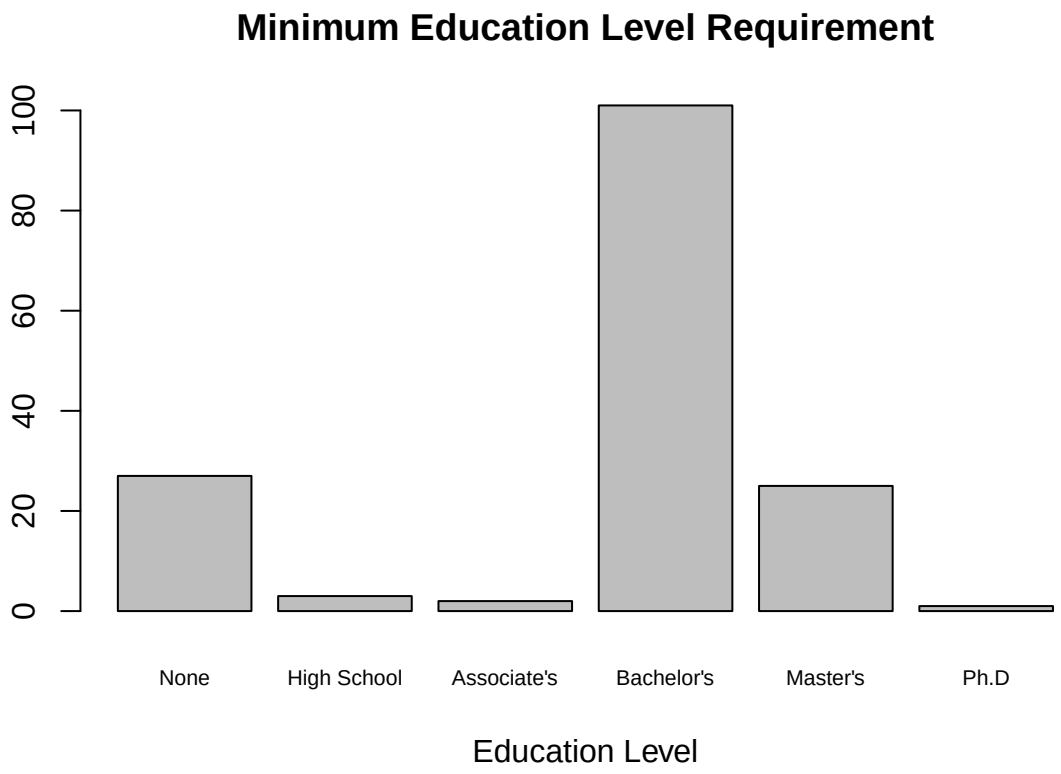
```
##                                       Position
## 1                  Data Scientist Master - Remote
## 2                                   Statistician
## 3                              Tableau Developer
## 4                  Research Data Specialist I & II
## 5 Senior Statistician - Innovative Biopharma - USA
## 6          Lead Statistician/Programmer Healthcare
##                                Company           City State EmploymentType
## 1                             Schneider           <NA>  <NA>           <NA>
## 2   National Older Worker Career Center     Sacramento    CA           <NA>
## 3 John Burns Real Estate Consulting LLC         Folsom    CA      Full-time
## 4      California Department of Justice     Sacramento    CA      Full-time
## 5                         Warman O'Brien           <NA>  <NA>      Full-time
## 6           Harmony Technology Services Rancho Cordova    CA           <NA>
##   MinEducationLevel
## 1          Master's
## 2          Master's
## 3        Bachelor's
## 4          Master's
## 5              None
## 6          Master's
##
## 1                                                                https://www.indeed.com/pagead/clk?
## 2
## 3
## 4
## 5 https://www.indeed.com/pagead/clk?mo=r&ad=-6NYlbfkN0Bw-rRrlDr6nrfscfj_vVoPM6Iu_8zo-cqs0II8k4Z4L09_
## 6
```

```
# required skills
reqSkills = tolower(unlist(strsplit(indeed$description, "@@")))
reqSkills_keywords = unlist(lapply(reqSkills, function(x) {
```

