DARL Notebook: Greenwich HR dataset in R

Data Analytics Research Lab (DARL) (Fall 2017)

Abhishek Choudhary

Contents

The file is located at "~/data/DARL__Fall2017/GREENWICH/abhi/Notebook__abhi.Rmd".

Overview

This is an R Markdown Notebook submitted by one member of the team ThunderGreen. Our team is working on analyzing the data provided by our client Greenwich HR, a job market data provider company in healthcare.

Introduction

In this report, I will first describe the data provided by the client. I will then list the quetsions our team intend to answer via this dataset, i.e. what information can be obtained regarding employment statistics. Then I will provide a detailed overview of the exploration we have done so far along with the code whereever necessary.

What is our dataset?

The original dataset consists of 4 excel files with total size in the order of a few 100 MBs. I will describe below the contents and purpose of each file.

1. `master.xls` This is the main file which contains all the data in one place. It cosnits of 10 sheets in continuation. There are 16 columns which are respectively Job Id (a large integer), posting date (format yyyy-mm-dd), filling date (format yyyy-mm-dd), time to fill (in number of days, format integer), company name (format string), verticle (or sector, only consisting of value 'healthcare'), job location (format string in the form of city, state), salary (format integer), city (format string), state (format 2 letter string e.g. NY), zip (format 5 digit integer), full name of state (format string), county (format string), latitude (format float with 16 significant digits), longitude (format float with 16 significant digits), and region (format string describing county state and type of the area).

A close look at the first few rows of this file:

```
library(readxl)
master1 <- read.csv("/data/DARL_Fall2017/GREENWICH/master_csvs/master_sheet1.csv")
head(master1, 3)
```

```
##         job_id  post_date  fill_date time_to_fill
## 1 9.568341e+31 2016-11-13 2016-12-02           19
## 2 9.821270e+31 2016-09-07 2016-09-17           10
## 3 4.572041e+31 2017-01-24 2017-02-25           32
##                        company   vertical     location salary     city
## 1 Marsh, Berry & Company, Inc. Healthcare Lakeland, FL  81964 Lakeland
## 2                  FedEx GENCO Healthcare  Coppell, TX  26208  Coppell
## 3    Santa Rosa Medical Center Healthcare   Milton, FL  59411   Milton
##   state   zip state_long     county latitude longitude
## 1    FL 33801    Florida       Polk  28.0381  -81.9392
## 2    TX 75019      Texas     Dallas  32.9673  -96.9805
## 3    FL 32570    Florida Santa Rosa  30.6604  -87.0473
##                           region_state
## 1                     Lakeland, FL MSA
## 2 Dallas-Fort Worth-Arlington, TX MSA
## 3  Pensacola-Ferry Pass-Brent, FL MSA
```

2. `filter.xls` This file consists of 37 sheets in continuation and it lists all the jobs with their corresponding skill set. There are three columns: source (a five digid alphanumeric code format), job id (format large integer), and skill (format string) respectively. A close look at the first few rows of this file:

```r
filter1 <- read_excel(paste("~/data/DARL_Fall2017/GREENWICH/all_sheets/",
                            "filter_sheets/a0004_stg_3_skill.xlsx",sep=""), skip=0)
head(filter1,3)
```

```
## # A tibble: 3 × 3
##   source                           job_id      skill
##    <chr>                            <chr>      <chr>
## 1  A0004  7274202424508126431456148048430   Clerical
## 2  A0004 78925993620992907371847482294504    Finance
## 3  A0004 81417224156492343399450023024800 Acute Care
```

3. `role.xls` This file consists of 13 sheets in continuation and it lists all the jobs with their corresponding roles. There are two columns: job id (format large integer), and role (format string). A close look at the first few rows of this file:

```r
role1 <- read_excel(paste("~/data/DARL_Fall2017/GREENWICH/all_sheets/",
                    "role_sheets/a0004_stg_3_role.xlsx",sep=""), skip=0)
head(role1,3)
```

```
## # A tibble: 3 × 2
##                              job_id  role
##                               <chr> <chr>
## 1  6476319890481755095128327611895    RN
## 2  2293247481198657180190634649939    RN
## 3 59902270325782628318450118187725    RN
```

4. `role_hierarchy_hc.xls` This file lists the details of all the roles in a single sheet using 4 columns, which are all in format string and respectively are: role, role_type, role_sub_family and role_family. A close look at the first few rows of this file:

```r
library(readxl)
role_hierarchy_hc <- read_excel("~/data/DARL_Fall2017/GREENWICH/role_hierarchy_hc.xls", skip=0)
head(role_hierarchy_hc,3)
```

```
## # A tibble: 3 × 4
##                           role                  role_type1 job_sub_family
##                          <chr>                       <chr>          <chr>
## 1               Acupuncturist               Acupuncturist  Allied Health
## 2              Admixture Tech              Admixture Tech  Allied Health
## 3 Alcohol Drug Counselor/ACLS Alcohol Drug Counselor/ACLS  Allied Health
## # ... with 1 more variables: job_family <chr>
```

What are our goals?

Using this dataset which lists the details of at least a few hundered thousand jobs in healthcare sector in four states in the United States, we plan to extract some useful information which could be utilized by potential employers, business leaders, labor market analysts, and populace at lerge. The intelligence we seek to gather can be summarized as following key questions which can be classified in 4 different categories:

1. Seek Some Patterns

We can find some basic patterns from this data regarding job market in the four states with the questions like:

- The general distribution of salaries
- #of jobs with each skill set
- distribution of the times to fill positions
- average pay scale across different regions

2. Trends Over Time

We can find how the job market has changed over the year by analyzing things like:

- # of jobs posted and filled v/s time (total and role wise)

3. Trends Over Region

We can try to understand how the location affects the job market by answering the questions like:

- wages in a region for different jobs
- # of jobs posted and filled in each region

4. What's Hot?

We can try to predict the future to see what's attractive by looking at the statistics and gathering information such as:

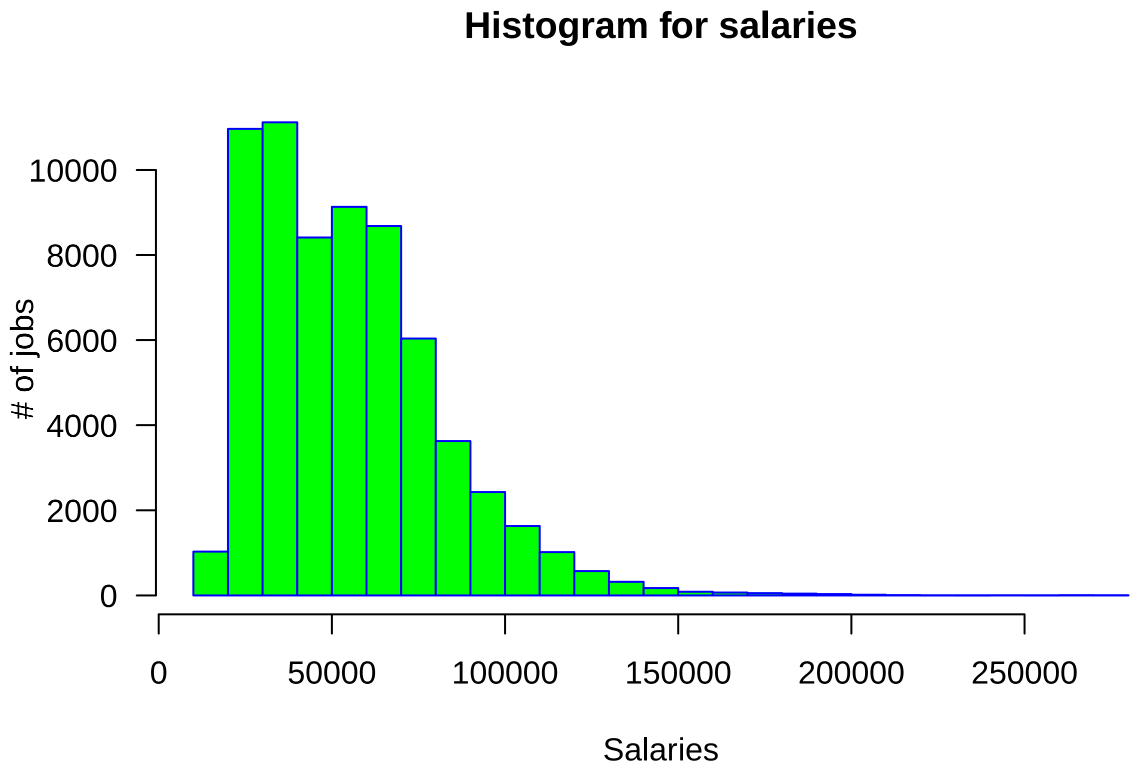
- # of jobs posted in different roles
- time taken to fill different positions (good locations to start a business and hiring people etc.?)
- region wise plot of difference between wages and cost of living (what's good job to find in a certain location?)
- graphs of skills with average salaries (what's a good salary for a certain job?)
- # of jobs with demographics, and type of location (urban/rural etc.) (What's driving pay differences i.e. location, profile etc.?)
- what's driving pay, length and demand

This list is dynamic and we will be updating it as more questions fit to be answered come up in oure analysis.

Let’s start exploring

To start with, I have plotted the histogram of salaries from one of the sheets in master file.

```
# Since one of my questions is relating to job and salaries,
#I pick these two keys and compose a new vector
id_sar <- master1[,c(1,8)]
#hist(id_sar$salary)

hist(id_sar$salary,
     main="Histogram for salaries",
     xlab="Salaries",
     ylab="# of jobs",
     border="blue",
     col="green",
     las=1,
     breaks=20)
```



Big data yet not “Big Data”

Our dataset doesn’t qualify as big data but still given the number of jobs and other details in our dataset, it comprises of at least two >150 MB excel files which can only be read when they are broken up in separate sheets. But this doesn’t wok for the analysis purpose where we need the full list at one place. So the first step is to combine all the sheets corresponding to each file and get the respective table. The code below illustrates this.

```r
readMasterData <- function() {
  master <- "~/data/DARL_Fall2017/GREENWICH/all_sheets/master_sheets/a0004_stg_table_2_slim.xlsx"
  filePaths <- c(
    "~/data/DARL_Fall2017/GREENWICH/all_sheets/master_sheets/a0004_stg_table_2_slim(2).xlsx",
    ...
    ...
    "~/data/DARL_Fall2017/GREENWICH/all_sheets/master_sheets/a0004_stg_table_2_slim(10).xlsx")
  return(readAnyData(master,filePaths))
}

#Automatically reads the role data and returns a single array.
readRoleData <- function() {
  role <- "~/data/DARL_Fall2017/GREENWICH/all_sheets/role_sheets/a0004_stg_3_role.xlsx"
  filePaths <- c(
    "~/data/DARL_Fall2017/GREENWICH/all_sheets/role_sheets/a0004_stg_3_role(2).xlsx",
    ...
    ...
    "~/data/DARL_Fall2017/GREENWICH/all_sheets/role_sheets/a0004_stg_3_role(13).xlsx")
  return(readAnyData(role,filePaths))
}

#Automatically reads the filter data and returns a single array.
readFilterData <- function() {
  filter <- "~/data/DARL_Fall2017/GREENWICH/all_sheets/filter_sheets/a0004_stg_3_skill.xlsx"
  filePaths <- c(
    "~/data/DARL_Fall2017/GREENWICH/all_sheets/filter_sheets/a0004_stg_3_skill(2).xlsx",
    ...
    ...
    "~/data/DARL_Fall2017/GREENWICH/all_sheets/filter_sheets/a0004_stg_3_skill(37).xlsx")
  return(readAnyData(filter,filePaths))
}

#This function takes in a string filePath for the file with the headers. It then takes a list
#of the the file paths for all other files without headers.
readAnyData <- function(firstFilePath, otherFilePaths) {
  first <- read_excel(firstFilePath, skip=0)
  other_files <- lapply(otherFilePaths, read_excel, skip=1)

  #fix the column names so that the lists can be rbinded
  for (i in 1:length(other_files)) {
    colnames(other_files[[i]]) <- colnames(first)
  }

  #bind all the lists together
  all_data <- Reduce(rbind,other_files,first)
  return(all_data)
}

#MAIN
all_data <- readMasterData()
role_data <- readRoleData()
filter_data <- readFilterData()
```

Now tables all_data, role_data and filter_data have all the rows (in the order of hundreds of thousands) from master file, role file and filter file respectively. The fourth file in the provided dataset role_hierarchy_hc.xlsx is still untouched but that will be included in our analysis at a later point.

Where do I belong?

One important classfier for the job database (and this is true in general to some extent) is the state in which jobs are located. Since we have data from four states (VA,TX,FL,IL), it's only natural to extract the jobs from all_data pertaining to an individual state. extractState function shown below does that.

```
#this function extract a state
extractState <- function(all_data, m_state) {
  return(all_data[all_data$state == m_state,])
}

#MAIN
va_data <- extractState(all_data, "VA")
il_data <- extractState(all_data, "IL")
fl_data <- extractState(all_data, "FL")
tx_data <- extractState(all_data, "TX")
```

Tie it together

The tables that we have in the dataset represent differnt information. The master file is the primary source which lists the details about salary, location, times etc. But this file doesn't link the job to its corresponding role(s) (roles plural, more on that later). For that, we need to look up the role table. Now each role has a set of skills which aren't listed either in the master table. For those, we need to refer the filter table. The linking entry to all these tables together is "job_id", which is a unique identifier of the job. In principle, one can keep the tables saparately and do the analyis by referring to two or more of them anytime outside information is needed but there is one problem with that.

The probelem is that the role that defines a job is not unique. In other words, there may be multiple roles associated with each job. And in the role table, these multiple entries aren't listed in the same row but different rows, at times with far away indices. The same goes for skills. The number of skills defining a job is almost alwyays greater than one and they too are not found in one place in the filter table. So as part of cleaning up and aggrageting all the information, it becomes essential to put all the data related to an individual job at one place without the need to parse thousand of rows to find one attribute of it.

So data from 3 tables (namely master, filter and role) is tied together in the following way: Master file is taken as the base table. For each job_id in master, all the roles associated to it in the role table are put in the list and a column of such lists corresponding to all jobs is appended to the already existing 16 columns in master file. The same action is performed with respect to skills where the skills are read from filter table and added in the list form in the master base file. Luckily, all of this is done in one line via merge command in R the script for which is below.

```
#MAIN
all_roles <- unique(role_data$role)
num_roles <- length(all_roles)
all_skills <- unique(filter_data$skill)
num_skills <- length(all_skills)

all_data_DT = data.table(all_data)
role_data_DT = data.table(role_data)
filter_data_DT = data.table(filter_data)
```

```
role_data_unique = role_data_DT[ , lapply(.SD, list), by = job_id]
filter_data_unique = filter_data_DT[ , lapply(.SD, list), by = job_id]
filter_data_unique$source <- NULL

all_role_data <- merge(all_data,role_data_unique,by="job_id",all.x=TRUE)
all_filter_data <- merge(all_data,filter_data_unique,by="job_id",all.x=TRUE)

all_role_filter_data <- merge(all_role_data,filter_data_unique,by="job_id",all.x=TRUE)
arfd <- all_role_filter_data
```

arfd variable defined above now consists of all the columns from original master file as well as two addition columns for roles and skills respectively each of which are stored in their individual lists. But as expected, some incongruencies in the data necessitate a resolution. Namely, the job_ids in the master table don't all match with those in role table, i.e. there are some jobs in master file with no corresponding entries in the role table and vice versa. The same is true for filter table, although less often. The way this is dealt with is by only considering the jobs which are in the master file since that is the primary (and most) source of information. The option all.x = TRUE above in merge function is nothing but the encoding of this choice.

How many nurses, doctor?

In the code pasted above, we have alredy obtained the unique list of roles from the complete dataset. Now the natural question arises: which kind of jobs are most frequent? That can be answered by ploting the frequencies of different roles on x-y axis. Note that for this purpose, we directly use the role table and count the occurence of each category. For representational elegance, we only focus on 20 most freuent roles but code can always be modified to include more. We see that RN (registered nurses) are far more common than any other role. So in a way, title fits :).

```
#idea taken from Zuhao's work, count the occurences of roles

# for roles
freq_roles <- data.frame(matrix(0,1,num_roles))
colnames(freq_roles) <- all_roles

for (i in 1:num_roles){
  #freq_roles[i] <- nrow(arfd[which(arfd$role %in% all_roles[i]),]) #is very slow
  freq_roles[i] <- nrow(role_data[which(all_roles[i] == role_data$role),])
}

freq_roles_sorted = sort(freq_roles,decreasing=TRUE)
top_role_freqs = as.vector(freq_roles_sorted,mode="numeric")

op <- par(mar=c(11,4,4,1)) # the 10 allows the names.arg below the barplot
labs <- colnames(freq_roles_sorted)[1:20]
barplot(top_role_freqs[1:20],main="Roles Frequencies",
        names.arg = labs,
        ylab="Frequencies",
        col="skyblue",
        las=2)
rm(op)
```

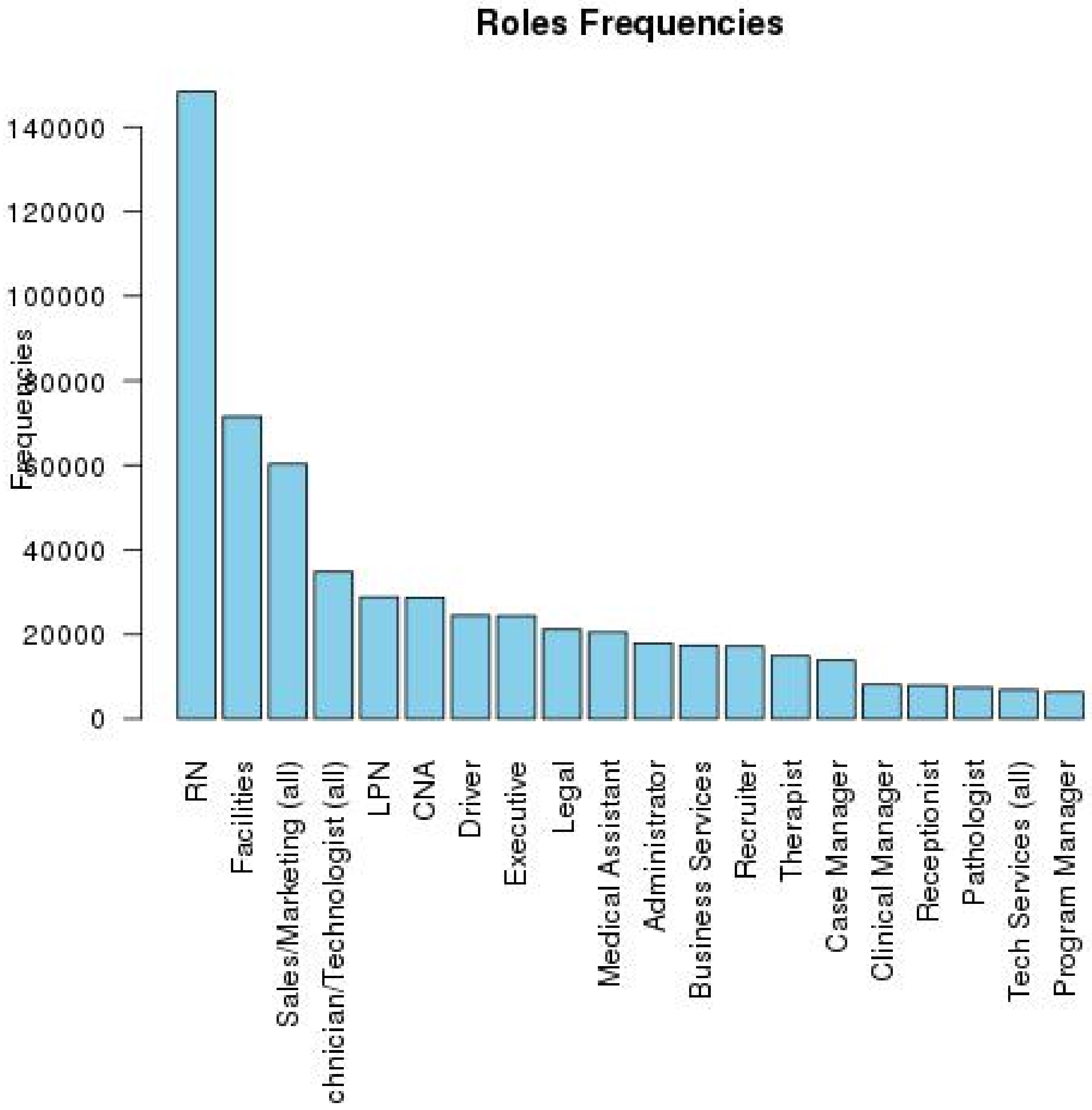

Roles Frequencies
Frequencies
140000
120000
100000
80000
60000
40000
20000
0
RN
Facilities
Sales/Marketing (all)
chnician/Technologist (all)
LPN
CNA
Driver
Executive
Legal
Medical Assistant
Administrator
Business Services
Recruiter
Therapist
Case Manager
Clinical Manager
Receptionist
Pathologist
Tech Services (all)
Program Manager

Figure 1:

People with skills

A related question that can be raised is which are the most frequent skills among healthcare professionals. A similar probing of filter table can provide us with answers. Code and result for the same is presented below. Here we see education is the most popular skill.

```r
#counting the occurences of skills
freq_skills <- data.frame(matrix(0,1,num_skills))
colnames(freq_skills) <- all_skills

for (i in 1:num_skills){
  freq_skills[i] <- nrow(filter_data[which(all_skills[i] == filter_data$skill),])
}

freq_skills_sorted = sort(freq_skills,decreasing=TRUE)
top_skill_freqs = as.vector(freq_skills_sorted,mode="numeric")

op <- par(mar=c(11,4,4,1)) # the 10 allows the names.arg below the barplot
labs <- colnames(freq_skills_sorted)[1:20]
barplot(top_skill_freqs[1:20],main="Skills Frequencies",
        names.arg = labs,
        ylab="Frequencies",
        col="skyblue",
        las=2)
rm(op)
```

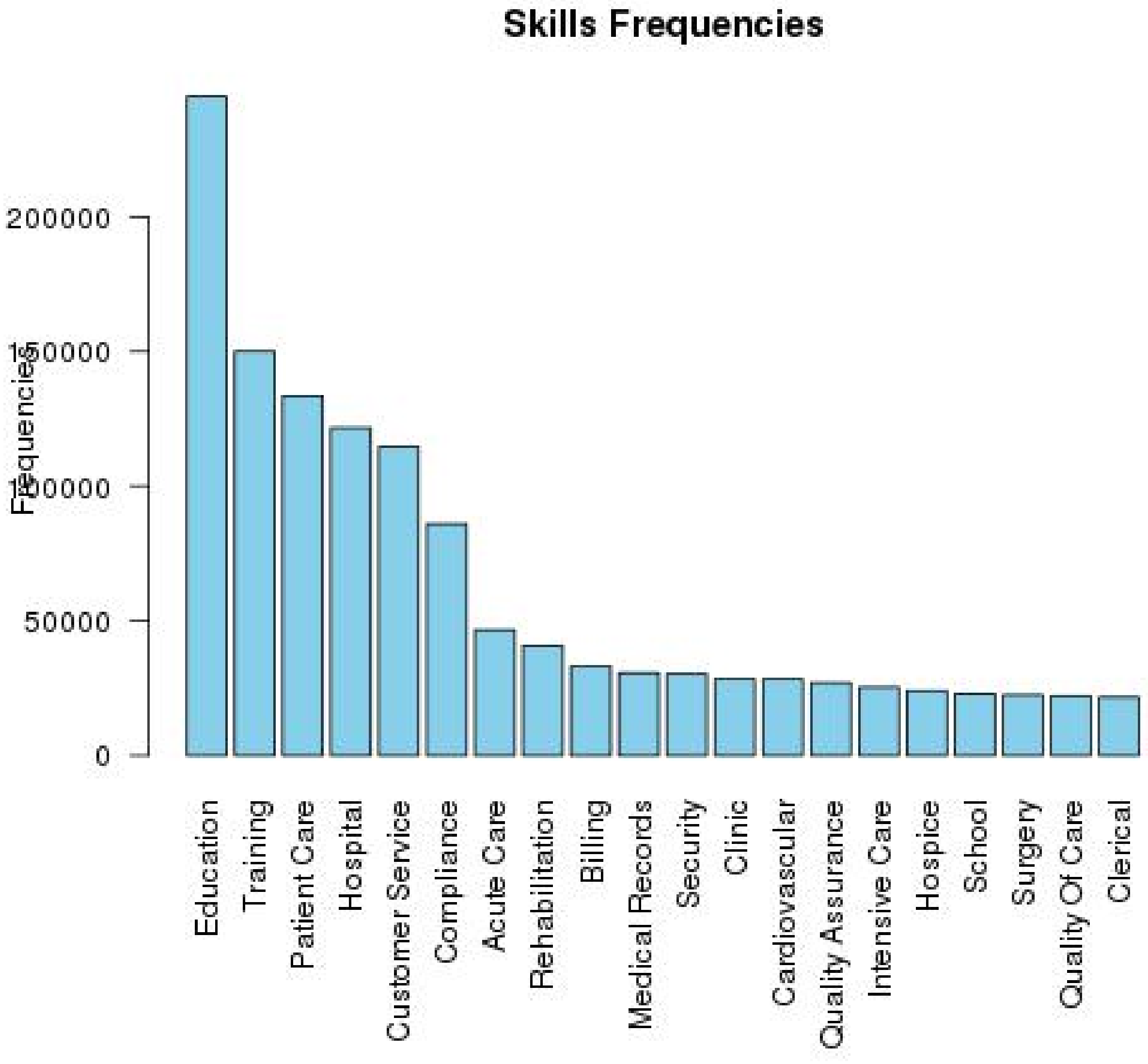

Skills Frequencies
Frequencies
0
50000
100000
150000
200000
Education
Training
Patient Care
Hospital
Customer Service
Compliance
Acute Care
Rehabilitation
Billing
Medical Records
Security
Clinic
Cardiovascular
Quality Assurance
Intensive Care
Hospice
School
Surgery
Quality Of Care
Clerical

Figure 2: .

What role should I play?

Another important question is, which roles are the most valuable? And in a world shaped by monetary desires, we pick but one variable (salary) to present the gains of different roles. The trends shown throw some light on current statistics. Not surprisingly, people in actual patient care earn more than affiliated hostpital staff.

```
#average salary for roles
avsal_roles <- freq_roles
for (i in 1:num_roles){
  print(i)
  temp <- arfd[which(arfd$role %in% all_roles[i]),]
  avsal_roles[i] <- mean(temp$salary)
}

avsal_roles_sorted = sort(avsal_roles,decreasing=TRUE)
top_role_avsals = as.vector(avsal_roles_sorted,mode="numeric")

op <- par(mar=c(11,4,4,1)) # the 10 allows the names.arg below the barplot
labs <- colnames(avsal_roles_sorted)[1:20]
barplot(top_role_avsals[1:20],main="Roles Average Slaries",
        names.arg = labs,
        ylab="Pays",
        col="skyblue",
        las=2)
rm(op)
```

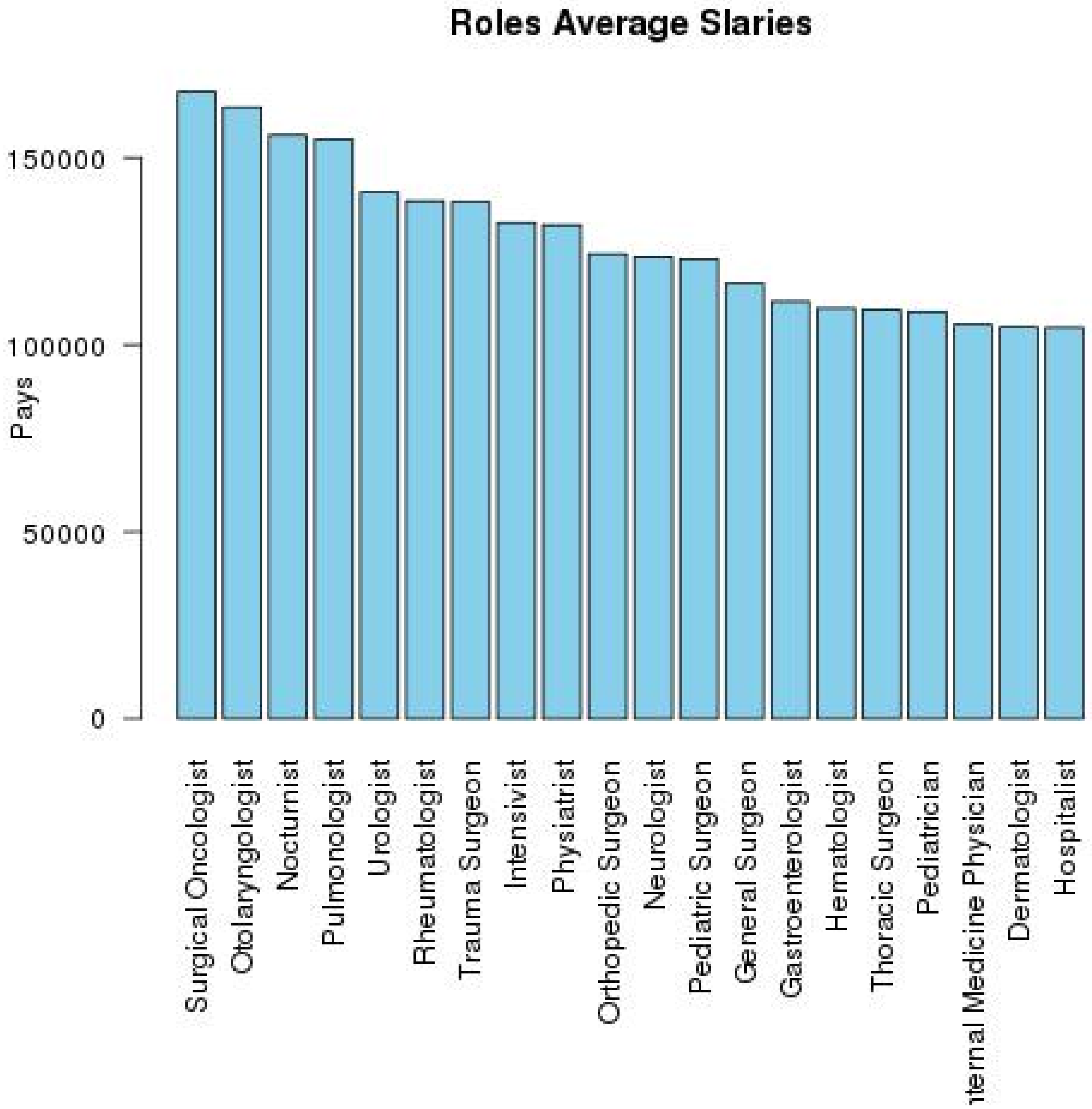

Roles Average Slaries
Pays
150000
100000
50000
0
Surgical Oncologist
Otolaryngologist
Nocturnist
Pulmonologist
Urologist
Rheumatologist
Trauma Surgeon
Intensivist
Physiatrist
Orthopedic Surgeon
Neurologist
Pediatric Surgeon
General Surgeon
Gastroenterologist
Hematologist
Thoracic Surgeon
Pediatrician
nternal Medicine Physician
Dermatologist
Hospitalist

Figure 3: .