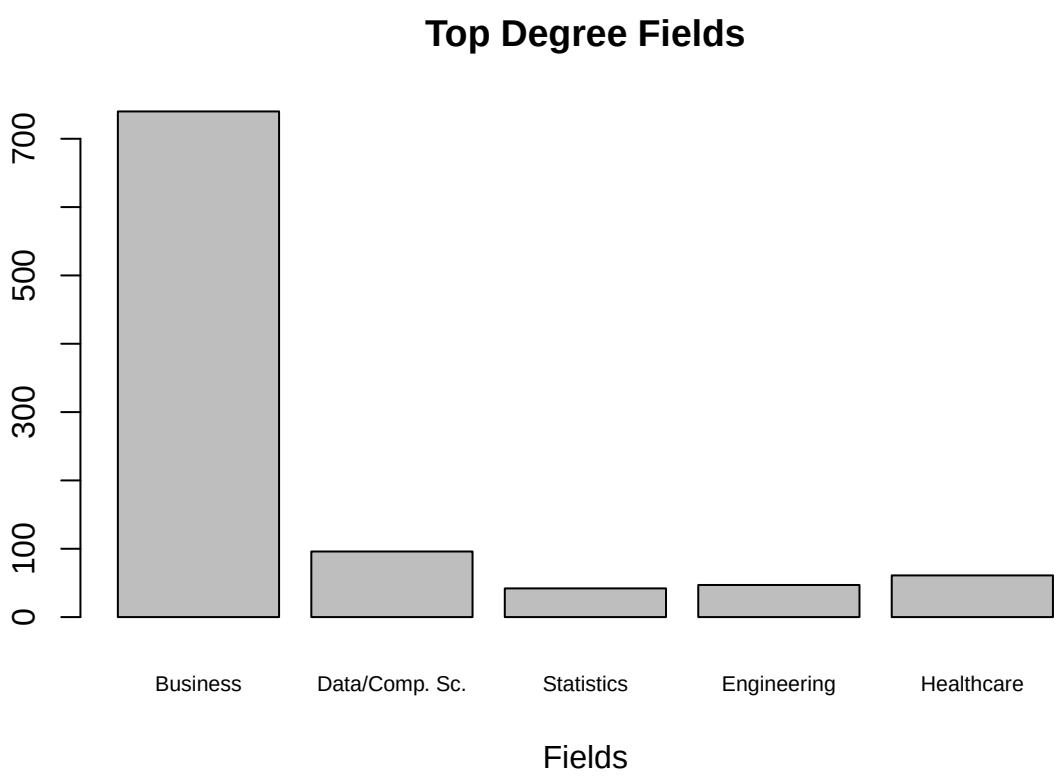
```
  m = gregexpr("([[:alnum:]]+[[:punct:]][[:alnum:]]+|[[:alnum:]]+)", x)
  results = regmatches(x, m)
  return (results)
}))
reqSkills_keywords = reqSkills_keywords[!(reqSkills_keywords %in% stopWords)]
reqSkills_table = sort(table(reqSkills_keywords), decreasing = TRUE)
topReqSkills = reqSkills_table[c("business", "analysis", "sql", "python", "office")]
names(topReqSkills) = c("Business", "Analysis", "SQL", "Python", "MS Office")
barplot(topReqSkills, main = "Top Required Skills", xlab = "Skills")
```



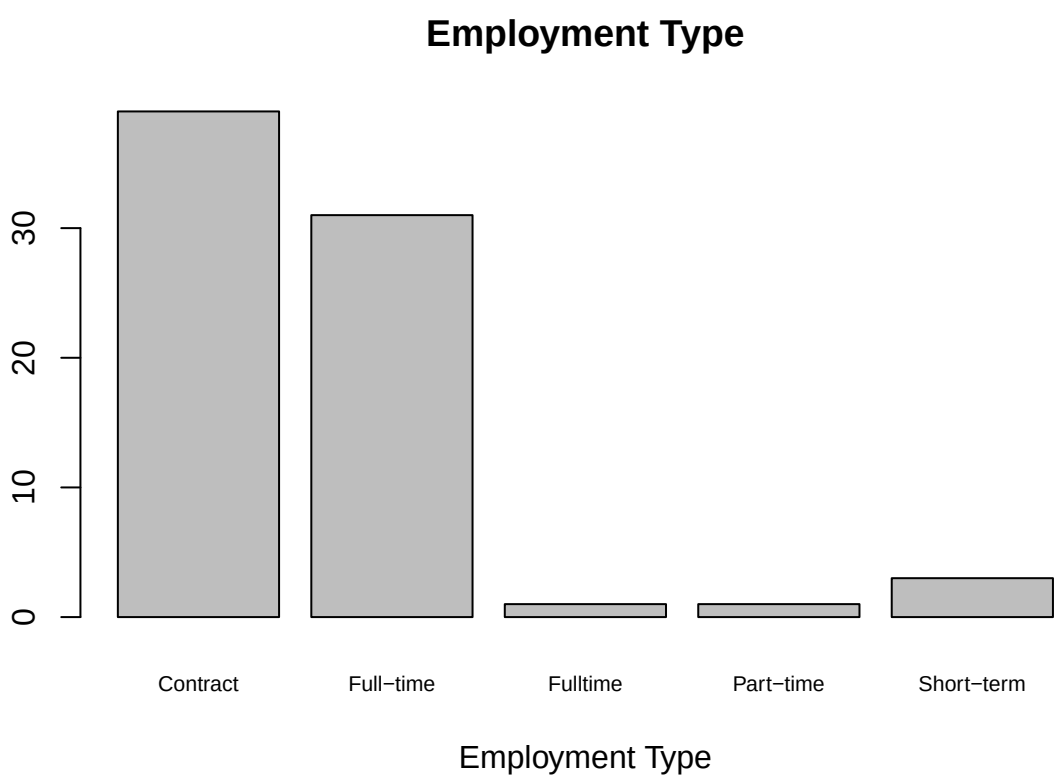

```
# minimum required education level
minEducationLevel_table = table(minEducationLevel)
barplot(minEducationLevel_table, main = "Minimum Education Level Requirement", xlab = "Education Level"
```