Where does the money lie?

A prospective employee in healthcare would be most curious to know which skills lead to most high paying jobs in the current market. Keeping this in view, a bar plot of 20 largest income inducing skills and generating code is presented below. For some inexplicable reason, pediatric dermatology pays much higher than any other job.

```r
#average salary for skills
avsal_skills <- freq_skills
for (i in 1:num_skills){
  print(i)
  temp <- arfd[which(arfd$skill %in% all_skills[i]),]
  avsal_skills[i] <- mean(temp$salary)
}

avsal_skills_sorted = sort(avsal_skills,decreasing=TRUE)
top_skill_avsals = as.vector(avsal_skills_sorted,mode="numeric")

op <- par(mar=c(11,4,4,1)) # the 10 allows the names.arg below the barplot
labs <- colnames(avsal_skills_sorted)[1:20]
barplot(top_skill_avsals[1:20],main="Skills Average Slaries",
        names.arg = labs,
        ylab="Pays",
        col="skyblue",
        las=2)
rm(op)
```

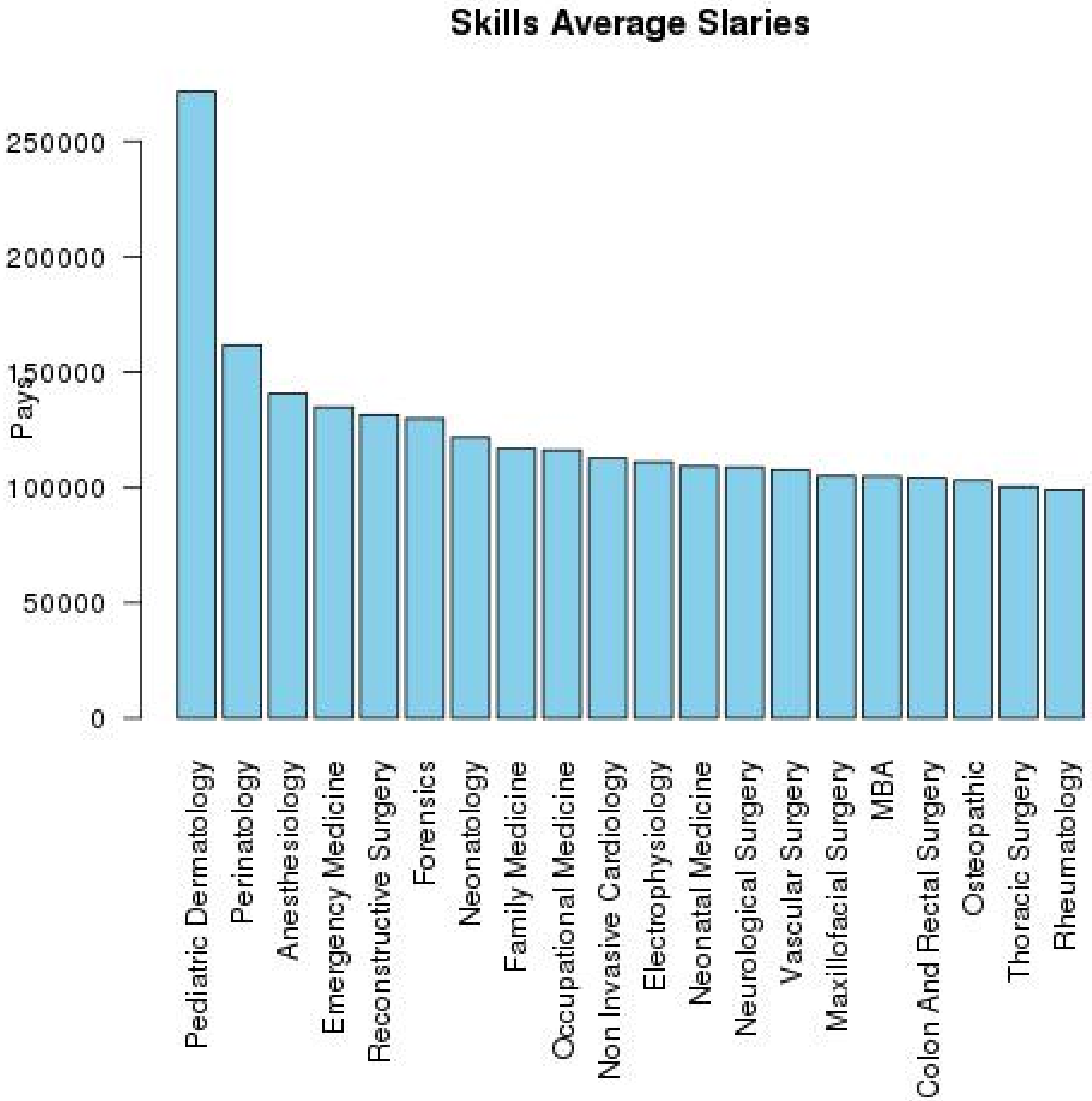

Skills Average Slaries
Pays
250000
200000
150000
100000
50000
0
Pediatric Dermatology
Perinatology
Anesthesiology
Emergency Medicine
Reconstructive Surgery
Forensics
Neonatology
Family Medicine
Occupational Medicine
Non Invasive Cardiology
Electrophysiology
Neonatal Medicine
Neurological Surgery
Vascular Surgery
Maxillofacial Surgery
MBA
Colon And Rectal Surgery
Osteopathic
Thoracic Surgery
Rheumatology

Figure 4: .

Break it down when posting is up

For the reason part of which would be clear in the incoming section, we break down the post_date for the job into year, month and day of the week. The day is fetched just to see (wil be done later) *when* during the week most jobs are posted.

```
#time analysis
arfd_new <- arfd
arfd_new["post_year"] <- NA
arfd_new["post_month"] <- NA
arfd_new["post_day"] <- NA
arfd_new["post_weekday"] <- NA
arfd_new <- arfd_new[, colnames(arfd_new)[c(1:2,19:22,3:18)]] # to change the order of columns
temp <- data.frame(date = arfd_new$post_date)
temp <- separate(d, "date", c("Year", "Month", "Day"), sep = "-")
temp$Year <- as.numeric(temp$Year)
temp$Month <- as.numeric(temp$Month)
temp$Day <- as.numeric(temp$Day)
temp$Month <- month.abb[temp$Month]
arfd_new$post_year <- temp$Year
arfd_new$post_month <- temp$Month
arfd_new$post_day <- temp$Day
arfd_new$post_weekday <- weekdays(as.Date(arfd_new$post_date))
```

arfd now contains 22 columns including 18 old and 4 new columns corresponding to post_year, post_month and post_day and post_weekday added right after the original post_date coulumn.

A tale of missing data

The data is not complete. This may not be an unusual sitatuion when dealing with large datasets, yet it requires a careful treatment. One of the most common missing piece is the fill_date and time_to_fill for a job and about 11% of jobs in the dataset are affected by it. These jobs may never have been filled (within a five month limit reaching which the posting is taken down), or they may have been filled after August 31, 2017, the last date for which information is collected in the avaialable dataset. It all depends on when the job (which hasn't been filled) was posted.

Part of the reason the post date was broken into pieces as mentioned in last section was exactly this. After doing that, unfilled jobs are classified based on the month they are posted in and a freqency table is computed. The output is shown below:

```
#missing entries
# count how many missing entries from each month of posting date
nrow(arfd[which(is.na(arfd$time_to_fill)),])
# output = 70668
temp <- arfd[which(is.na(arfd$time_to_fill)),]
temp <- table(temp$post_year,temp$post_month)
temp <- temp[, colnames(temp)[c(5,4,8,1,9,7,6,2,12,11,10,3)]] # for reordering
#output is
#        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
#2016     0     0     0     0     0     0     0     0   318  1011  1172  1378
#2017  2574  4975    84   139  2681  6388 12157 37791     0     0     0     0
```

What we observe is out of 70668 unfilled jobs, more than half are posted in the last month of the dataset period. Among the remaining, about 3/4 are posted in the threee months period preceiding that. This is not surprising and it shows that most of the jobs which remain unfilled were posted in the last months of dataset period and didn't cross the 5 month cap which would have resulted into taking those jobs down. What *is* surprising is that the number of unfilled jobs with time doesn't linearly increase. Namely, there are far more unfilled jobs in December 16 or February 17 than March 17. The reason for this pattern is yet unclear.

Let's be simple

Regardless of where on the time scale the posting dates of unfilled jobs lie, a pedestrian way of dealing with the missing information is by assiging it the average values from existent data in the same category. In particular, the code below finds all the rows within the same city and state and consiting of the same role which defines the job that has missing time_to_fill entry and fills it with the average time_to_fill values from these rows. The city criteria is relaxed in the event of no such rows. This procedure should be able to stuff most missing entry rows with time_to_fill values but is this even appropriate? Well, in time we hope to find out.

```
# handling missing time_to_fill (takes too much time for now)
missing_ttf_ind <- which(is.na(arfd$time_to_fill)) #indices to the rows for missing time_to_fill
num_missing_ttfl <- length(missing_ttf_ind)
print(paste0("No. of missing time_to_fill entries in arfd: ", num_missing_ttfl))
print(paste0("No. of missing states in missing time_to_fill rows in arfd: ",
             length(which(is.na(arfd[missing_ttf_ind,]$state)))))
print(paste0("No. of missing cities in missing time_to_fill rows in arfd: ",
             length(which(is.na(arfd[missing_ttf_ind,]$city)))))
# that means we can use the state and city information to fill the missing time_to_fill entries
a <- 0
arfd_new <- arfd
for (i in missing_ttf_ind){
  a <- a+1
  print(a)
  missing_entry_jobid <- arfd[i,]$job_id
  missing_entry_state <- arfd[i,]$state
  missing_entry_city <- arfd[i,]$city
  missing_entry_role <- arfd[i,]$role #verified that all these entries have
                                                        #just one role
  temp <- arfd[which((arfd$role %in% missing_entry_role) &
                       (arfd$city == missing_entry_city) &
                       (arfd$state == missing_entry_state) &
                       (!is.na(arfd$time_to_fill)) &
                       (arfd$job_id != missing_entry_jobid)),]
  print(length(temp$job_id))
  if(length(temp$job_id)==0){
    # if city filter is too much, just do the state average for that role
    temp <- arfd[which((arfd$role %in% missing_entry_role) &
                         (arfd$state == missing_entry_state) &
                         (!is.na(arfd$time_to_fill)) &
                         (arfd$job_id != missing_entry_jobid)),]
  }
  if(length(temp$job_id)!=0){ #only replace if there is something to replace with
    arfd_new[i,]$time_to_fill <- ceiling(mean(temp$time_to_fill))
  }
}
```

Juxtapose doctors and drivers

After ranking the frequencies of different roles in each state, it was observed that the order of popularity isn't same in these states even for the top roles. For a relative comparison among states, the code below generates the graph for overall most frequent roles and contribution from each state in this frequeny. What is observed is that the individual counts from these states isn't always reflective of their relative sizes. Possible reasons for this are: certain roles have different titles in different places, or the availability and collection of job postings is nonuniform across different locations due to variety of sources, or the demand for diffent roles is actually different among all four states. Whatever the reason, the graph below certainly looks interesting and gives a clear measure of relative popularity.

```r
#contribution of frequency for roles from each state
Roles <- colnames(freq_roles_sorted)
Roles_FL <- as.vector(freq_roles_sorted_FL,mode="numeric")
Roles_VA <- as.vector(freq_roles_sorted_VA,mode="numeric")
Roles_IL <- as.vector(freq_roles_sorted_IL,mode="numeric")
Roles_TX <- as.vector(freq_roles_sorted_TX,mode="numeric")
Roles <- Roles[2:15]
Roles_FL <- Roles_FL[2:15]
Roles_VA <- Roles_VA[2:15]
Roles_IL <- Roles_IL[2:15]
Roles_TX <- Roles_TX[2:15]

#using https://plot.ly/r/bar-charts/
library(plotly)
data <- data.frame(Roles, Roles_FL, Roles_VA, Roles_IL, Roles_TX)
#to keep the same order instead of deafault alphabetical, we do
data$Roles <- factor(data$Roles, levels=Roles)

plot_ly(data, x = ~Roles, y = ~Roles_FL, type = 'bar',
        name = 'FL portion') %>%
  add_trace(y = ~Roles_VA, name = 'VA portion') %>%
  add_trace(y = ~Roles_IL, name = 'IL portion') %>%
  add_trace(y = ~Roles_TX, name = 'TX portion') %>%
  layout(yaxis = list(title = 'Count'), margin = list(b = 100),
         xaxis = list(title='',ticks = "outside",tickangle=45),
         barmode = 'stack')
```

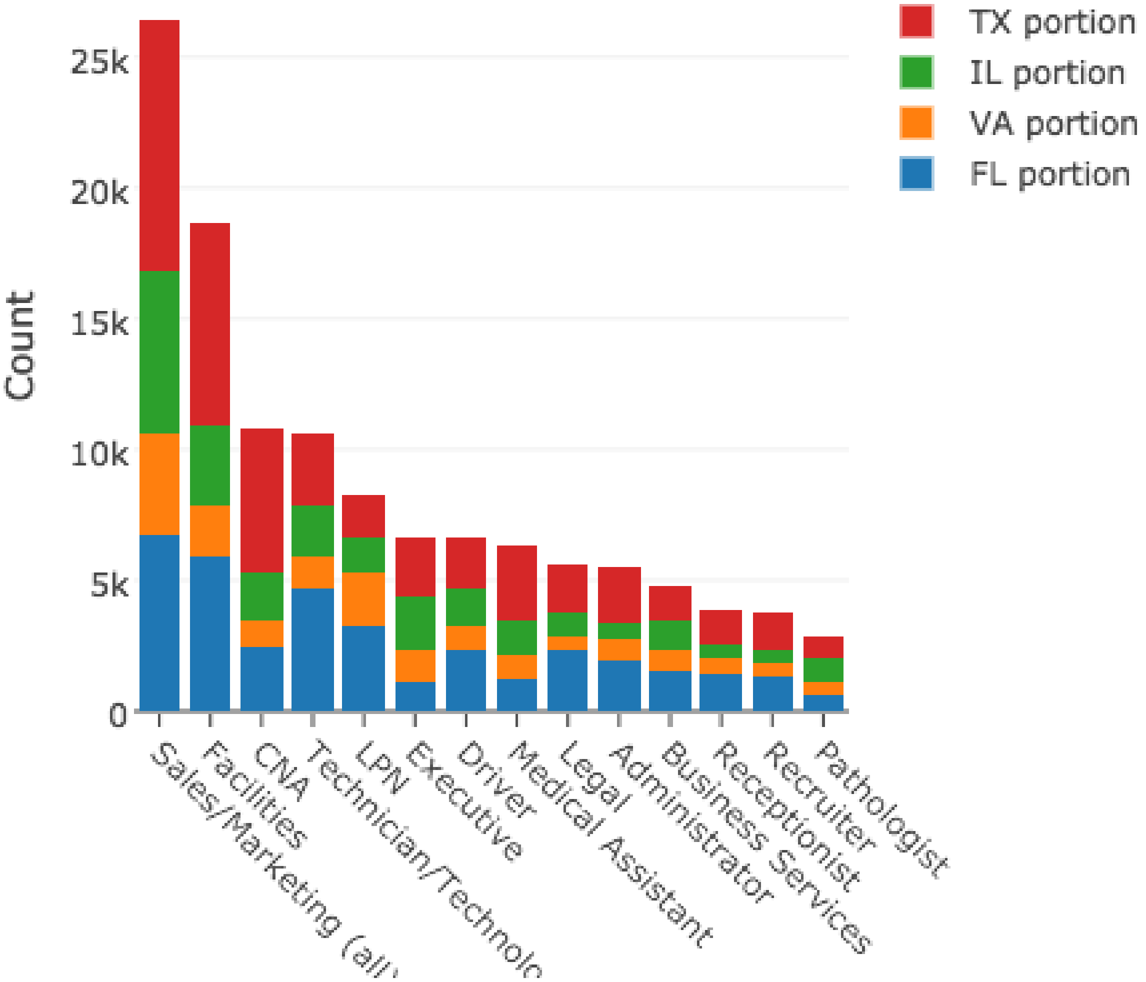

TX portion
IL portion
VA portion
FL portion
Count
25k
20k
15k
10k
5k
0
Sales/Marketing (all
Facilities
CNA
Technician/Technolo
LPN
Executive
Driver
Medical Assistant
Legal
Administrator
Business Services
Receptionist
Recruiter
Pathologist

Figure 5: .

What's the role of state

We already looked at the highest paying roles on a cumulative basis. But graph shown there doesn't reflect the differences *among* the states. So another analysis that was done was that all the states were put on the same graph with display of the highest paying roles in each. As the graph below indicates, topmost roles on this scale aren't uniform across the states and other than few common titles we see that valuation of jobs varies based on location. The code has to be understood in conjunction with previously pasted code blocks in earlier sections.

```
#ordering of avsal for each states w.r.t. in the order of the decreasing
#overall frequency
temp <- order(freq_roles,decreasing=TRUE)
avsal_roles_sorted_FL <- avsal_roles_FL[temp]
avsal_roles_sorted_VA <- avsal_roles_VA[temp]
avsal_roles_sorted_IL <- avsal_roles_IL[temp]
avsal_roles_sorted_TX <- avsal_roles_TX[temp]

top_role_avsals_FL <- as.vector(avsal_roles_sorted_FL,mode="numeric")
top_role_avsals_VA <- as.vector(avsal_roles_sorted_VA,mode="numeric")
top_role_avsals_IL <- as.vector(avsal_roles_sorted_IL,mode="numeric")
top_role_avsals_TX <- as.vector(avsal_roles_sorted_TX,mode="numeric")

par(mfrow=c(2,2))
op <- par(mar=c(10,4,4,1))
labs <- colnames(avsal_roles_sorted_FL)[1:10]
barplot(top_role_avsals_FL[1:10],main="Roles Average Salaries in FL",
        names.arg = labs, col="deepskyblue", las=2)
op <- par(mar=c(10,4,4,1))
labs <- colnames(avsal_roles_sorted_VA)[1:10]
barplot(top_role_avsals_VA[1:10],main="Roles Average Salaries in VA",
        names.arg = labs, col="firebrick1", las=2)
op <- par(mar=c(10,4,4,1))
labs <- colnames(avsal_roles_sorted_IL)[1:10]
barplot(top_role_avsals_IL[1:10],main="Roles Average Salaries in IL",
        names.arg = labs, col="orangered1",las=2)
op <- par(mar=c(10,4,4,1))
labs <- colnames(avsal_roles_sorted_TX)[1:10]
barplot(top_role_avsals_TX[1:10],main="Roles Average Salaries in TX",
        names.arg = labs, col="green", las=2)
dev.off()
```

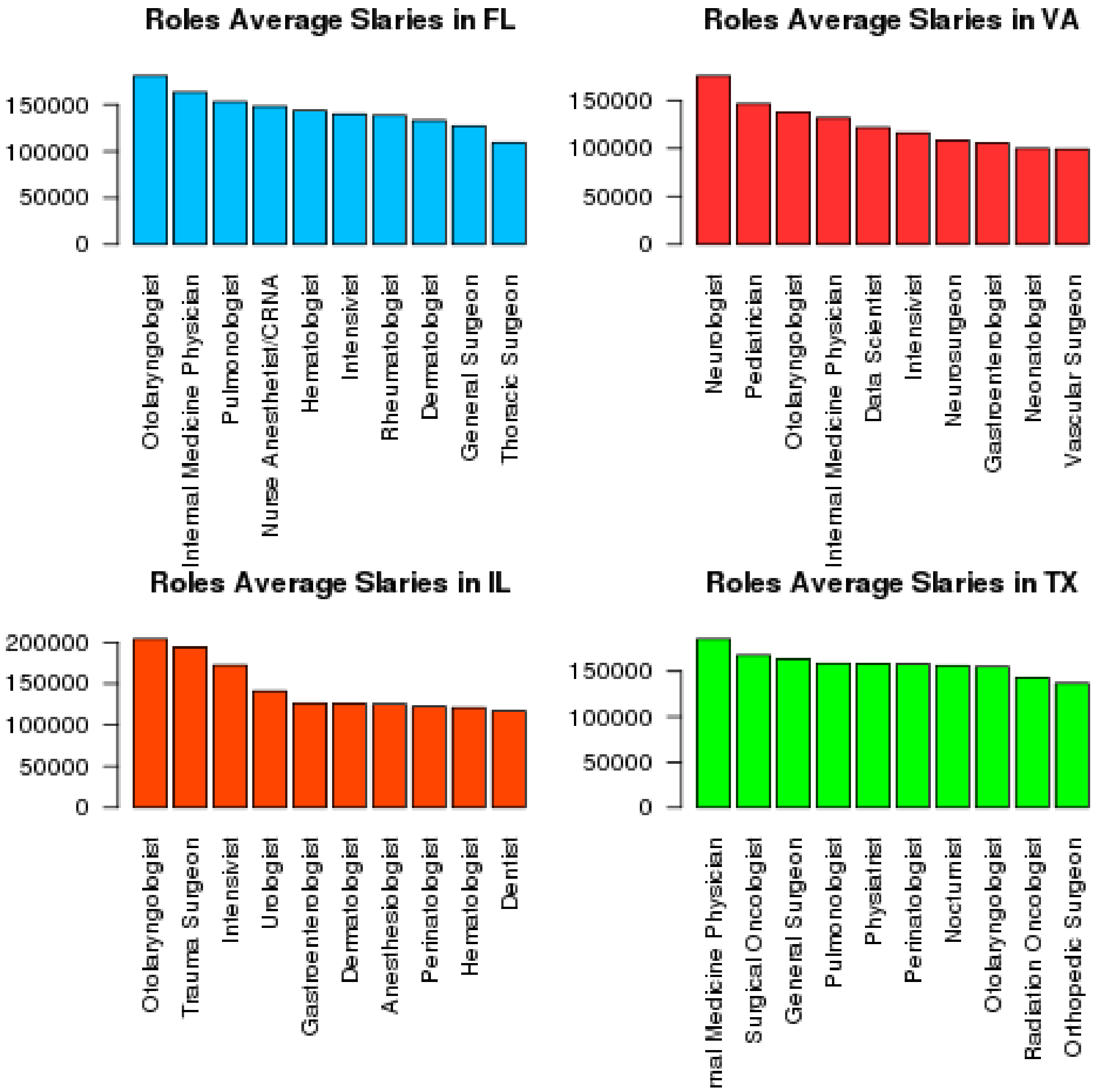

Roles Average Slaries in FL
0
50000
100000
150000
Otolaryngologist
Internal Medicine Physician
Pulmonologist
Nurse Anesthetist/CRNA
Hematologist
Intensivist
Rheumatologist
Dermatologist
General Surgeon
Thoracic Surgeon
Roles Average Slaries in VA
0
50000
100000
150000
Neurologist
Pediatrician
Otolaryngologist
Internal Medicine Physician
Data Scientist
Intensivist
Neurosurgeon
Gastroenterologist
Neonatologist
Vascular Surgeon
Roles Average Slaries in IL
0
50000
100000
150000
200000
Otolaryngologist
Trauma Surgeon
Intensivist
Urologist
Gastroenterologist
Dermatologist
Anesthesiologist
Perinatologist
Hematologist
Dentist
Roles Average Slaries in TX
0
50000
100000
150000
mal Medicine Physician
Surgical Oncologist
General Surgeon
Pulmonologist
Physiatrist
Perinatologist
Nocturnist
Otolaryngologist
Radiation Oncologist
Orthopedic Surgeon

Figure 6: .

Where does this role play a role?

The statewise listing of highest playing roles fails to give a clue regarding the relative valuation of a role compared to other states. For this purpose, one scale has to be kept the same, namely the titles. For the next graph therefore, a few most frequent roles are chosen and average salaries for these roles for all states are plotted together. This gives a clear view of where the role is paid highest. We observe that statewise average is highest in Illinois for most roles which makes sense as it has more urban population. This (and other salaries) difference is most striking among the roles which require richer set skills, e.g. Legal and Executive, compared to say RN or Facilities. Needless to say, same code could be applied to rank all the roles down the frequency list to observe the statewise differences.

```
# comparison of salaries for roles in different states
Roles <- colnames(freq_roles_sorted)
Salaries_FL <- as.vector(avsal_roles_sorted_FL,mode="numeric")
Salaries_VA <- as.vector(avsal_roles_sorted_VA,mode="numeric")
Salaries_IL <- as.vector(avsal_roles_sorted_IL,mode="numeric")
Salaries_TX <- as.vector(avsal_roles_sorted_TX,mode="numeric")
Roles <- Roles[1:10]
Salaries_FL <- Salaries_FL[1:10]
Salaries_VA <- Salaries_VA[1:10]
Salaries_IL <- Salaries_IL[1:10]
Salaries_TX <- Salaries_TX[1:10]

library(plotly)
data <- data.frame(Roles, Salaries_FL, Salaries_VA, Salaries_IL, Salaries_TX)
#to keep the same order instead of deafault alphabetical, we do
data$Roles <- factor(data$Roles, levels=Roles)

plot_ly(data, x = ~Roles, y = ~Salaries_FL, type = 'bar',
        name = 'FL Salary') %>%
  add_trace(y = ~Salaries_VA, name = 'VA Salary') %>%
  add_trace(y = ~Salaries_IL, name = 'IL Salary') %>%
  add_trace(y = ~Salaries_TX, name = 'TX Salary') %>%
  layout(yaxis = list(title = 'Count'), margin = list(b = 100),
         xaxis = list(title='',ticks = "outside",tickangle=45),
         barmode = 'group')
```

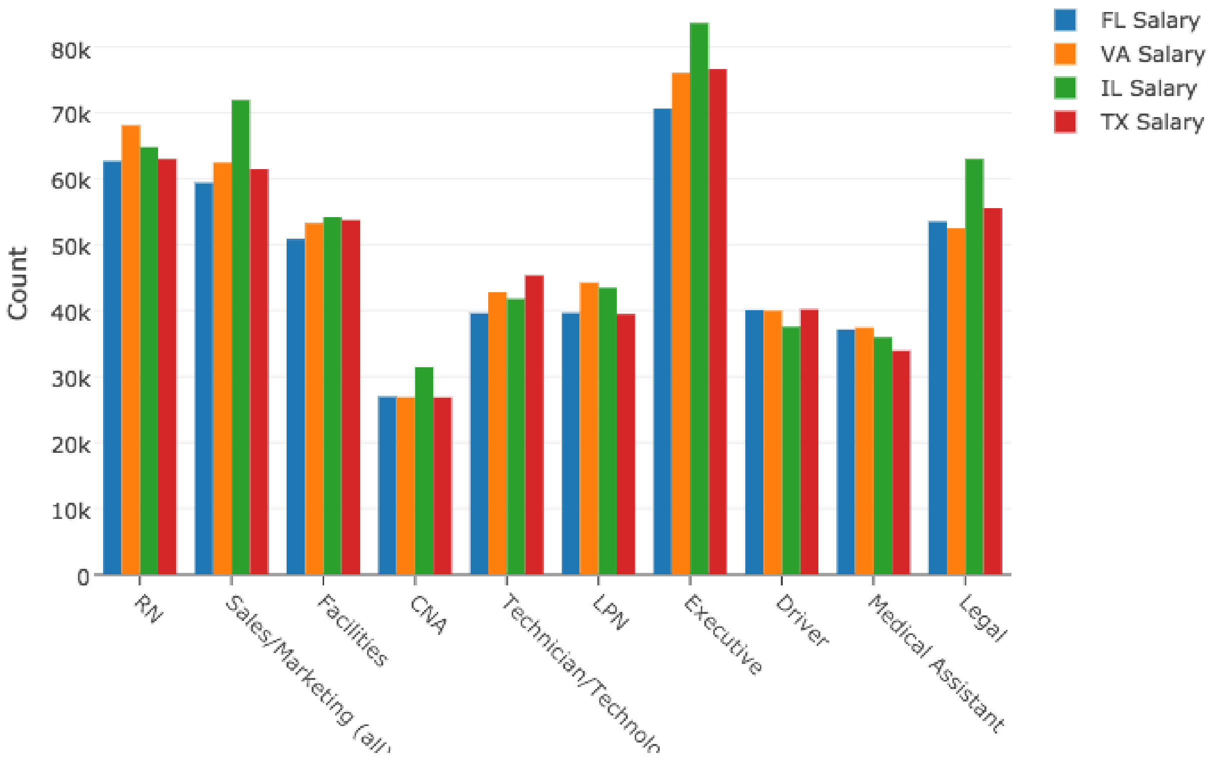

FL Salary
VA Salary
IL Salary
TX Salary
Count
0
10k
20k
30k
40k
50k
60k
70k
80k
RN
Sales/Marketing (all)
Facilities
CNA
Technician/Technolo
LPN
Executive
Driver
Medical Assistant
Legal

Figure 7: .