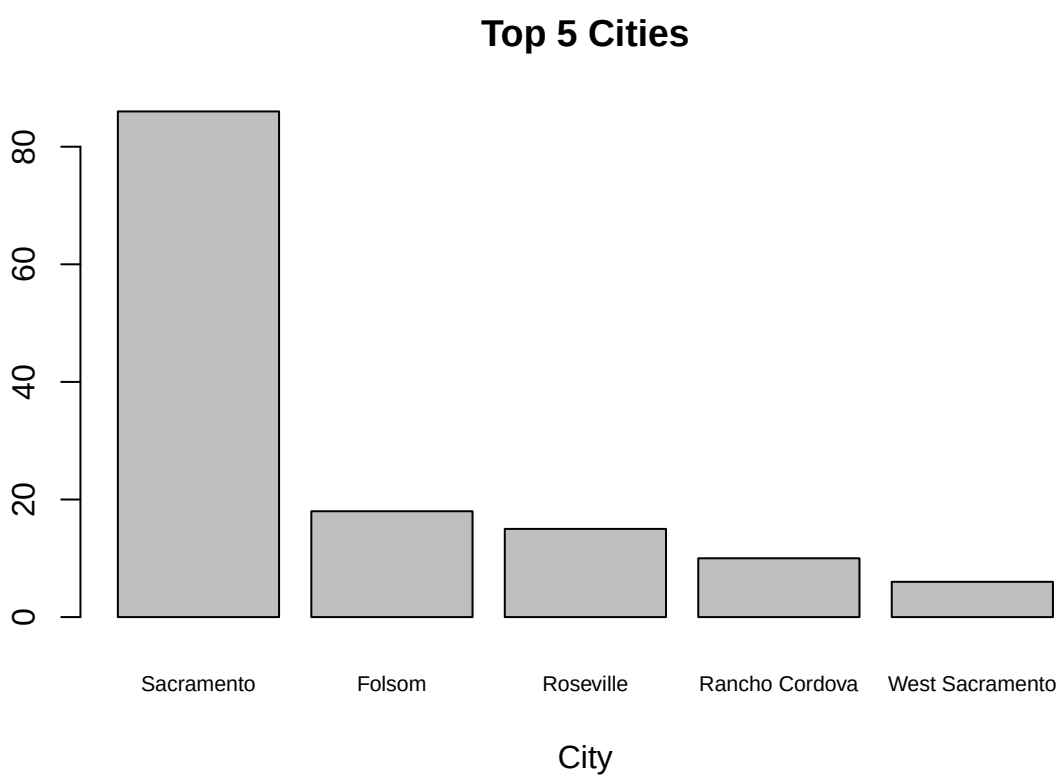

```
# top degree fields
topFields = reqSkills_table[c("business", "science", "statistics", "engineering", "healthcare")]
names(topFields) = c("Business", "Data/Comp. Sc.", "Statistics", "Engineering", "Healthcare")
barplot(topFields, main = "Top Degree Fields", xlab = "Fields", cex.names = 0.7)
```




```
# employment types
barplot(table(employmentType), main = "Employment Type", xlab = "Employment Type", cex.names = 0.7)
```




```
# location
barplot(sort(table(city), decreasing = TRUE)[1:5], main = "Top 5 Cities", xlab = "City",
        cex.names = 0.7)
```



From the plots above, here are some findings:

- Just like the job postings from Monster, the majority of the employment type here are contract based.
- A huge amount of jobs requires at least a Bachelor's degree. This is also a huge contrast to previous findings where many job postings did not specify the minimum required education level.
- SQL, Python, and MS Office are some of the most sought after skills here. While business sits up there with numbers significantly larger than the rest of the skills, it is hard to tell if the word "business" here comes in the context of skills, or degree field, or something else.