Common careers combined

Clustering is a good way to see the pattern among different attributes of similar roles. Two such important attributes are no. of occurances and average salary. Normally it would be hard to see a strict relation even with enough data points but clustering helps to understand it better. The graph below shows the data points for 30 most frequent roles on freqency and average salary scales. From the lack of any observable pattern, we conclude that high frequency roles don't necessarily correspond to the lowest paid jobs. In fact, they offer more median range salaries. Clustering was done using k-means from inbuilt function in r.

```r
#clustering of roles with respect to frequencies and average salaries
#decreasing frequency (no pattern)
temp <- freq_avsal_roles[order(freq_avsal_roles$frequency,
                               decreasing = TRUE),]
temp <- temp[1:30,]
set.seed(123456789) ## to fix the random starting clusters
grpRoles <- kmeans(temp[,c("frequency","average_salary")],
                   centers=5, nstart=10)
# to see the cluster assignments
o=order(grpRoles$cluster)
data.frame(temp$role[o],grpRoles$cluster[o])
plot(temp$average_salary, temp$frequency,
     type="n", xlab="Average Salary", ylab="Frequency")
text(x=temp$average_salary, y=temp$frequency, col=grpRoles$cluster+1)
```

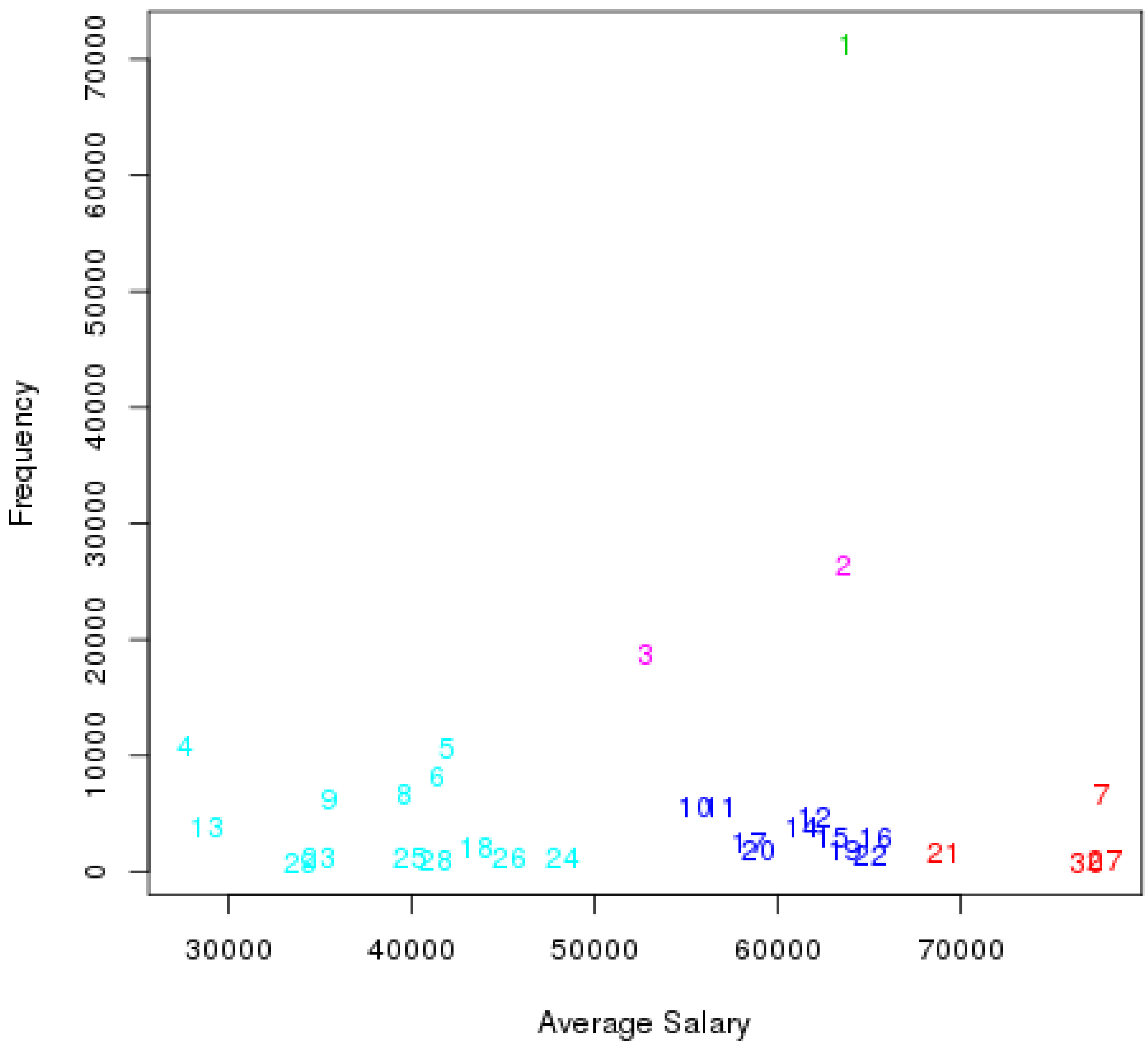

Frequency
0
10000
20000
30000
40000
50000
60000
70000
Average Salary
30000
40000
50000
60000
70000
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

Figure 8: .

Rich roles nearby

Another kind of clustering on the same x and y scale gives a nice pattern. For this we cluster based on the average salary of roles and put the points on the average time to fill and salary scale again. Here we see an overall inverse behavior which indicates that higher the job salary is, more rarely it will be posted. This behavior is intuitive and understandable. One could ask the question of the specific gains of cluserting because expectantly, such a behaviour could have been obtained by plotting the two axis variables. The answer to this concern is that the clustering dwarfs to the noise, specifically with regard to very low occuring jobs. In the example below, cluster means can shown to follow a nice inverse trend which is helpful in deducing the relationship.

```r
#decreasing salaries (pattern)
png(filename="clustering_roles_average_salary.png")
temp <- freq_avsal_roles[order(freq_avsal_roles$average_salary,
                               decreasing = TRUE),]
temp <- temp[1:20,]
set.seed(123456789) ## to fix the random starting clusters
grpRoles <- kmeans(temp[,c("frequency","average_salary")],
                   centers=4, nstart=10)
# to see the cluster assignments
o=order(grpRoles$cluster)
data.frame(temp$role[o],grpRoles$cluster[o])
plot(temp$average_salary, temp$frequency,
     type="n", xlab="Average Salary", ylab="Frequency")
text(x=temp$average_salary, y=temp$frequency, col=grpRoles$cluster+1)
```

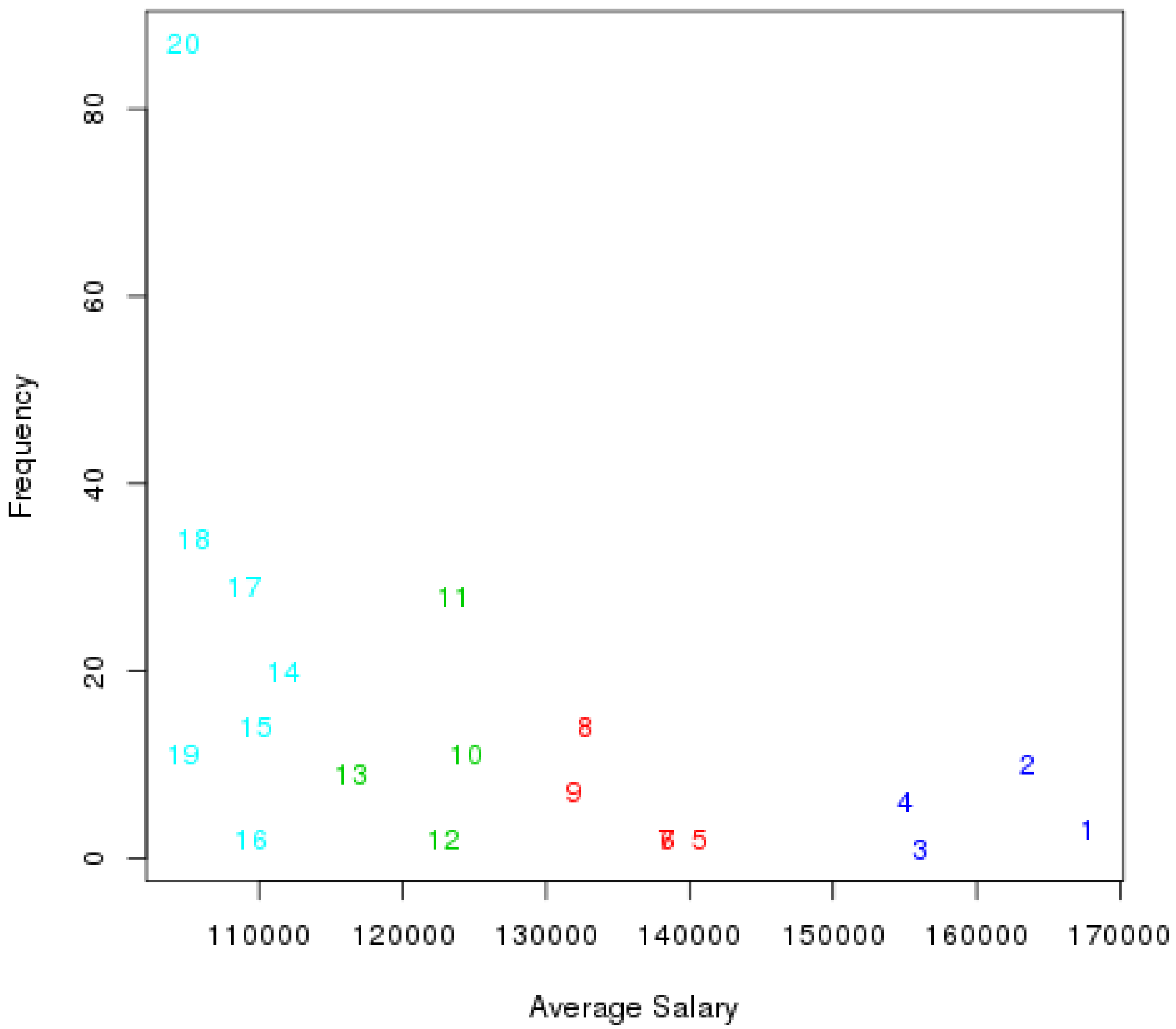
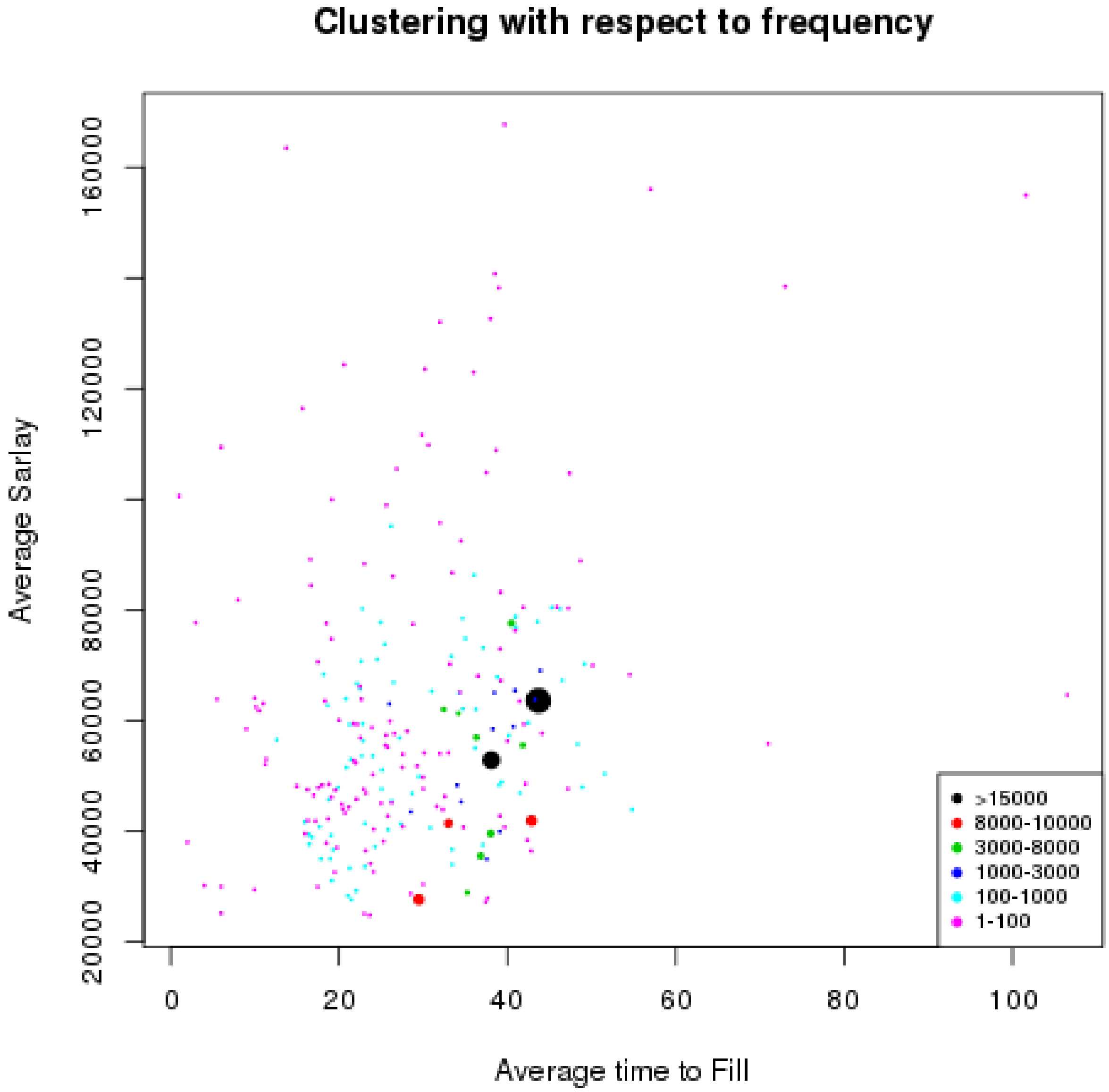

Frequency
0
20
40
60
80
Average Salary
110000
120000
130000
140000
150000
160000
170000
20
18
17
11
14
15
8
19
10
2
13
9
4
1
16
12
7
6
5
3

Figure 9: .

Long wait for whom?

Another important attribute for jobs is the average time it takes for different roles to get the posting filled. For this we look at graph bewteen average time to fill and average salary for the respective roles. This grpah as expected spread uniformly on the plane. We now cluster these data points based on frequency of appearance of these roles. The sizes of the dots in the graph represent the corresponding counts of jobs for that role and these numbers are shown in the legend. What we observe is that less common roles (i.e. the jobs which are ponsted only once in while) have varying range of salaries and take times to fill which also have large scope. But as the role occurrences increase (i.e. most common jobs in healthcare) lie in a limited space on this plane. These jobs offer salary in a small (close to median) range and take a somewhat regular time to get filled (this is clear from the positions of larger spheres in the plot).

```r
role <- all_roles
average_ttf <- as.vector(avttf_roles,mode="numeric")
average_salary <- as.vector(avsal_roles,mode="numeric")
frequency <- as.vector(freq_roles,mode="numeric")
temp = data.frame(role, average_ttf, average_salary, frequency)
avttf_avsal_roles <- temp[which(!is.na(temp$average_ttf)
                                & !is.na(temp$average_salary)
                                & !is.na(temp$frequency)),]

#clustering of roles with respect to frequencies on average salary
#and average time to fill scale
temp <- avttf_avsal_roles[order(avttf_avsal_roles$frequency,
                               decreasing = TRUE),]
temp <- temp[2:lengths(temp)[1],]
#temp <- temp[1:30,]
set.seed(123456789) ## to fix the random starting clusters
#grpRoles <- kmeans(temp[,c("average_ttf","average_salary")],
 #                  centers=5, nstart=10)
grpRoles <- kmeans(temp[,c("frequency")],
                   centers=5, nstart=5000)
#the labeling of clusters by default is not sorted so we change that

grpRoles$cluster[1:2] <- c(1,1)
grpRoles$cluster[3:5] <- c(2,2,2)
grpRoles$cluster[6:13] <- rep(3,8)
#grpRoles$cluster[39:103] <- rep(6,65)
grpRoles$cluster[104:244] <- rep(6,141)

# to see the cluster assignments
o=order(grpRoles$cluster)
data.frame(temp$role[o],grpRoles$cluster[o], temp$frequency[o])
png(filename="clustering_roles_avttf_avsal.png")
plot(temp$average_ttf, temp$average_salary,
     col =(grpRoles$cluster),
     main="Clustering with respect to frequency",
     pch=20, cex=temp$frequency/10000,
     ylab = "Average Sarlay",
     xlab = "Average time to Fill")
legend("bottomright",
       legend = c(">15000","8000-10000", "3000-8000",
                  "1000-3000", "100-1000","1-100"),
       pch=20, col=1:6, pt.cex=1, cex = 0.7)
```


Clustering with respect to frequency
Average Sarlay
20000
40000
60000
80000
120000
160000
0
20
40
60
80
100
Average time to Fill
>15000
8000-10000
3000-8000
1000-3000
100-1000
1-100

New Work to be graded

(will be merged together later in appropriate sections)

From having very little idea about how to analyze this huge dataset and only knowing some goals that we aimed to fulfull through our work, we started to explore different directdions. In a collaborative effort, where each team member brought unique set of skills to the table and made significant contribution to the common objective, we have been able to produce some compelling results. To minimize the repetition, I will briefly describe the group collaboration (excluding mine) under the next headin. After that I am going to describe my point of view of the project and the work and analysis performed just by me (to maintain the universality of the prose, I am going to use passive voice but unless otherwise specified, agent of the action is me).

Group Contribution

My teammates have been working on different aspects of this project. Wilson has been wroking on merging of role hierarchy and role family. Huan has been looking into correlations among urban/rural population and average salary. Zuohao has been investigating the role of specfic skills in offered pays for different jobs as well as looking at the specific profiles for different companies.

My Unique Contribution

Upto the last submission, I had cleaned up the data, merged different tables togehter for ease of use and anallyzed the salary variations among top roles and skills. I had also described a simple technique to deal with missing data. Then I had looked at the statewise differences in frequencies of jobs for top roles as well as salary variations among different locations. My clustering analysis consisted of putting the roles on average salary and frequency/average time to fill scale and grouping them based on frequencies.

The new work in this submission starting from the next section includes an analysis of citywise spread of job postings in all four states on the map of the United States. We go a step further and try to explain the strange results in the frequencies of top roles based on this visualization. Furthermore, I did the time based analysis for job over the period of postings avaialable in the database.

Spatial separation

A prospective employee wants to scan the map of the country and hopes to pinpoint where they will have the option of most potential job opportunities in their chosen area of work. A number alone might look big on paper but becomes small when seen relative to another even larger number. The effect is even amplified when the numbers are represented not in the numeric form but as size of an object. The graph below does exactly that. To create this, all the job postings were classified into their loations among different cities. An external data set was imported to obtain the information of latitudes and longitudes of these cities and finally each city was displayed with a sphere of size indicating the relative number of postings in that city. This graph and next couple of graphs are more impactful when seen using an interactive tool, nonetheless the overall design is still visible on still paper.

```r
#putting the jobs on US-map
#city and state wise frequency table
temp <- as.data.frame(table(arfd$city,arfd$state))
colnames(temp)[1] <- "city"
colnames(temp)[2] <- "state"
colnames(temp)[3] <- "freq"
```

```
#removing the rows corresponding to non existent cities in the all states
temp <- temp[-which(temp$freq==0),]

#downloaded from
#https://www.gaslampmedia.com/download-zip-code-latitude-
#     longitude-city-state-county-csv/
df1 <- read.csv("~/data/DARL_Fall2017/GREENWICH/abhi/zip_codes_states.csv")
df1 <- df1[which(df1$state %in% c('VA','TX','IL','FL')),]
df1$zip_code <- NULL
df1$county <- NULL
df1 <- aggregate(.~city+state, df1, mean)

df1 <- merge(df1,temp,by=c("city","state"))
colnames(df1)[3] <- "lat"
colnames(df1)[4] <- "lon"

#doesn't list all the cities in the arfd file but then we never had all #the lat and lon in the first p

library(plotly)
#using https://plot.ly/r/bubble-maps/ downloaded csv file is below
#df <- read.csv("~/data/DARL_Fall2017/GREENWICH/abhi/2014_us_cities.csv")

df1$q <- with(df1, cut(freq, quantile(freq)))
levels(df1$q) <- paste(c("1st", "2nd", "3rd", "4th", "5th"), "Quantile")
df1$q <- as.ordered(df1$q)

g <- list(
  scope = 'usa',
  projection = list(type = 'albers usa'),
  showland = TRUE,
  landcolor = toRGB("gray85"),
  subunitwidth = 1,
  countrywidth = 1,
  subunitcolor = toRGB("white"),
  countrycolor = toRGB("white")
)
t <- list(
  family = "sans serif",
  size = 12,
  color = 'blue')

plot_geo(df1, locationmode = 'USA-states', sizes = c(1,75)) %>%
  add_markers(
    x = ~lon, y = ~lat, size = ~freq, hoverinfo = "text",
    text = ~paste(df1$city, "<br />", df1$freq, " jobs")
  ) %>%
  layout(title=paste('# of healthcare jobs-citywise',
                     '<br>   '), font=t, geo = g)
```

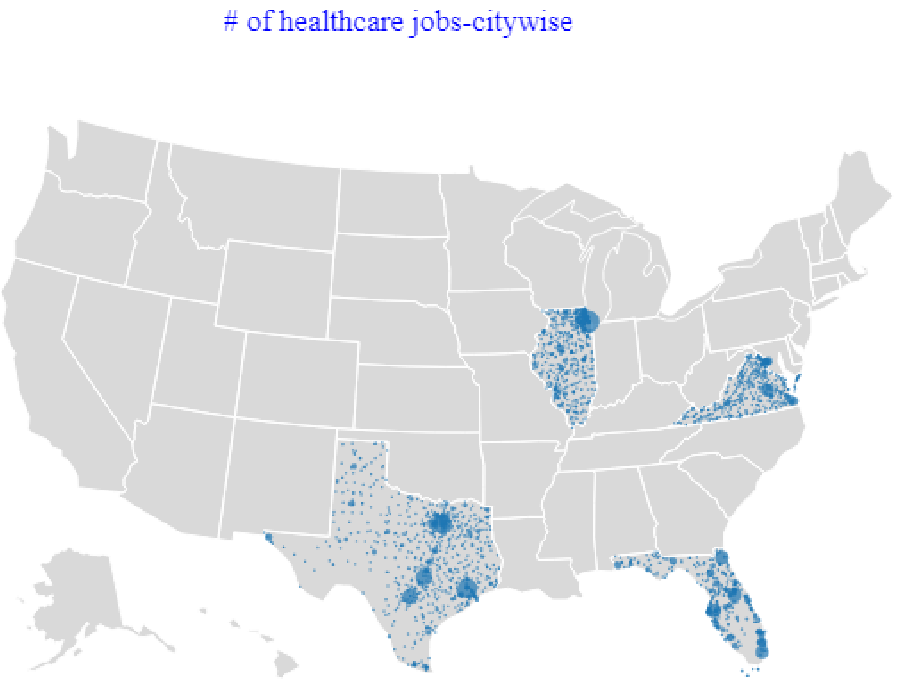


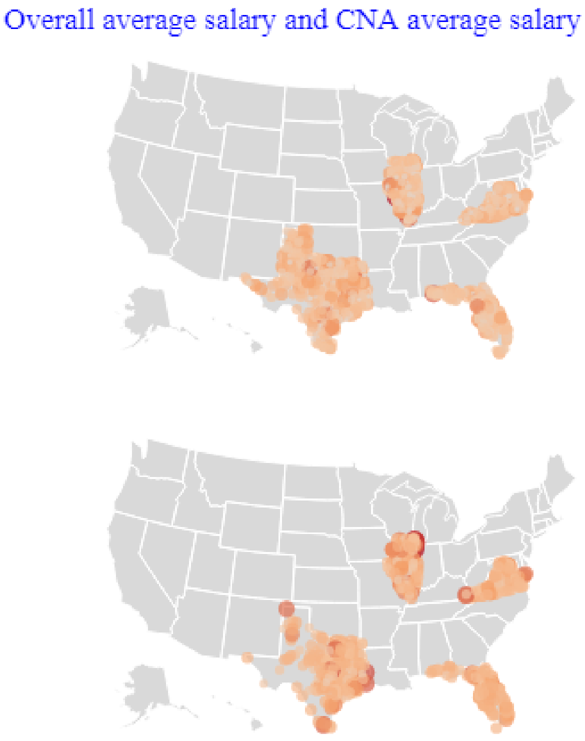
Figure 10:

CNA sole salary

One of the key observation that can be made from frequency contribution graph from different states (see section: Juxtapose doctors and drivers) is that the no. of CNA postings in Florida aren't on a par with its size and the postings contributed from state among other top roles. One visual way to explore this is by looking at average salary difference for CNA from typical average salary (taking all roles into account). This is presented in the next set of maps. What we see is the number and intensity of darker spots going down from first to second graph in the region of Florida. This clearly indicates the CNA roles aren't popular in the state because they are not paid in accordance with the average salaries of other roles in healthcare. This analysis is pretty qualitative and is supposed to present a visulization tool to understand how one factor possibly might be responsible for the differences observed.

```r
temp <- arfd[,c("city","state","salary")] #for all roles
temp_sp <- arfd[which(arfd$role %in% "CNA"),] #for one particular role
temp_sp <- temp_sp[,c("city","state","salary")]

#contains the mean of salries for all cities
temp_fn <- function(temp) { #input would be temp or temp_sp
  temp <- aggregate(.~city+state, temp, mean)
  df1 <- read.csv("~/data/DARL_Fall2017/GREENWICH/abhi/zip_codes_states.csv")
  df1 <- df1[which(df1$state %in% c('VA','TX','IL','FL')),]
  df1$zip_code <- NULL
  df1$county <- NULL
  #contains the mean of latitudes and longitudes
  #(taken from outside data source as our data is not
  # complete with all location information)
  df1 <- aggregate(.~city+state, df1, mean)
  df1 <- merge(df1,temp,by=c("city","state"))
  df1$salary <- round(df1$salary, digits = 2)
  colnames(df1)[3] <- "lat"
  colnames(df1)[4] <- "lon"
  return(df1)
}
df1 <- temp_fn(temp)
df1_sp <- temp_fn(temp_sp)

library(plotly)
g <- list(
  scope = 'usa',
  projection = list(type = 'albers usa'),
  showland = TRUE,
  landcolor = toRGB("gray85"),
  subunitwidth = 1,
  countrywidth = 1,
  subunitcolor = toRGB("white"),
  countrycolor = toRGB("white")
)
t <- list(
  family = "sans serif",
  size = 12,
  color = 'blue')

par(mfrow=c(2,1))
```

```
p1 <- plot_geo(df1, locationmode = 'USA-states', sizes = c(1,75)) %>%
  add_markers(
    x = ~lon, y = ~lat, size = ~salary/1e17,
    marker=list(color = ~salary, showscale=FALSE),
    colors = 'Reds', hoverinfo = "text",
    showlegend=FALSE,
    text = ~paste(df1$city, "<br />$", df1$salary, ".")
  ) %>%
  layout(title=paste('Average Salaries of different cities',
                     '<br>   '), font=t, geo = g)

p2 <- plot_geo(df1_sp, locationmode = 'USA-states', sizes = c(1,75)) %>%
  add_markers(
    x = ~lon, y = ~lat, size = ~salary/1e17,
    marker=list(color = ~salary, showscale=FALSE),
    colors = 'Reds', hoverinfo = "text",
    showlegend=FALSE,
    text = ~paste(df1_sp$city, "<br />$", df1_sp$salary, ".")
  ) %>%
  layout(title=paste('Overall average salary and CNA average salary',
                     '<br>   '), font=t, geo = g)

subplot(p1,p2, nrows = 2, margin = 0.05, titleX = TRUE, titleY = TRUE)
```



Overall average salary and CNA average salary

Figure 11:

State the state of states

At this point, we are mature enough to zoom in on one state and see the distribution of jobs (all). Not playing any favorites, here we represent corresponding map for each state. What we see is a very biased distribution in all states (except may be Florida) where few big metropolitan cities claim the largest share of jobs postings. This pattern is not surprising although strikes emphatically when seen on a geographical image.

```r
# shading of states with respect to no. of jobs
#reference https://plot.ly/r/county-level-choropleth/ read data here using
#df <- read.csv("~/data/DARL_Fall2017/GREENWICH/abhi/californiaPopulation.csv")
library(tidyverse)
library(plotly)

df1 <- arfd[which(arfd$state=="FL"),12:18]
df2 <- arfd[which(arfd$state=="IL"),12:18]
df3 <- arfd[which(arfd$state=="VA"),12:18]
df4 <- arfd[which(arfd$state=="TX"),12:18]

cali1 <- map_data("county") %>% filter(region == 'florida')
cali2 <- map_data("county") %>% filter(region == 'illinois')
cali3 <- map_data("county") %>% filter(region == 'virginia')
cali4 <- map_data("county") %>% filter(region == 'texas')

pop1 <- summarize(group_by(df1,county), count=n())
pop2 <- summarize(group_by(df2,county), count=n())
pop3 <- summarize(group_by(df3,county), count=n())
pop4 <- summarize(group_by(df4,county), count=n())

pop1$county <- tolower(pop1$county) #matching string
pop2$county <- tolower(pop2$county) #matching string
pop3$county <- tolower(pop3$county) #matching string
pop4$county <- tolower(pop4$county) #matching string

cali_pop1 <- merge(cali1, pop1, by.x = "subregion",
                   by.y = "county",all.x=FALSE, all.y=TRUE)
cali_pop2 <- merge(cali2, pop2, by.x = "subregion",
                   by.y = "county",all.x=FALSE, all.y=TRUE)
cali_pop3 <- merge(cali3, pop3, by.x = "subregion",
                   by.y = "county",all.x=FALSE, all.y=TRUE)
cali_pop4 <- merge(cali4, pop4, by.x = "subregion",
                   by.y = "county",all.x=FALSE, all.y=TRUE)

cali_pop1$pop_cat <- cut(cali_pop1$count,
                         breaks = c(seq(0, 30000, by = 3000)), labels=1:10)
cali_pop2$pop_cat <- cut(cali_pop2$count,
                         breaks = c(seq(0, 35000, by = 5000)), labels=1:7)
cali_pop3$pop_cat <- cut(cali_pop3$count,
                         breaks = c(seq(0, 25000, by = 5000)), labels=1:5)
cali_pop4$pop_cat <- cut(cali_pop4$count,
                         breaks = c(seq(0, 40000, by = 4000)), labels=1:10)

#to order the table according to proper order column (very important)
cali_pop1 <- cali_pop1[order(cali_pop1$order),]
cali_pop2 <- cali_pop2[order(cali_pop2$order),]
```

```
cali_pop3 <- cali_pop3[order(cali_pop3$order),]
cali_pop4 <- cali_pop4[order(cali_pop4$order),]

state_plot <- function(cali_pop,stname){
  p <- cali_pop %>%
    group_by(group) %>%
    plot_ly(x = ~long, y = ~lat, color = ~pop_cat, colors = 'Reds',
            text = ~subregion, hoverinfo = 'text', showlegend= FALSE) %>%
    add_polygons(line = list(width = 0.4)) %>%
    add_polygons(
      fillcolor = 'transparent',
      line = list(color = 'black', width = 0.5),
      showlegend = FALSE, hoverinfo = 'none'
    ) %>%
    layout(
      title = paste(stname, ": # of jobs by County", sep=" "),
      titlefont = list(size = 15),
      xaxis = list(title = "", showgrid = FALSE,
                   zeroline = FALSE, showticklabels = FALSE),
      yaxis = list(title = "", showgrid = FALSE,
                   zeroline = FALSE, showticklabels = FALSE)
    )
  return(p)
}

state_plot(cali_pop1,"FL")
state_plot(cali_pop2,"IL")
state_plot(cali_pop3,"VA")
state_plot(cali_pop4,"TX")
```

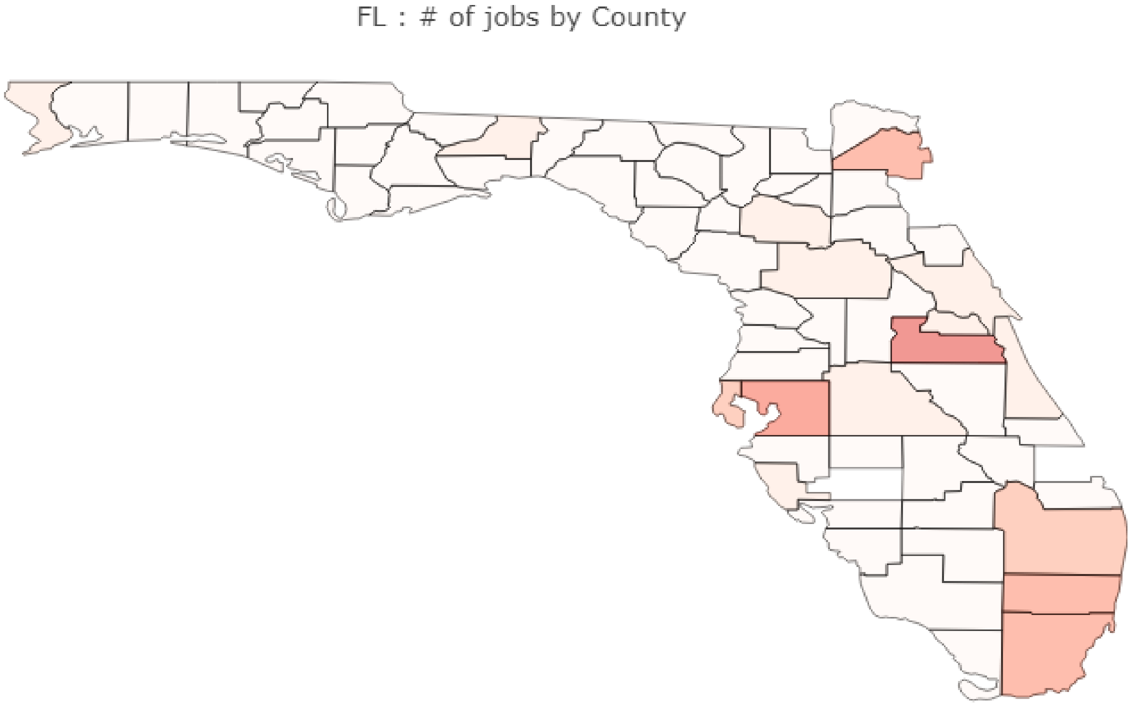


Figure 12:

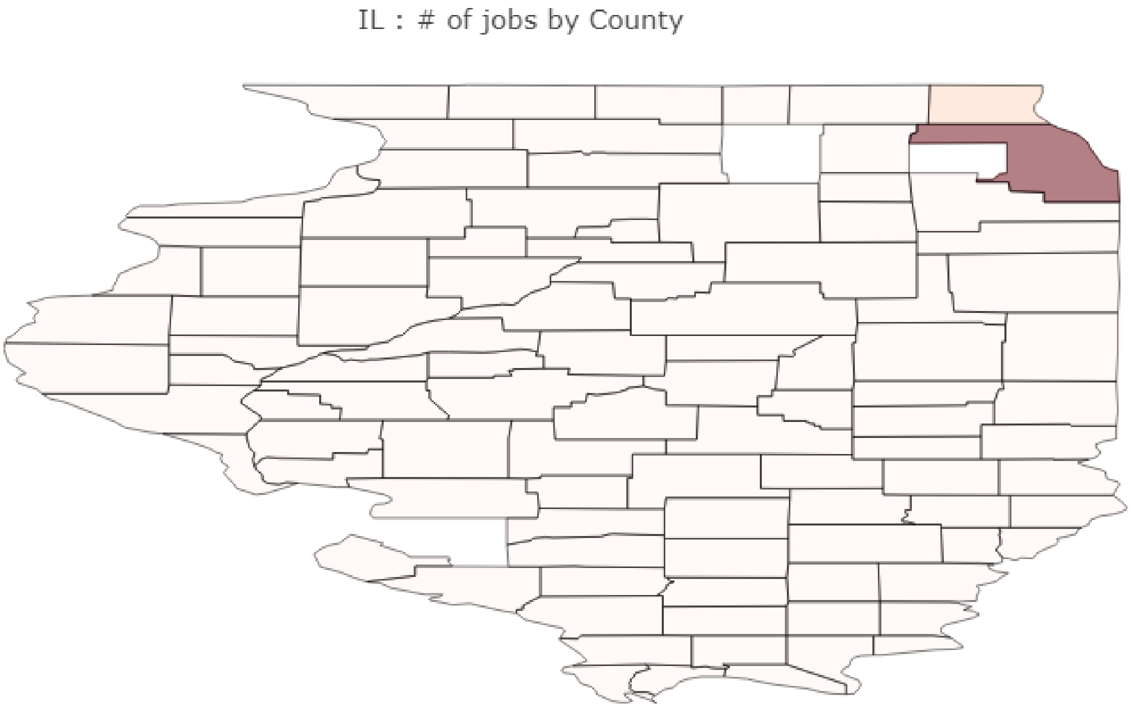


Figure 13:

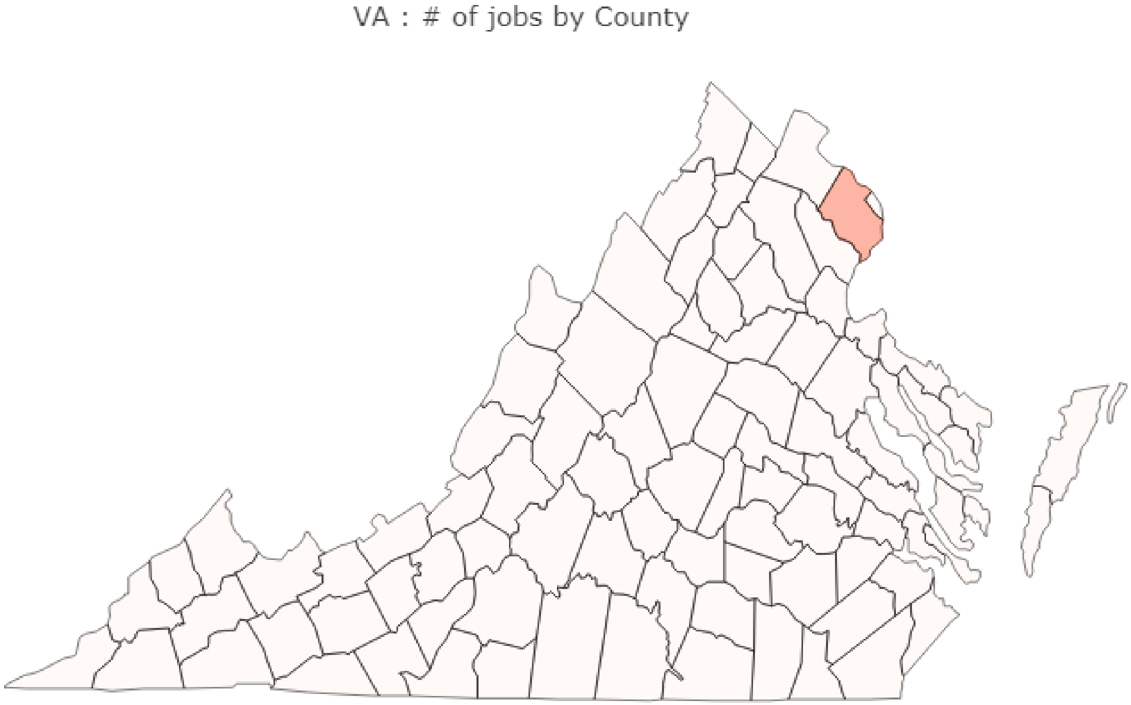


Figure 14:

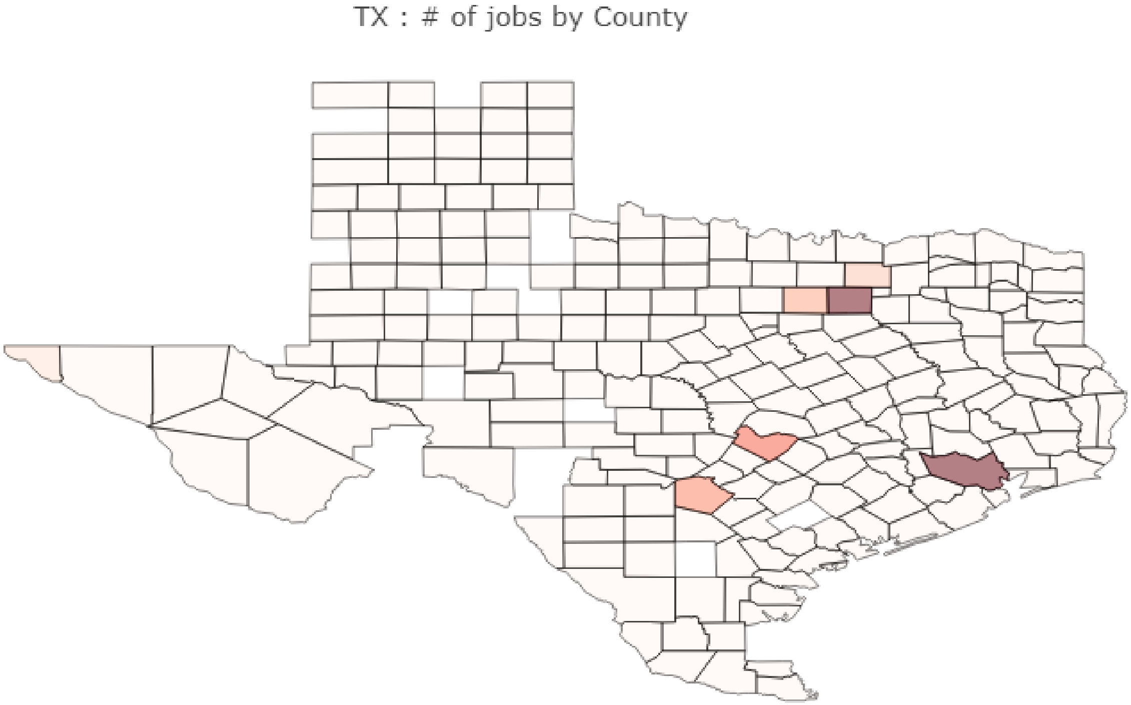


Figure 15:

Time will tell

Another focus of analysis was the variations among number of job postings with time. When one talks about time in the context of provided dataset, it could mean a few different things. And some of those things were analyzed and graphed in this project. For example, the trends of jobs postings with weekdays; whether most jobs are posted on Monday, or Saturday? The trend observed however, is not very interesting and can be guessed beforehand. The bars for Saturday and Sunday are lower as one might expect with Sunday being the loweest. Another significant low is seen for Mondays which can be attributed to Monday blues. Overall, this result doesn't provide with very enlightening news. So I will spare the reader from the trouble of having to look at it.

Same goes for the trends with dates for all months and across the months as well. For these results too, other than few mildly interesting discoeries, e.g. how the postings are generally higher in the beginning of the month and how the dataset contains most jobs from the final month of the period while December being the lowest, we don't get much revelations. What I have chosen to present instead is a graph where all this comparative behavior can be shown together, namely, day, date, week, and month.

The first graph below is a simple plotting of the job postings as the month goes by from 1st date to last. The 12 lines correspond to different months provided for in the dataset. And it seem pretty uniform and doesn't evince any pattern. One main reason for this is the timewise flow among months will only look alike if they consisted of same structure of weeks. Namely, if all months started on same day of the week and progressed, that should expectantly give a fair basis for comparison. So for the second graph, the months were shifted to align with first Monday of each and the end result is a beautiful recurrence of peaks and troughs indicating where the weeks started and ended. Not only that, we also see the aforementioned differences during a work week in number of postings as well as lows during middle of the month. Finally, the orange line for last month is an outlier and falls on top of others almost all days. The relavant code is included before the graphs.

```
#plotting number of posting with time in each month
temp<-as.data.frame(table(arfd$post_month,arfd$post_day))
colnames(temp) <- c("month","date","freq")
temp <- temp[with(temp,order(month,date)),]
#reordering with time (no pattern)
temp <- temp[c(342:372,311:341,280:310,63:93,125:155,94:124,
               218:248,1:31,249:279,187:217,156:186,32:62),]
plot_ly(x = 1:31, y = ~temp[1:31,]$freq, name="Sep 16",
        type = 'scatter', mode = 'lines') %>%
  add_trace(y = ~temp[32:62,]$freq, name="Oct 16",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[63:93,]$freq, name="Nov 16",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[94:124,]$freq, name="Dec 16",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[125:155,]$freq, name="Jan 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[156:186,]$freq, name="Feb 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[187:217,]$freq, name="Mar 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[218:248,]$freq, name="Apr 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[249:279,]$freq, name="May 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[280:310,]$freq, name="Jun 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[311:341,]$freq, name="July 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[342:372,]$freq, name="Aug 17",
            type='scatter', mode='lines') %>%
  layout(title = "Postings with time",
         xaxis = list(title = "Dates"),
         yaxis = list (title = "# of postings"))


# starting with Monday for each month and truncating in the end
plot_ly(x = 1:28, y = ~temp[c(5:31,4),]$freq, name="Sep 16",
        type = 'scatter', mode = 'lines') %>%
  add_trace(y = ~temp[c(34:61),]$freq, name="Oct 16",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[c(69:93,66:68),]$freq, name="Nov 16",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[c(98:124,97),]$freq, name="Dec 16",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[c(126:153),]$freq, name="Jan 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[c(161:186,159:160),]$freq, name="Feb 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[c(192:217,190:191),]$freq, name="Mar 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[c(220:247),]$freq, name="Apr 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[249:276,]$freq, name="May 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[c(284:310,283),]$freq, name="Jun 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[c(313:340),]$freq, name="July 17",
            type='scatter', mode='lines') %>%
  add_trace(y = ~temp[c(348:372,345:347),]$freq, name="Aug 17",
            type='scatter', mode='lines') %>%
  layout(title = "Postings with time \n (months shifted)",
         xaxis = list(title = "Dates"),
         yaxis = list (title = "# of postings"))
```

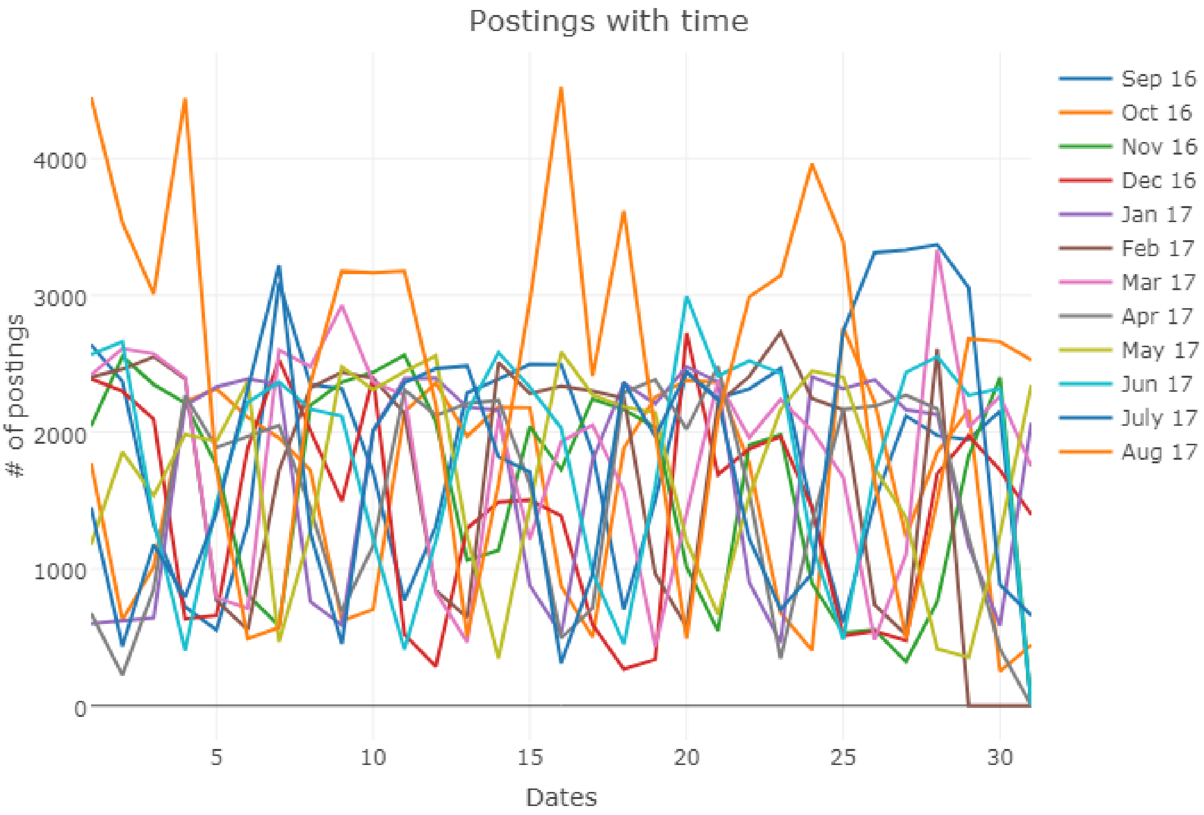

Postings with time
# of postings
0
1000
2000
3000
4000
Dates
5
10
15
20
25
30
Sep 16
Oct 16
Nov 16
Dec 16
Jan 17
Feb 17
Mar 17
Apr 17
May 17
Jun 17
July 17
Aug 17

Figure 16:

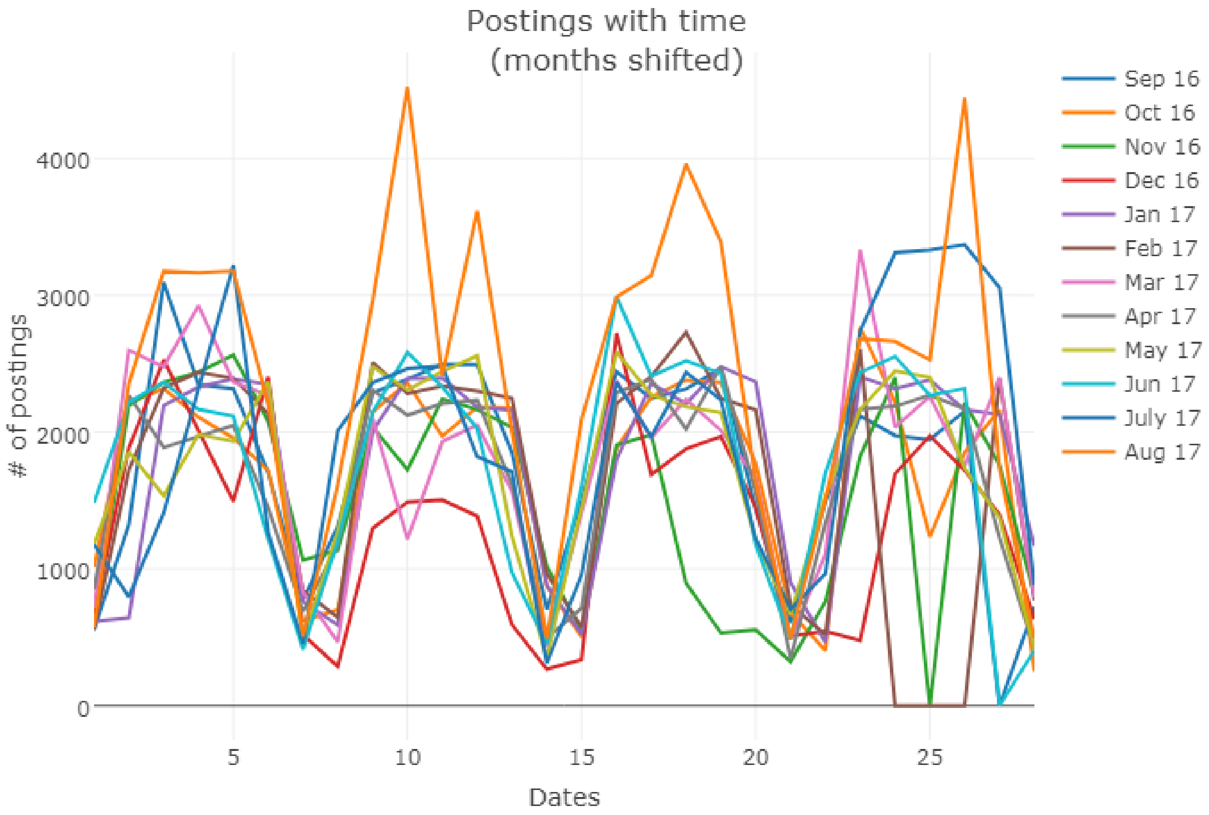

Postings with time
(months shifted)
# of postings
0
1000
2000
3000
4000
5
10
15
20
25
Dates
Sep 16
Oct 16
Nov 16
Dec 16
Jan 17
Feb 17
Mar 17
Apr 17
May 17
Jun 17
July 17
Aug 17

Figure 17:

Big boast(post) in bluk

It was observed that some of the companies (especially big ones) post most of their jobs in bulk, i.e. on few days during the year. This pattern was analyzed for top 200 companies and 71 companies out of them were found to post in bulk. The postings by a company on any specific day is defined as bulk here if the number of jobs posted account for more than 5% of the total jobs company has posted through the year. The code and graph indicating that is included below. Again, in the presence of an interactive visualization tool, dots can be hovered over to obtain the company name and total number of postings.

```r
#to see what percent of postings is in bulk by a company
#temp will contain total frequencies for top 200 companies
temp <- as.data.frame(table(arfd$company))
colnames(temp) <- c("company","total_freq")
temp <- temp[with(temp, order(-total_freq)), ]
temp <- temp[1:200,] #only keep top 200
#temp2 contains date was frequency for all companies
temp2 <- as.data.frame(table(arfd$company,arfd$post_date))
colnames(temp2) <- c("company","date","freq")
#selecting only the top 80, the ones which are in temp
temp2 <- temp2[which(temp2$company %in% temp$company),]
#merging for later purposes (basically to keep the total as well)
temp3 <- merge(temp2, temp, all.x = "TRUE")
#summing over the number of postings which are more than 5% of total
#postings on one day
temp3 <- summarize(group_by(temp3,company),
                   count=sum(freq[which(freq>(total_freq*0.05))]))
colnames(temp3) <- c("company","bulk_freq")
#merge again to put total frequency side by side
temp3 <- merge(temp3, temp, all.x = "TRUE")
#calculate percent bulk
temp3$percent_bulk = temp3$bulk_freq*100/temp3$total_freq
#ordering with respect to total frequency
temp3 <- temp3[with(temp3,order(-total_freq)),]
temp3 <- temp3[-which(temp3$bulk_freq==0),]
#plot
plot_ly(data = temp3, x = ~(1:length(temp3$company)),
             y = ~temp3$percent_bulk,
             text = ~paste(temp3$company,
                           "\n Total postings:",temp3$total_freq),
             marker = list(size = 10,
                           color = 'rgba(255, 182, 193, .9)',
                           line = list(color = 'rgba(152, 0, 0, .8)',
                                       width = 2))) %>%
      layout(title = paste('Companies postings in bulk',
                          '\n (among top 200 companies)'),
         yaxis = list(title="Bulk posting percent",
                      zeroline = FALSE),
         xaxis = list(title="Company size (decreasing)",
                      zeroline = FALSE))
```

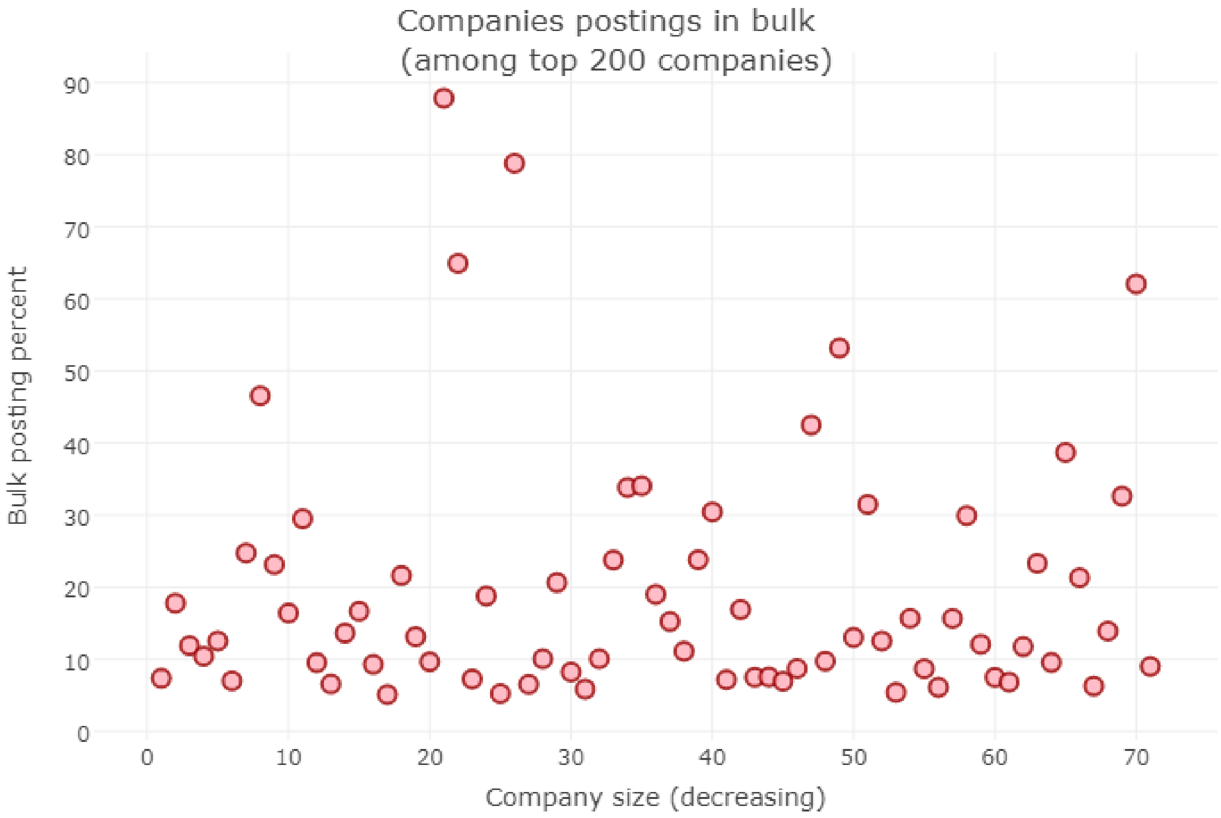

Companies postings in bulk
(among top 200 companies)
Bulk posting percent
0
10
20
30
40
50
60
70
80
90
Company size (decreasing)
0
10
20
30
40
50
60
70

Figure 18: