Healthcare Job Market Data Analysis

Data Analytics Research Lab, Rensselaer Polytechnic Institute
Team: Abhishek Choudhary, Wilson Gregory, Zuohao Lai, Huan Li

Abhishek Choudhary

December 4, 2017

Contents

1 Introduction

This R Markdown Notebook contains the comprehensive summary of exploration and analysis of job market dataset in healthcare as accomplished by the author working in the team of 4 as part of RPI's Data Analytics course project. In conjunction with other 3 notebooks supplied by rest of the team, this notebook serves as a complete account of goals accomplished through this group project.

The outline of this notebook is as follows: In the next section, I list the questions our team intend to answer via this dataset, i.e. what information can be obtained regarding employment statistics. In section 3, I describe the data provided for the project and detail the operations performed on the data to clean and integrate it. Sections 4 to 9 provide a detailed overview of the exploration that has been done. Section 10 discusses the deficiencies in the provided data which prevents a more thorough analysis. There we also point out the measures in data collection that should be taken to overcome these limitations. The last section 11 lists the key results and allocates the group feats to individual member.

2 Objectives

Using the provided dataset which lists the details of at least a few hundred 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 large. 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

3 Dataset

The original dataset consists of 4 excel files with total size in the order of a few 100 MBs. In the first subsection below, I will describe the contents and purpose of each file. In the next subsection I will provide the details of operations performed on the dataset.

3.1 Description

1. `master.xls` This is the main file which contains all the data in one place. It consists 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), vertical (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:

```
##         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 digit alphanumeric code format), job id (format large integer), and skill (format string) respectively. A close look at the first few rows of this file:

```
## # A tibble: 3 × 3
##   source                          job_id      skill
```

```
##    <chr>                              <chr>      <chr>
## 1  A0004  72742024245081264314561480484 30   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:

```
## # A tibble: 3 × 2
##                              job_id  role
##                               <chr> <chr>
## 1  64763198904817550951283276118 95    RN
## 2  22932474811986571801906346499 39    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:

```
## # 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>
```

3.1.1 A Note on Missing Entries

The data is not complete. This may not be an unusual situation 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.

If unfilled jobs are classified based on the month they are posted in and a frequency table is computed, the output is following:

```
#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 three months period preceding 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 posted in December 16 or February 17 than March 17.

3.2 Operations on the Dataset

In order to answer the questions listed in previous section efficiently, we perform some cleaning and merging operations on the data described below.

3.2.1 Collection

The dataset provided 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 work 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 in Appendix 12.0.2.1 illustrates this.

Now tables all_data, role_data, filter_data, and role_H_data have all the rows (in the order of hundreds of thousands) from master file, role file, filter file and role hierarchy file respectively.

3.2.2 Integration

The tables that we have in the dataset represent different 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 separately and do the analysis by referring to two or more of them anytime outside information is needed but there is one problem with that.

The problem 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 always greater than one and they too are not found in one place in the filter table. So as part of cleaning up and aggregating all the information, it becomes essential to put all the data related to an individual job at one place without the need to parse thousands 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 in 12.0.2.2.

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.

3.2.3 Classification

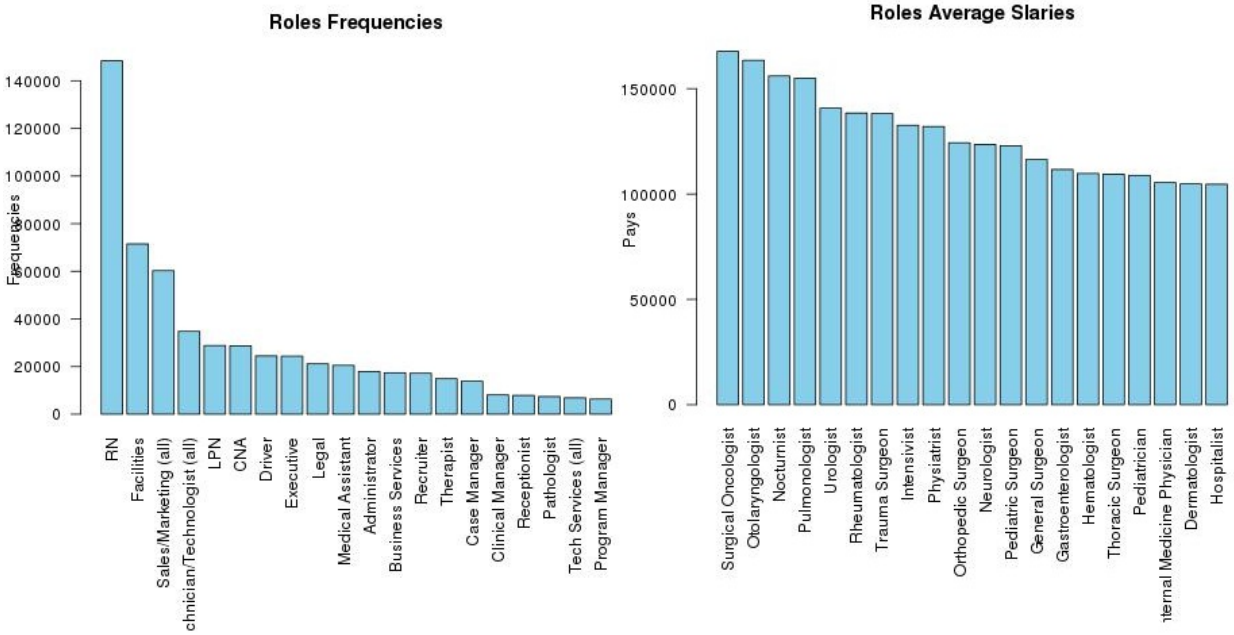
One important classifier for the job database (and this is true in general to some extent) is the state in which jobs are located. Since we have data from four states (VA,TX,FL,IL), it's only natural to extract the jobs from all_data pertaining to an individual state. extractState function shown in 12.0.2.3 does that.

4 Jobs Frequency and Salary Profiles

In the code pasted above, we have already obtained the unique list of roles (and skills) from the complete dataset. Now the natural question arises: which kind of jobs is most frequent and most valuable? To define value, we choose the most natural indicator salary as variable to present the gains of different roles and skills. We do a frequency and salary analysis of job postings based on roles and skills.

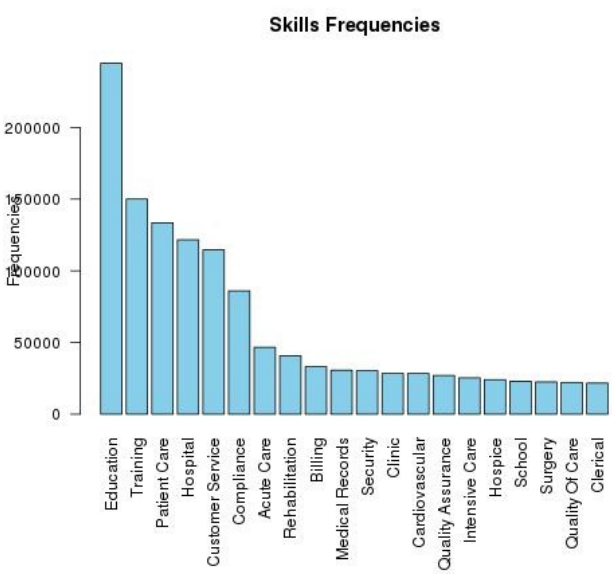
4.1 By Roles

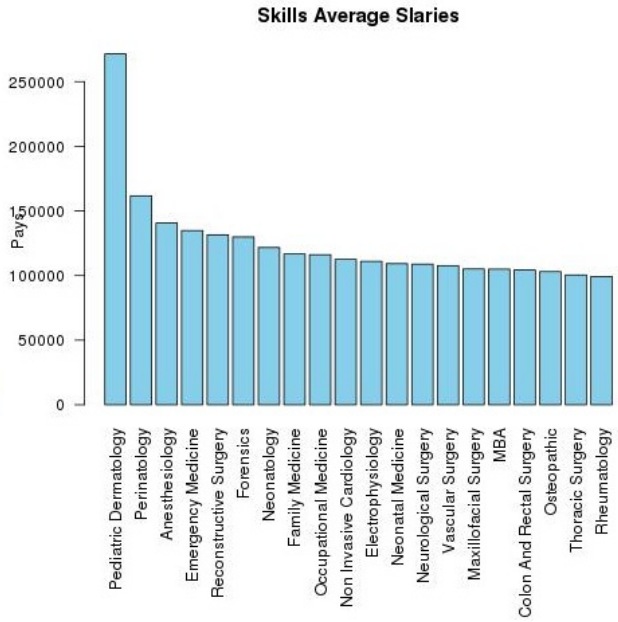
That can be answered by plotting the frequencies of different roles on x-y axis. Note that for this purpose, we directly use the role table and count the occurrence of each category. For representational elegance, we only focus on 20 most frequent roles but code can always be modified to include more. We see that RN (registered nurses) is far more common than any other role. Through the salary graph, we observe that people in actual patient care earn more than affiliated hospital staff.



4.2 By Skills

To see which are the most frequent skills among healthcare professionals, we do a similar probing of filter table and obtain the answers. Here we see education is the most popular skill. A bar plot of 20 largest income inducing skills is also presented below. For some inexplicable reason, pediatric dermatology pays much higher than any other job.





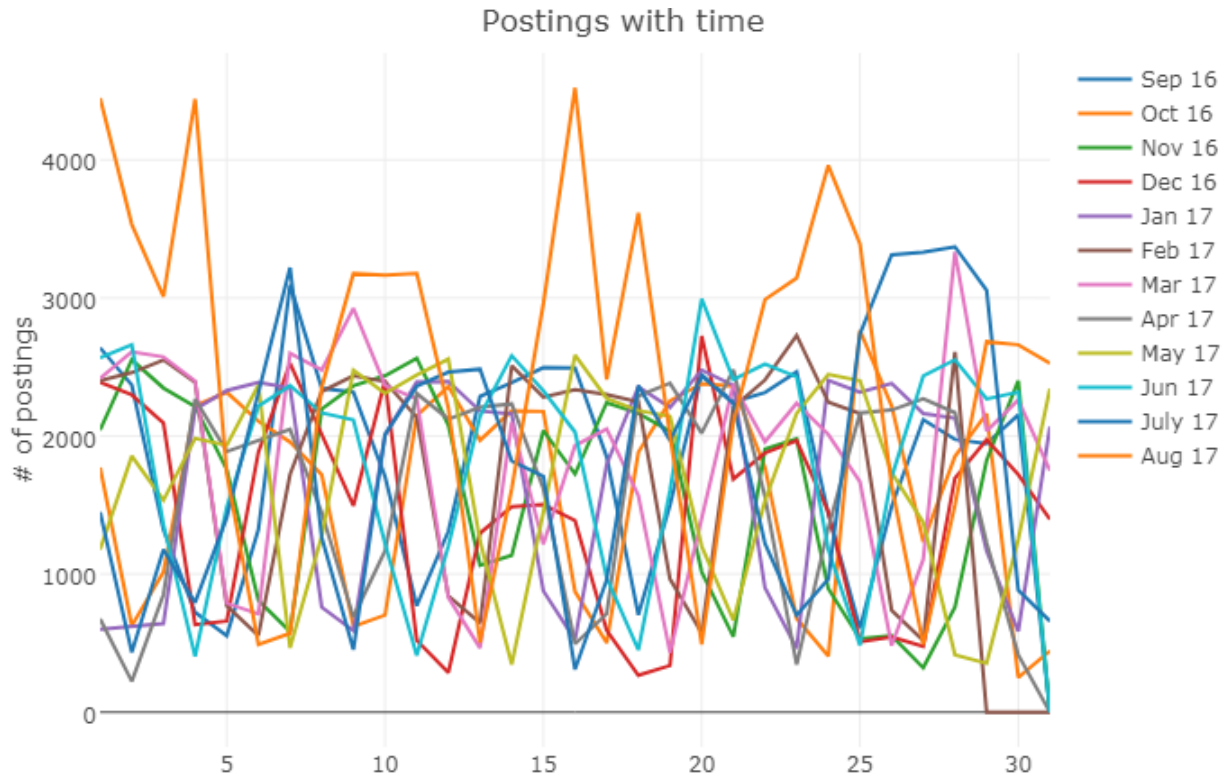
5 Variations with Time

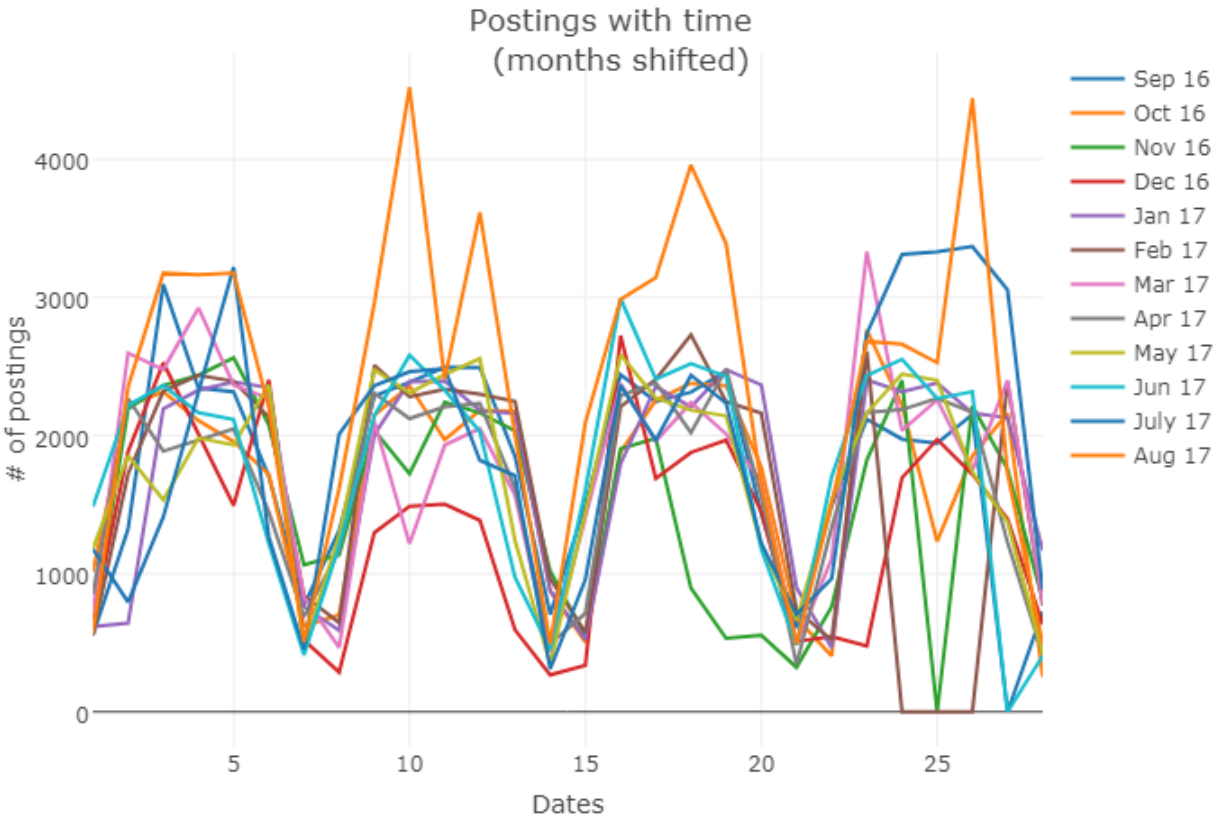
Another focus of analysis was the variations among number of job postings with time. To do that, we first create the necessary columns, by breaking down the post_date for the job into year, month and day of the week as illustrated in 12.0.2.6. The day is fetched just to see (wil be done later) *when* during the week most jobs are posted. After this processing arfd 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. We then look at the trends.

5.1 Trends: Job Postings

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 shows the bars for Saturday and Sunday are lower as one might expect with Sunday being the lowest. Another significant low is seen for Mondays which can be attributed to Monday blues. Same goes for the trends with dates for all months and across the months as well. There, we see the expected behavior that the postings are generally higher in the beginning of the month and the dataset contains most jobs from the final month of the period while December being the lowest. Here I present a graph where all this comparative behavior can be shown together, namely, day, date, week, and month.

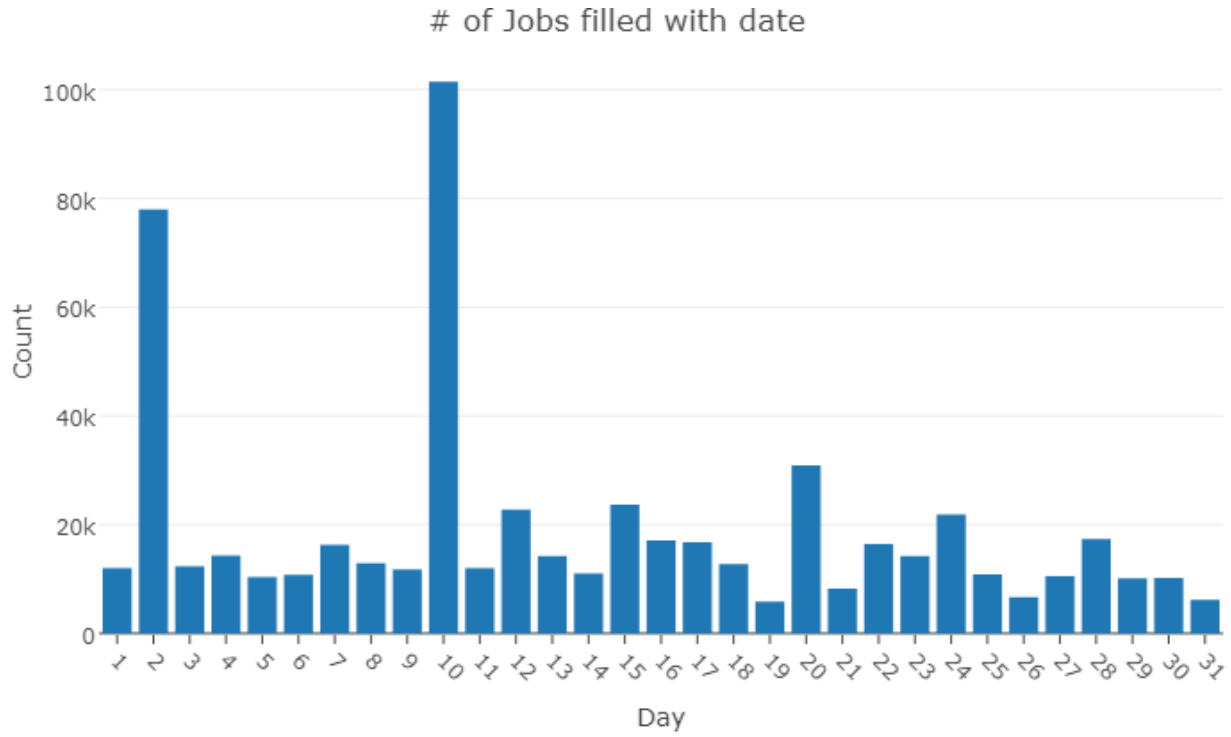
The first graph below is a simple plotting of the job postings as the month goes by from 1st date to last. The 12 lines correspond to different months provided for in the dataset. And it seems pretty uniform and doesn't evince any pattern. One main reason for this is the time-wise flow among months will 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 relevant code is included in 12.0.2.7.

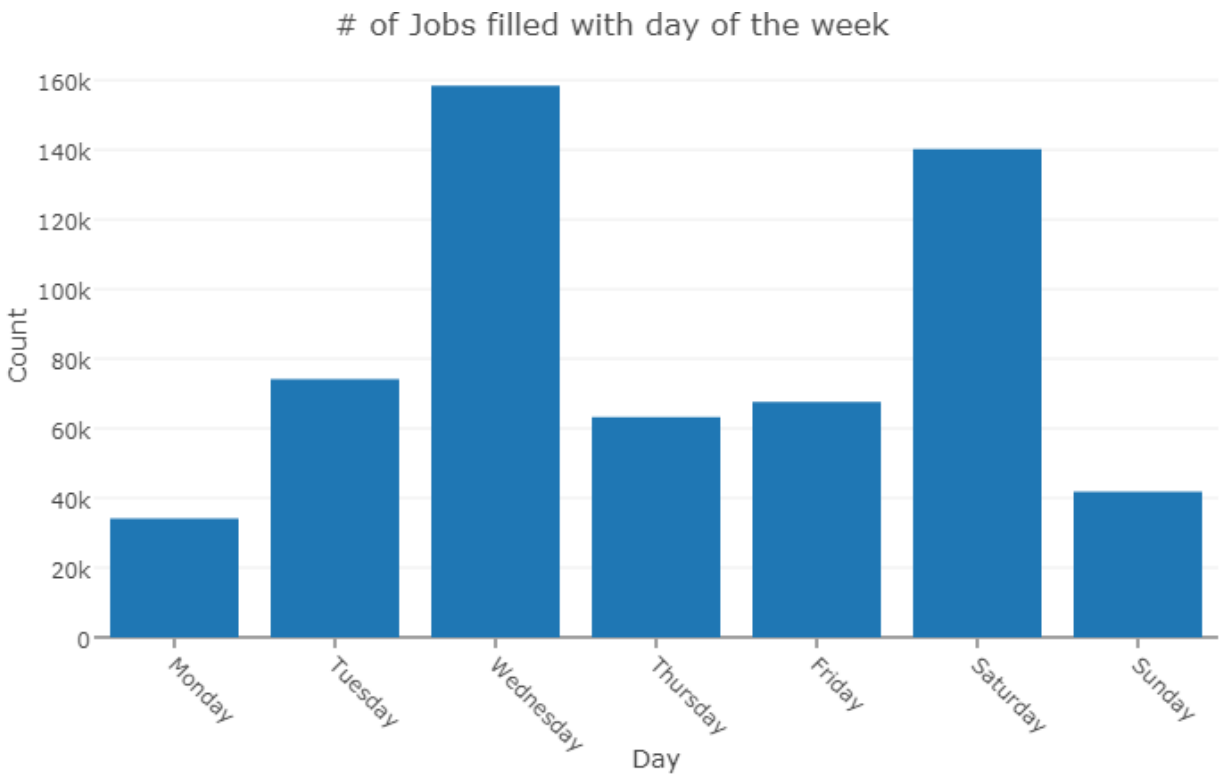

Postings with time
# of postings
4000
3000
2000
1000
0
5
10
15
20
25
30
Sep 16
Oct 16
Nov 16
Dec 16
Jan 17
Feb 17
Mar 17
Apr 17
May 17
Jun 17
July 17
Aug 17


Postings with time
(months shifted)
# of postings
4000
3000
2000
1000
0
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

5.2 Trends: Job Fillings

We do the same sort of analysis with the job fill_dates and observe that there are a few surprises with respect to filling time behavior of jobs. Namely, we see that most jobs are filled on 2nd and 10th of the month as they evince powerful maxima. During a week, most jobs are filled on Wednesday and Saturday. While we expect the heavy fillings at the start of the month, the other date is a curious case. The high on Wednesday also poses a surprise as to why are such a large number of jobs getting filled during the middle of the week. The code for this analysis is in 12.0.2.8.



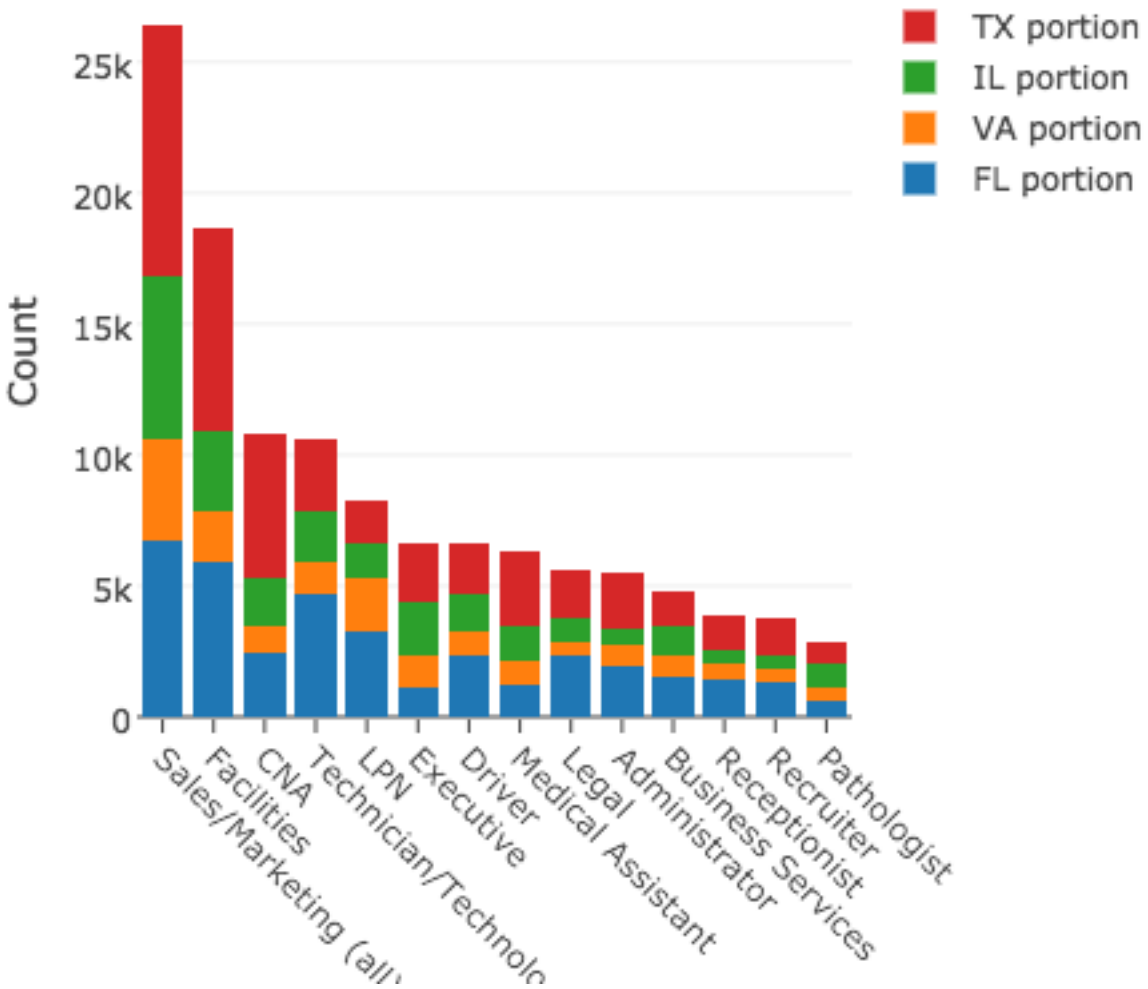


6 Region Based Differences

After ranking the frequencies of different roles in each state, it was observed that the order of popularity, either in terms of postings or salary, isn't same in these states even for the top roles. We present some relevant results indicated state-wise differences.

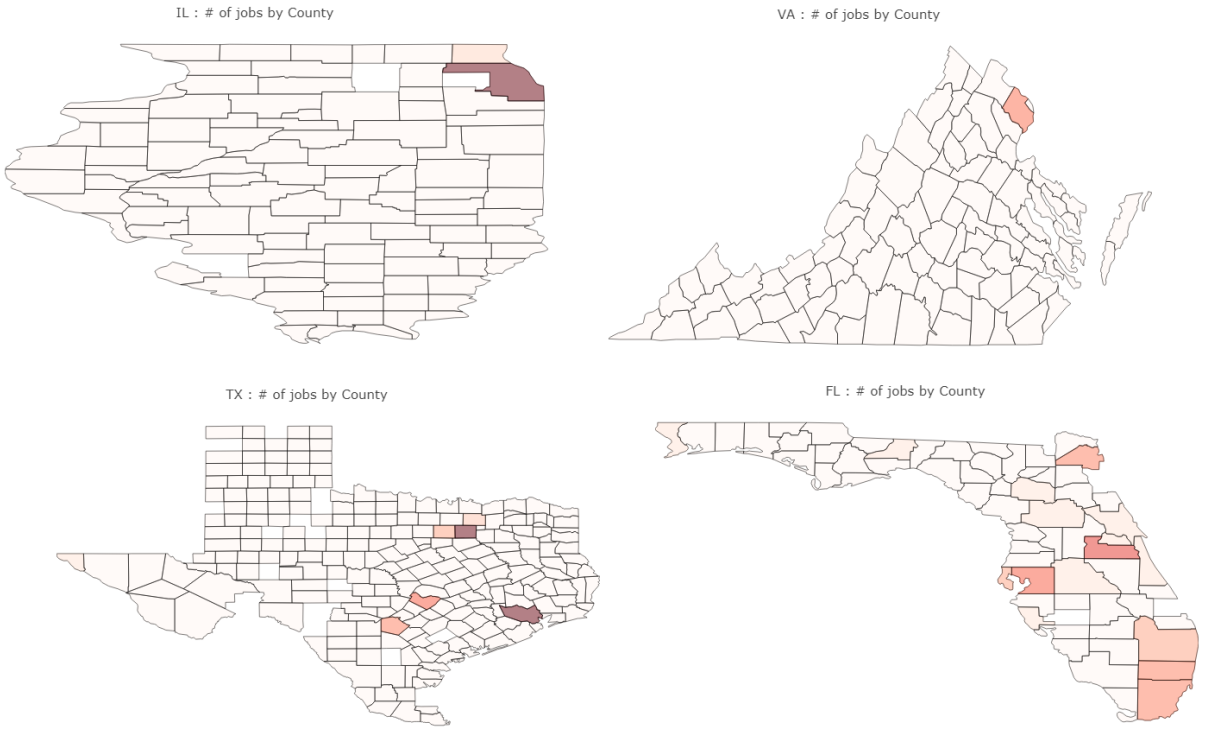
6.1 Frequency: Contribution from States

For a relative comparison among states, the code below generates the graph for overall most frequent roles and contribution from each state in this frequency. What is observed is that the individual counts from these states aren'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 non-uniform across different locations due to variety of sources, or the demand for different 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.



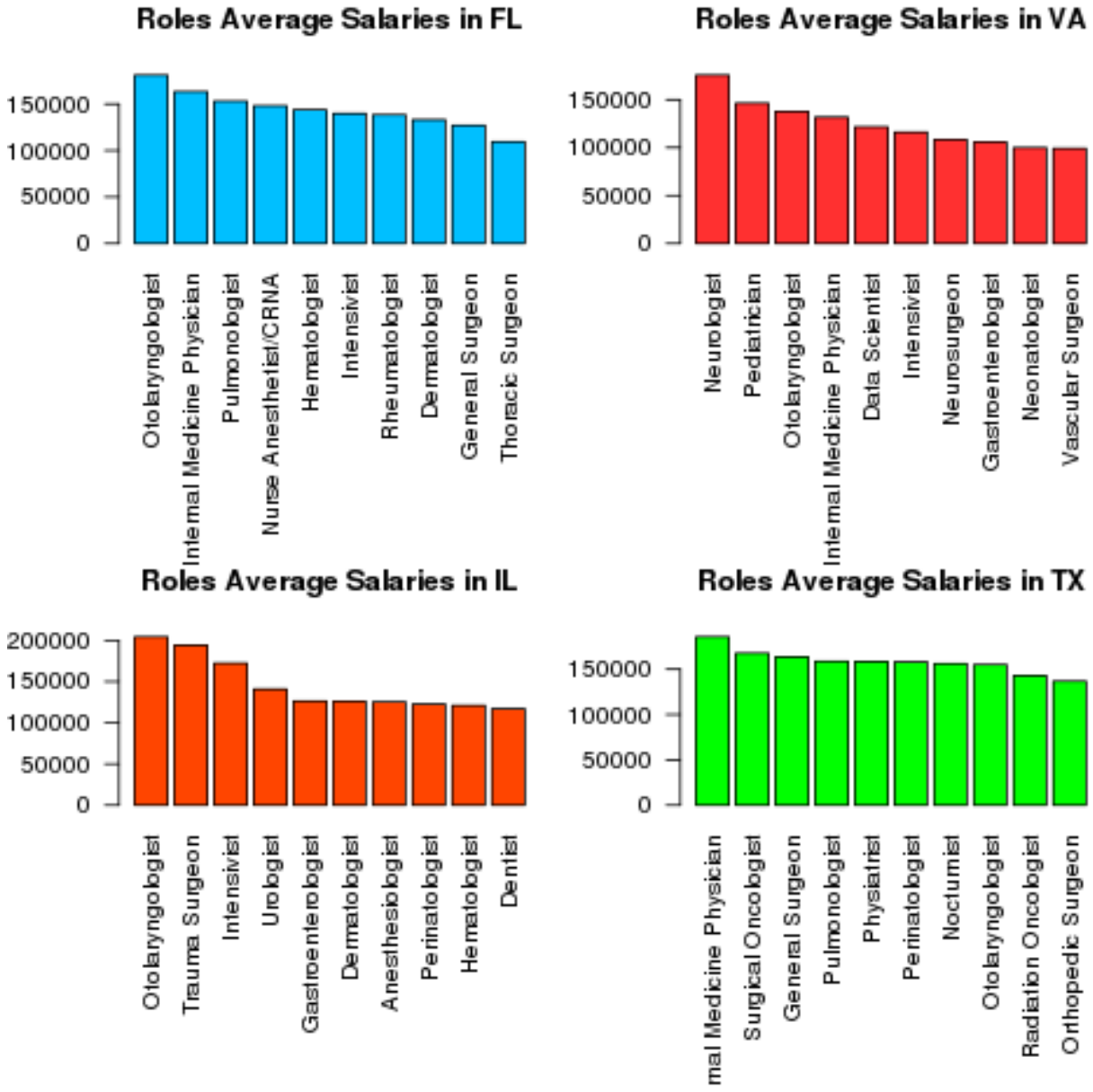
6.2 Frequency: Contribution within States

Here we represent distribution of job postings 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.



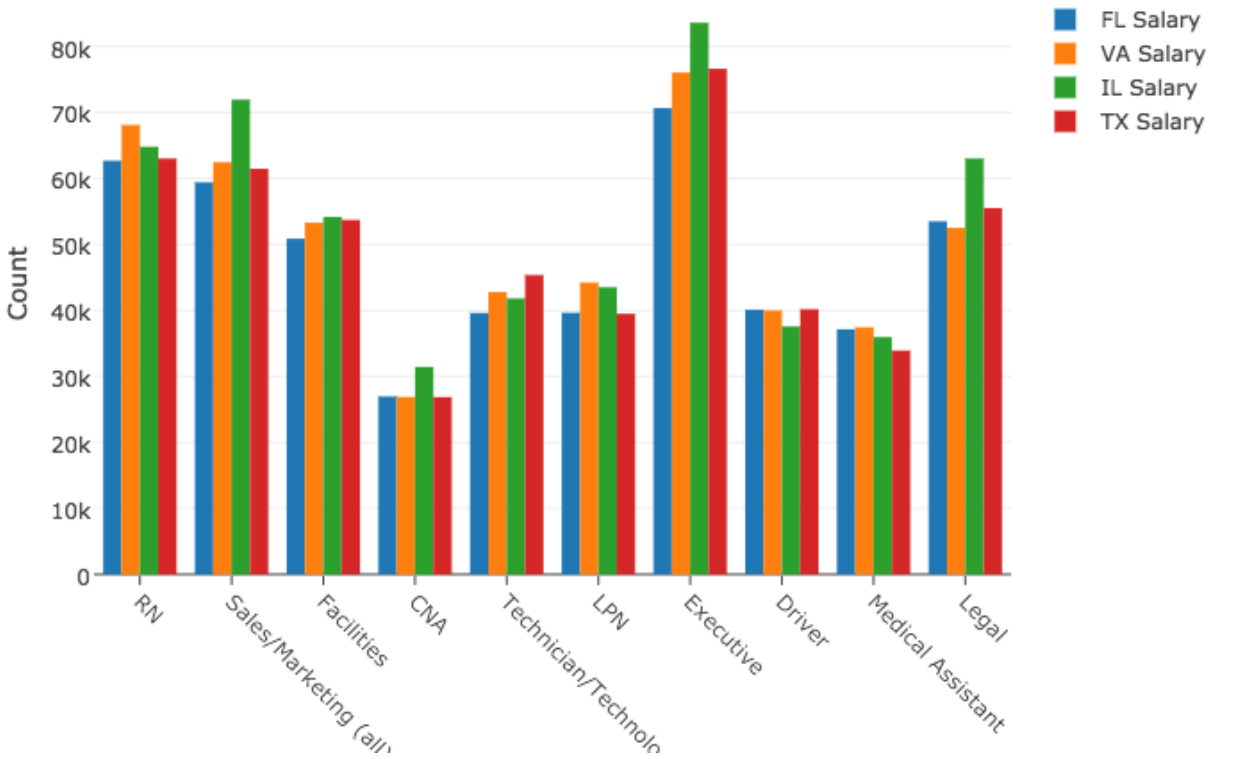
6.3 Salary: Top Roles within States

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 for this 12.0.2.11 should be understood in conjunction with code from earlier sections.



6.4 Salary: Comparison among States

The state-wise 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 state-wise 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, given in 12.0.2.12, could be applied to rank all the roles down the frequency list to observe the state-wise differences.

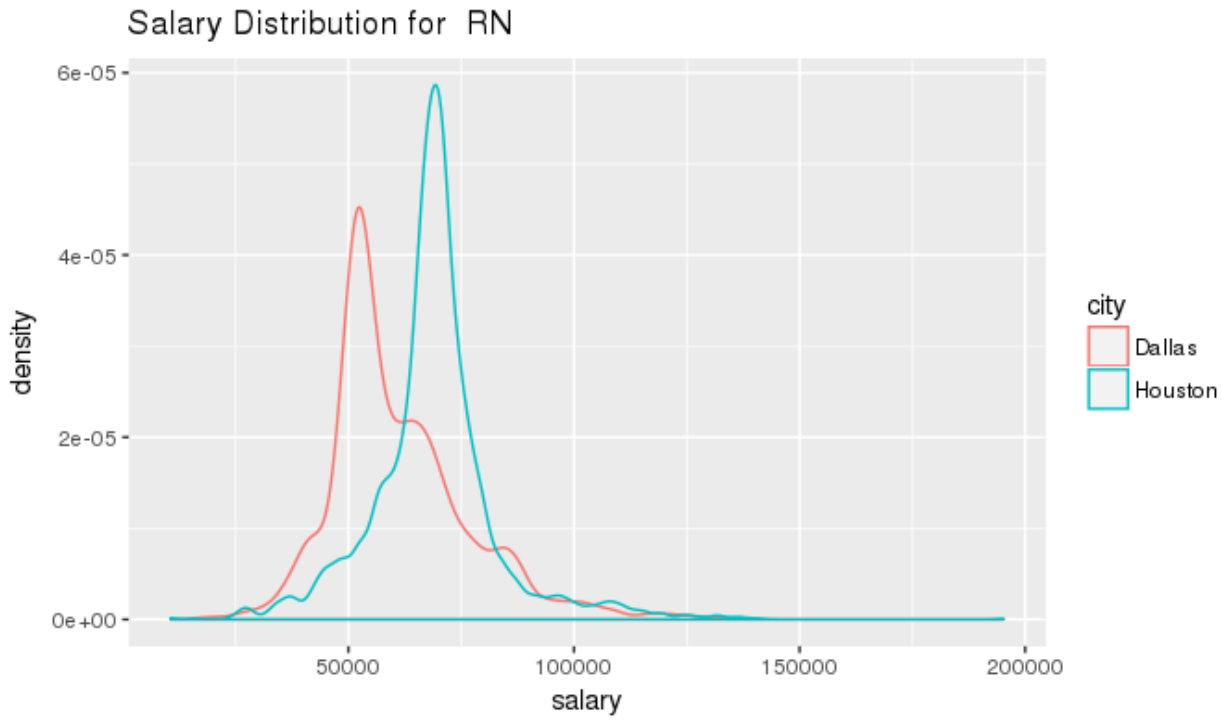


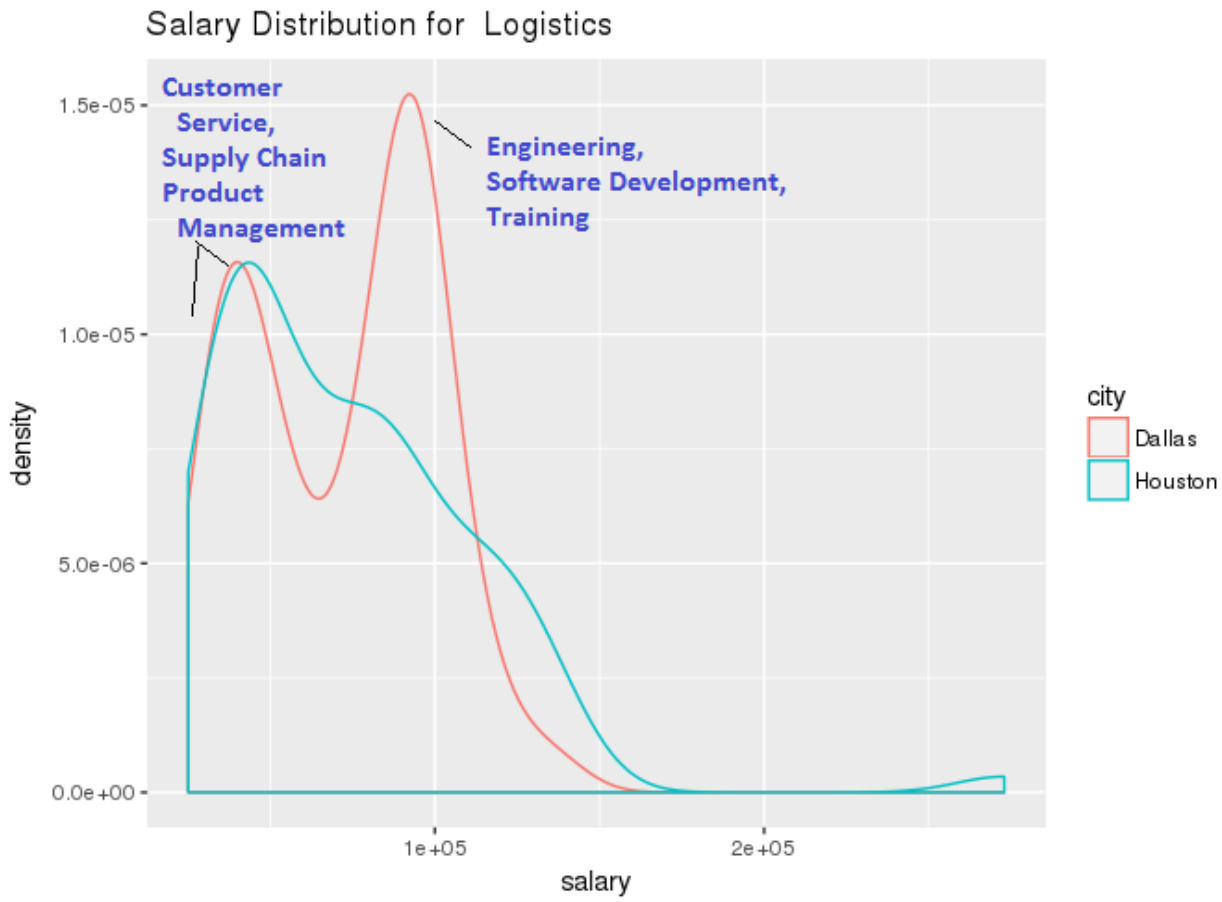
6.5 Salary: Comparison among Cities

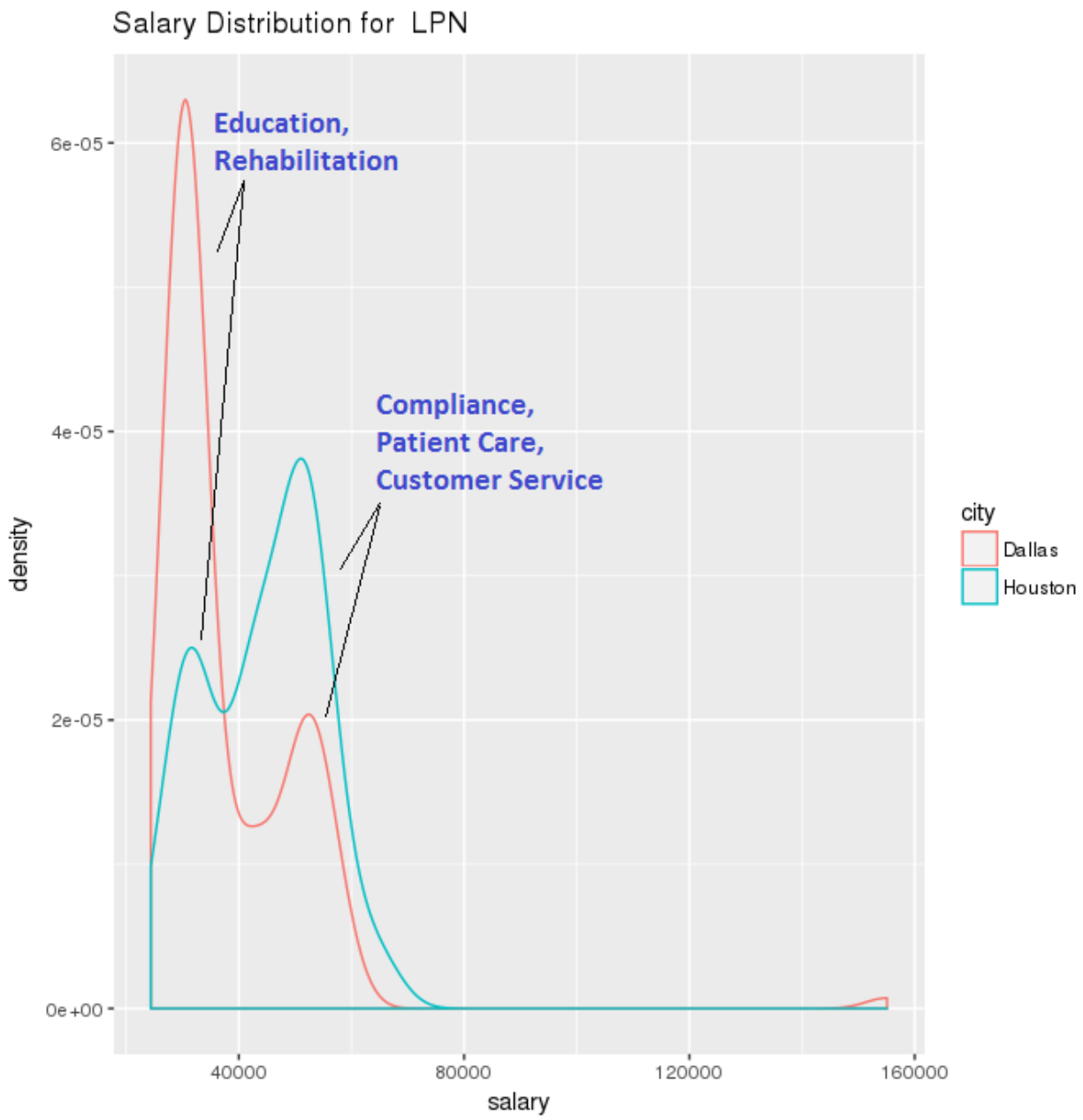
For the reasons described in the limitations in section 10, any analysis done within a city or small location, which attempts to distinguish jobs based on roles and skills attributes of job postings will inevitably lead to incorrect results. Therefore, we maintain our focus on a coarser level and compare the salary behavior between any two given cities. For this purpose, we fix the roles and look at the salary distribution in some of the biggest cities (based on # of postings) within a state. Fixing the unique role and considering just one city constrains the location, removes within roles differences, and only considers the salary variation for a given role. Here our focus will not be on the heights of distribution curves (which represent frequency) but rather their shape and relative locations w.r.t. x axis representing salary. The explanation of the dissimilar behavior based on skills is done manually and should only be taken in a broad sense. Code for this section is in appendix 12.0.2.13. A few such examples follow.

6.5.0.1 Houston vs. Dallas

Houston and Dallas are two biggest cities in Texas (based on # of job postings). We plot salary distributions for certain roles and present here some curious cases. A few observations are: 1) Salary of RN is higher in Houston. 2) In Logistics, a good number of jobs in Dallas pay high salary which require skills such as Engineering, Software Development and Training etc. 3) In LPN, unlike Dallas, Houston has more higher paying jobs which require skills like Compliance, Patient Care and Customer Service compared to low paying jobs mainly listing Education and Rehabilitation.

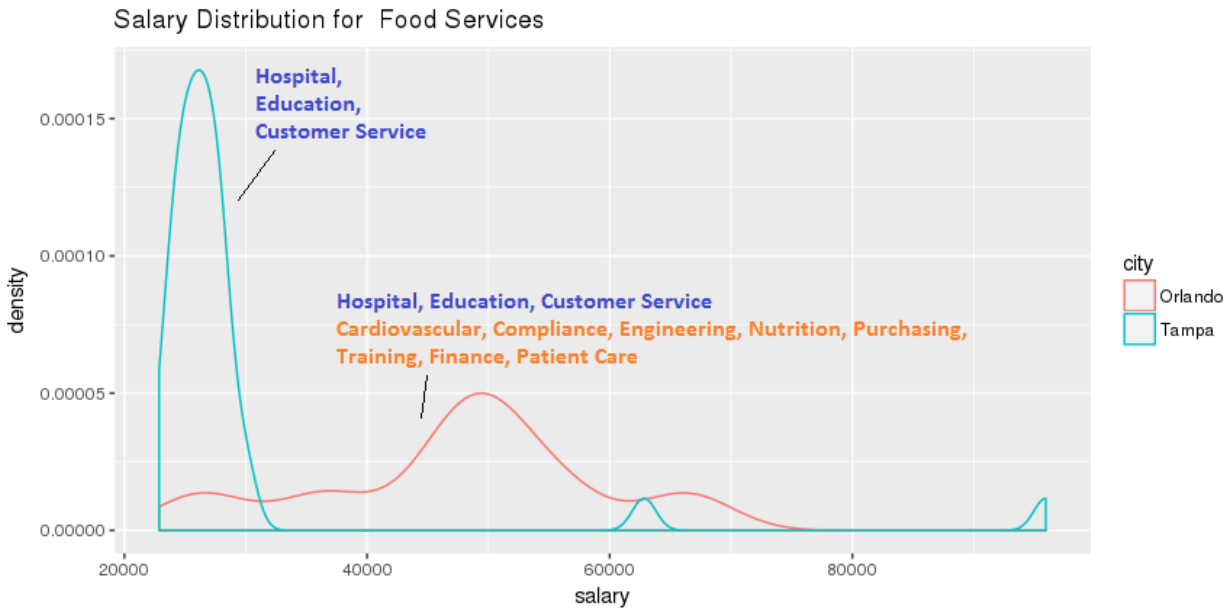

Salary Distribution for RN
6e-05
4e-05
2e-05
0e+00
density
50000
100000
150000
200000
salary
city
Dallas
Houston

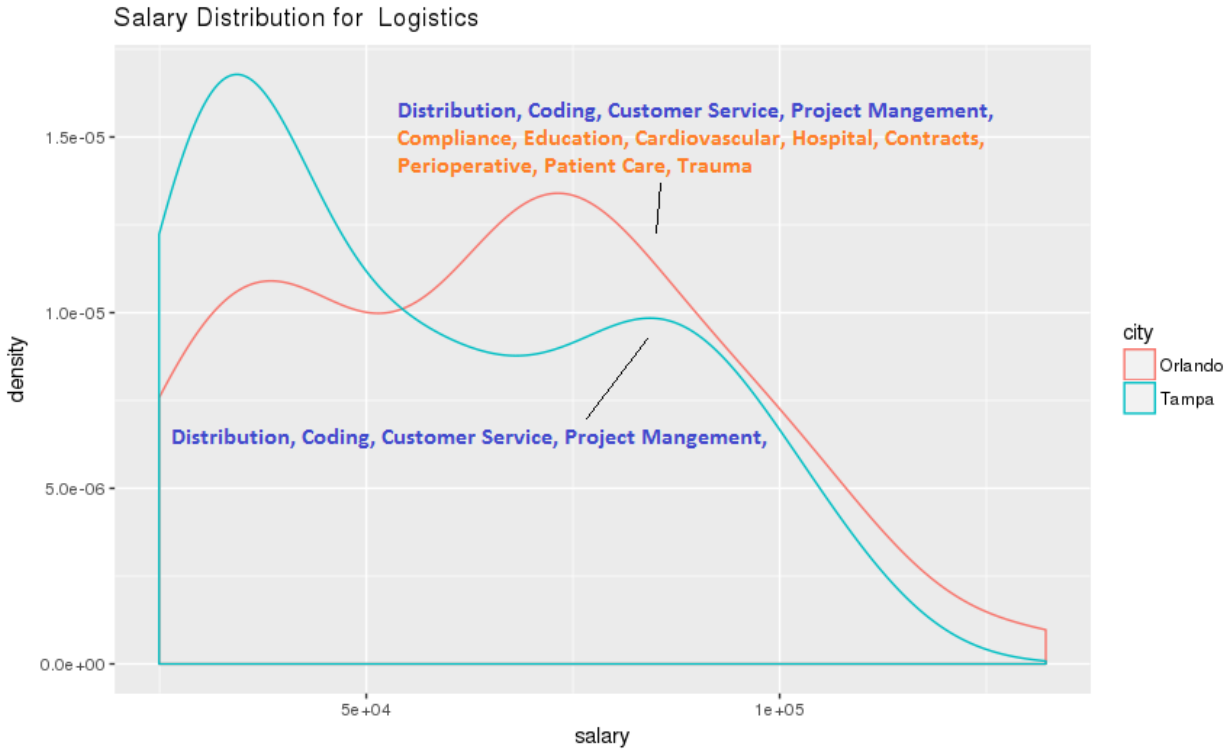

Salary Distribution for Logistics
Customer Service, Supply Chain Product Management
Engineering, Software Development, Training
1.5e-05
1.0e-05
5.0e-06
0.0e+00
density
1e+05
2e+05
salary
city
Dallas
Houston

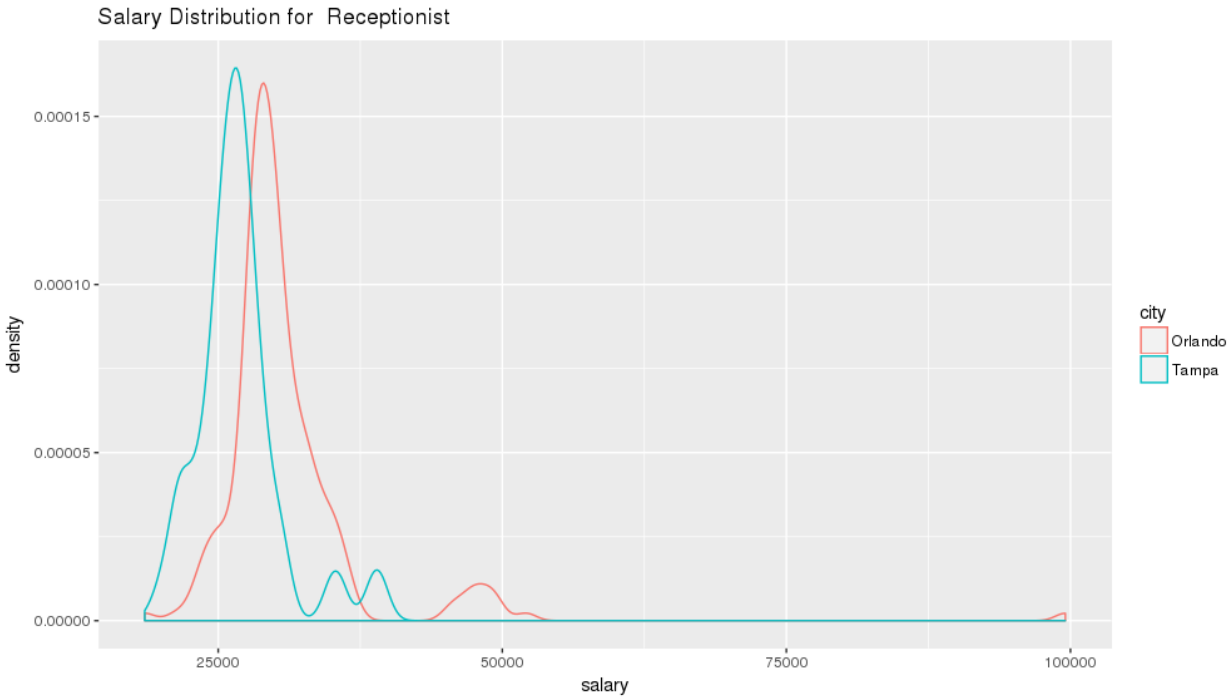


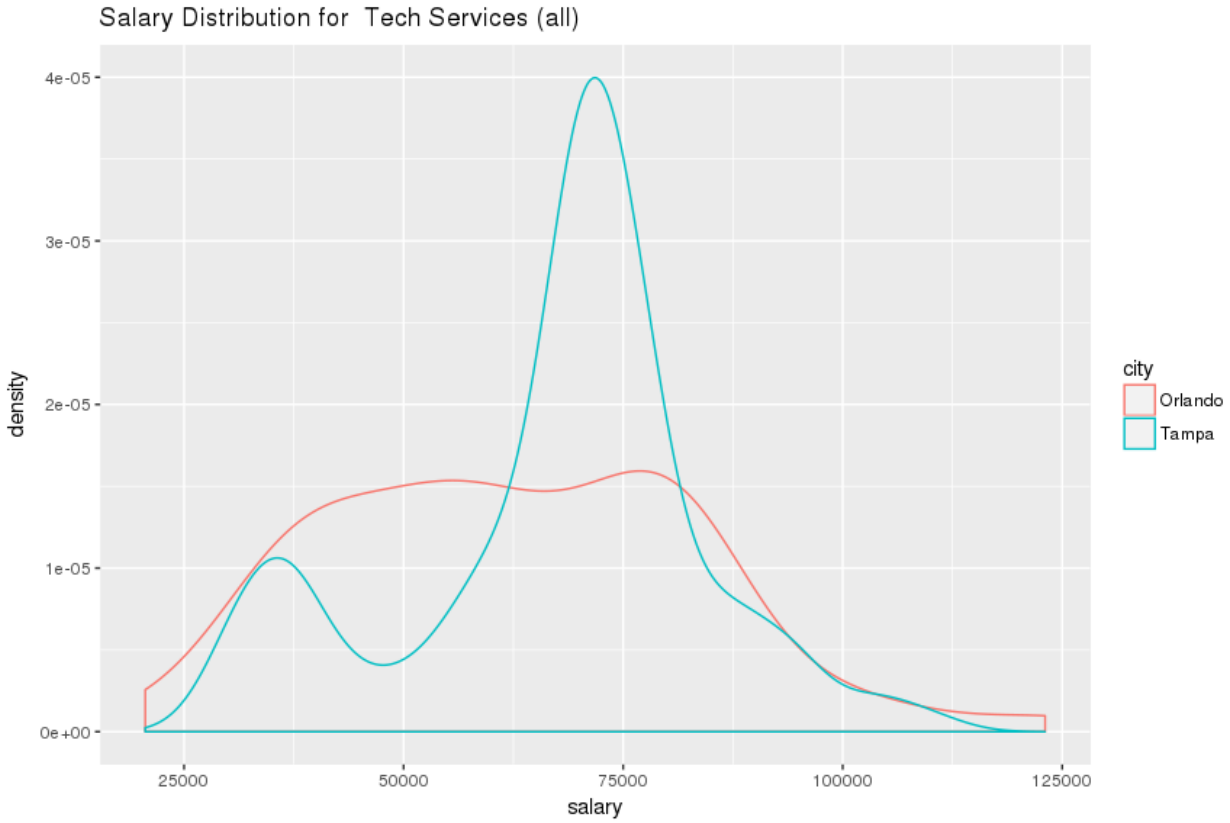
6.5.0.2 Tampa vs. Orlando

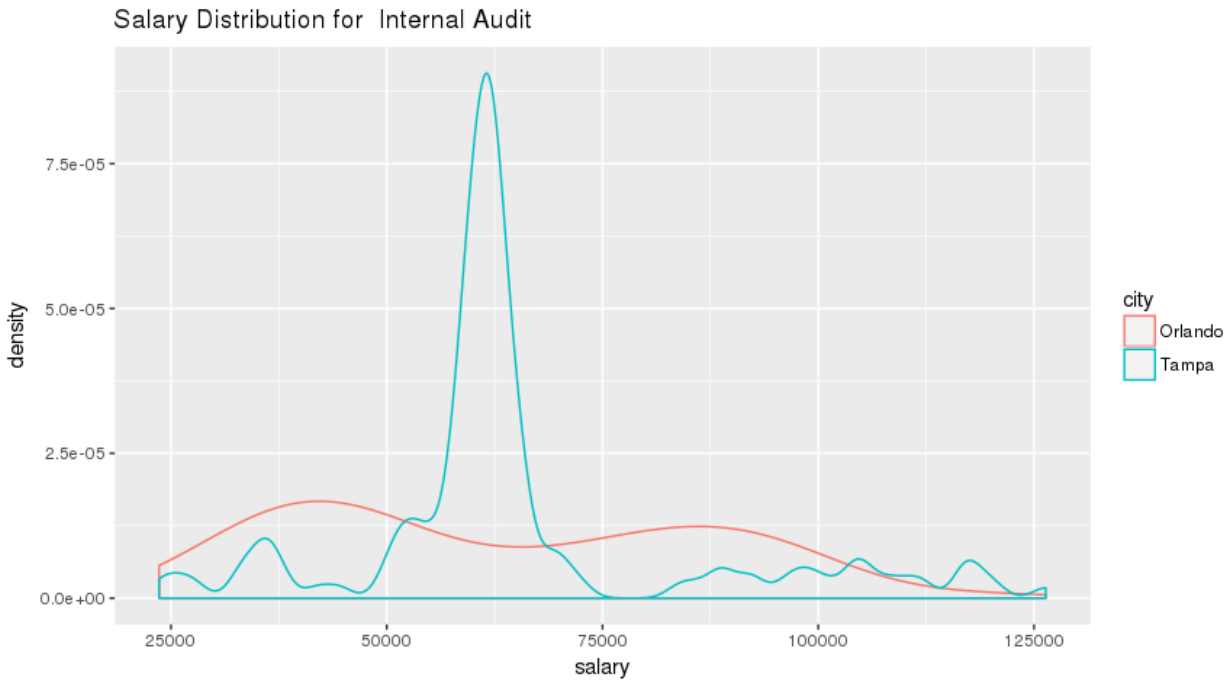
These are two biggest cities of Florida based on number of jobs. A few observations from some interesting plots below are: 1) In Food Services and Logistics, Orlando has a good number of higher paying jobs which list extra skills. 2) Salary of Receptionist is slightly higher in Orlando even though the average salary in general is almost equal in the two cities. 3) In Tech Services and Internal Audit, Tampa has almost 3 times as many roles as Orlando mostly localized in the median salary region.


Salary Distribution for Food Services
Hospital,
Education,
Customer Service
Hospital, Education, Customer Service
Cardiovascular, Compliance, Engineering, Nutrition, Purchasing,
Training, Finance, Patient Care
density
0.00015
0.00010
0.00005
0.00000
20000
40000
60000
80000
salary
city
Orlando
Tampa


Salary Distribution for Logistics
Distribution, Coding, Customer Service, Project Mangement,
Compliance, Education, Cardiovascular, Hospital, Contracts,
Perioperative, Patient Care, Trauma
Distribution, Coding, Customer Service, Project Mangement,
density
1.5e-05
1.0e-05
5.0e-06
0.0e+00
5e+04
1e+05
salary
city
Orlando
Tampa

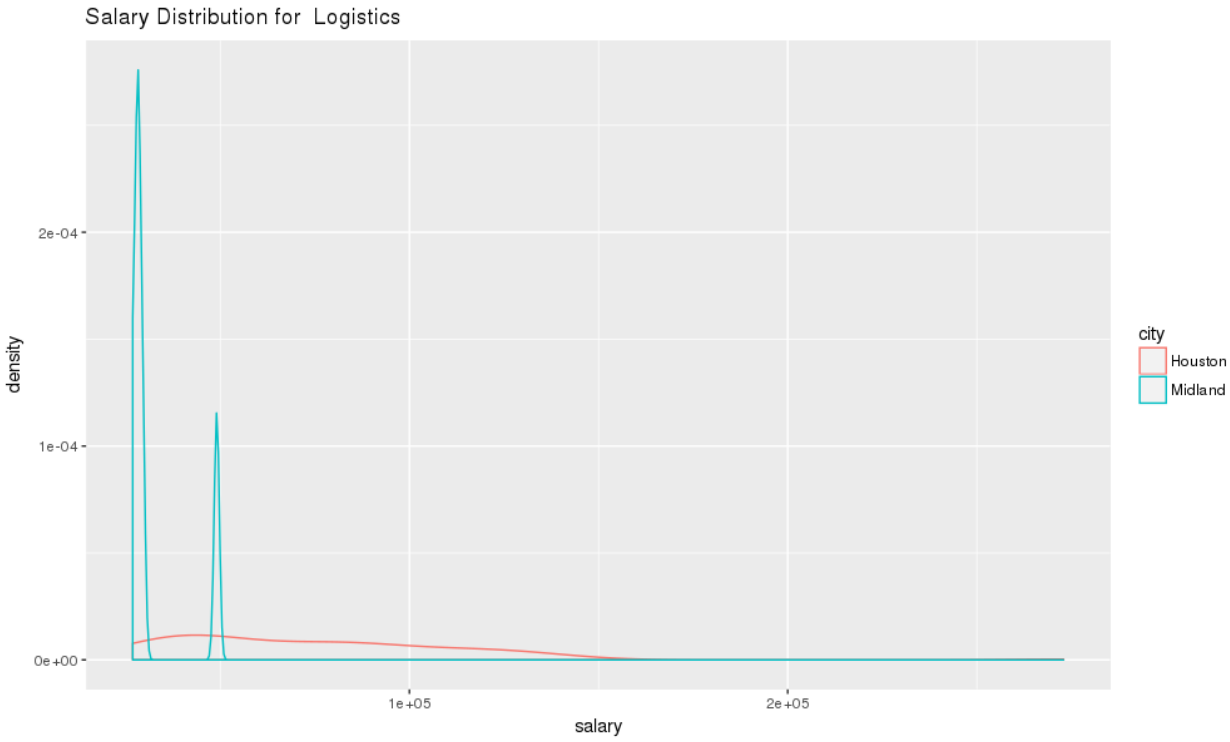

Salary Distribution for Receptionist
density
0.00015
0.00010
0.00005
0.00000
25000
50000
75000
100000
salary
city
Orlando
Tampa

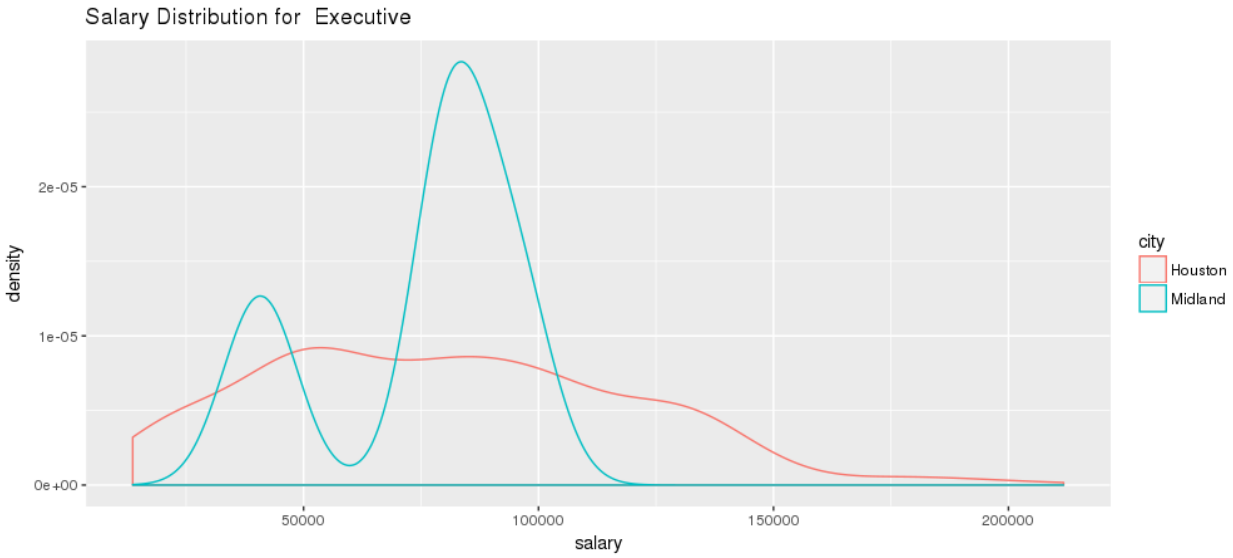

Salary Distribution for Tech Services (all)
density
4e-05
3e-05
2e-05
1e-05
0e+00
25000
50000
75000
100000
125000
salary
city
Orlando
Tampa

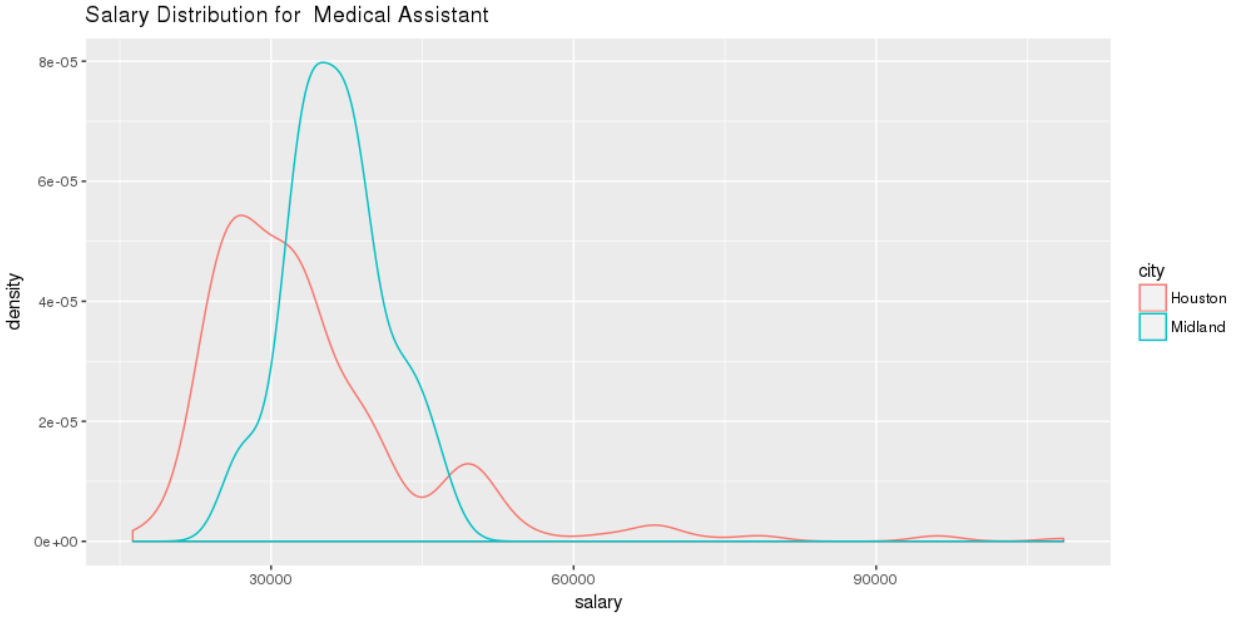


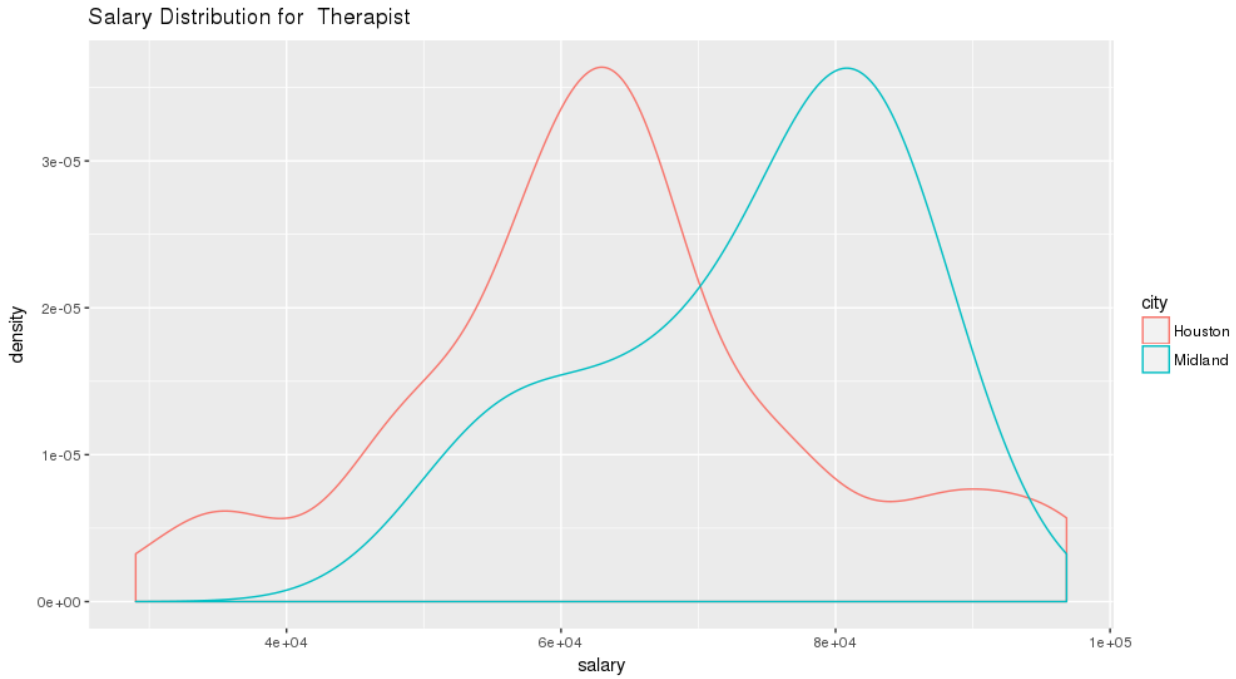
6.5.0.3 Houston vs Midland

These two are chosen as examples of two cities which are in same state yet are of significantly different sizes and average salary (Midland is much smaller with average salary more than 10,000 below Houston). Here some interesting observations are: 1) For Logistics, Midland has spikes in the salaries giving two extremes of job functions. Similar behavior holds for Executive role. 2) For Medical Assistant and Therapist, average salary in Midland is significantly higher than that in Houston.




Salary Distribution for Executive
density
2e-05
1e-05
0e+00
50000
100000
150000
200000
salary
city
Houston
Midland


Salary Distribution for Medical Assistant
density
8e-05
6e-05
4e-05
2e-05
0e+00
30000
60000
90000
salary
city
Houston
Midland



7 Clustering of Roles

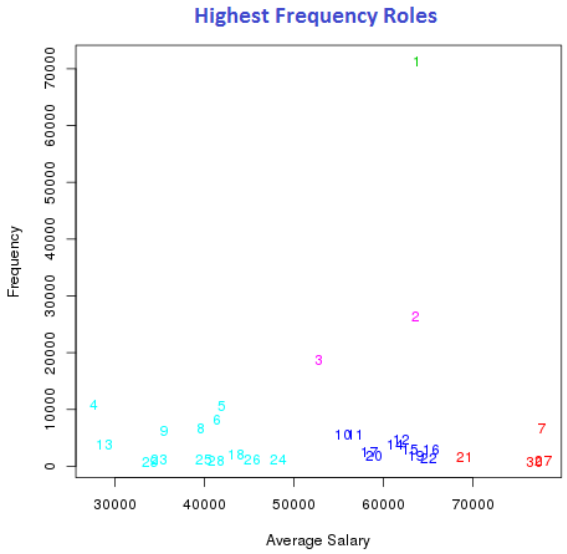
Clustering is a good way to see the pattern among different attributes of similar roles. Two such important attributes are no. of occurrences and average salary. Normally it would be hard to see a strict relation even with enough data points but clustering helps to understand it better.

7.1 Most Frequent

The graph below shows the data points for 30 most frequent roles on frequency and average salary scales. From the lack of any observable pattern, we conclude that among the most common roles, there isn't an inverse relationship between the average salary and frequency. This is expected given that we are choosing the most frequent roles so they should cover the ground for commons positions in healthcare belonging to different salary ranges. Clustering was done using k-means from inbuilt function in r. The corresponding code is in 12.0.2.14.

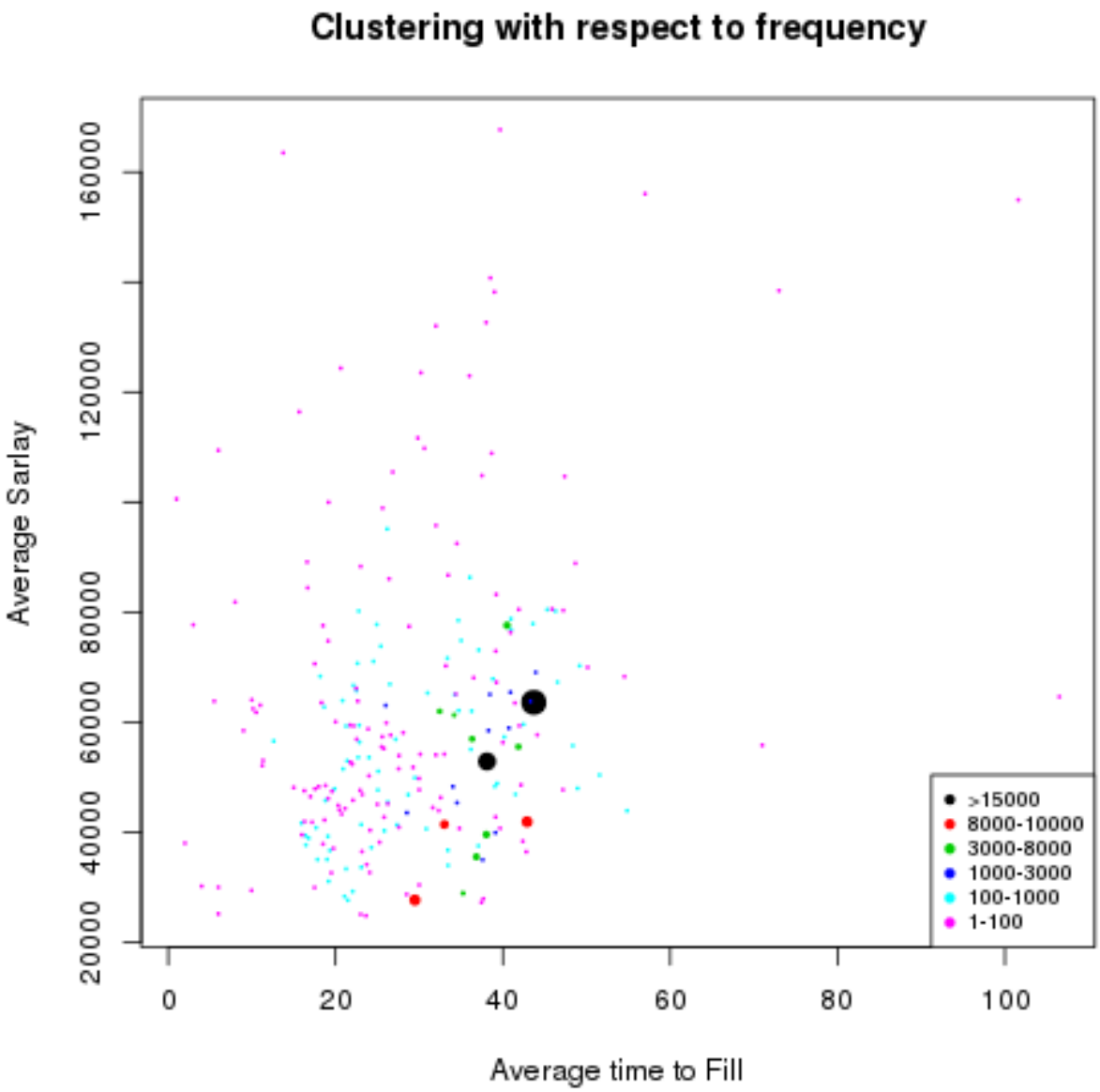
7.2 Highest Paying

Another kind of clustering on the same x and y scale gives a nice pattern. For this we rank the roles based on their average pay and cluster the top ones on the average salary and frequency scale again. Here we see an overall inverse behavior which indicates that for highy paying jobs, higher the job salary is, more rarely it will be posted. This is in a way opposite extreme of earlier graph where we are only looking at the rare roles. The corresponding code is in 12.0.2.15.



7.3 Against Time to Fill

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 between average time to fill and average salary for the list of roles. This grpah, as expected, spreads 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 posted 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 (and large) time to get filled (this is clear from the positions of larger spheres in the plot).

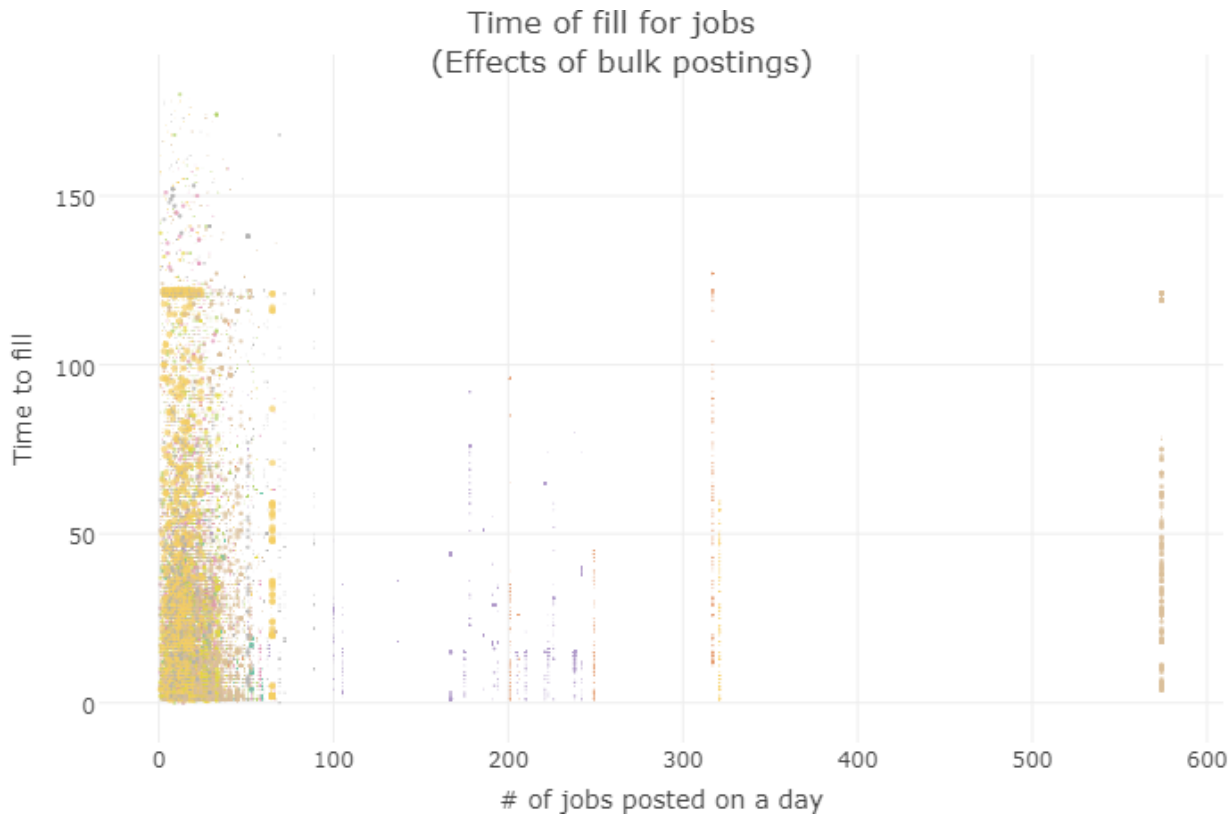

Clustering with respect to frequency
Average Sarlay
20000
40000
60000
80000
120000
160000
0
20
40
60
80
100
Average time to Fill
>15000
8000-10000
3000-8000
1000-3000
100-1000
1-100

8 Bulk Posting Behavior

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 is in appendix 12.0.2.17 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.

8.1 Effect on Filling Time

If we put the job data points on x-y scale where x axis represents the number of total jobs posted on the same day as the given job by the posting company and y axis presents the time to fill, we see that the jobs posted in bulk in general have smaller time to fill. This could be due to the fact that bulk posted jobs are taken down together after a certain period of time if they remain unfilled.



9 Relationship among Roles and Skills

Most of the jobs are defined by more than one roles and/or more than one skills. Any pair of roles (and skills respectively) which appear together often in jobs postings must be related close. This kind of relationship can be quantified using a similarity measure. We use a well known type of similarity measure, called Jaccard coefficient, to define the similarity between two roles R_1 and R_2 as:

$$ab$$

$$J(R_1, R_2) = \frac{|N_{R_1} \bigcap N_{R_2}|}{|N_{R_1} \bigcup N_{R_2}|} a - b\gamma = \mu \frac{1}{\sigma}$$

where N_{R_1} and N_{R_2} are the jobs containing roles R_1 and R_2 respectively. This coefficient is always between 0 and 1. Likewise we can define the similarity measure for skills.

Once defined this way, we can compute the similarity (or distance, given by $1 - J$) for any pair of roles (and skills). This gives a matrix for relationship of roles (and skills) which is symmetric and each element of this matrix (between 0 and 1) represents the corresponding distance between the two roles at the row and column indices. Furthermore, we can compute the closest roles corresponding to each role (and skill) by taking the roles corresponding to the smallest values in each row. For instance, the 3 closest roles for some roles (and skills) with the distances are shown below. The code for this section is given in appendix 12.0.2.19.

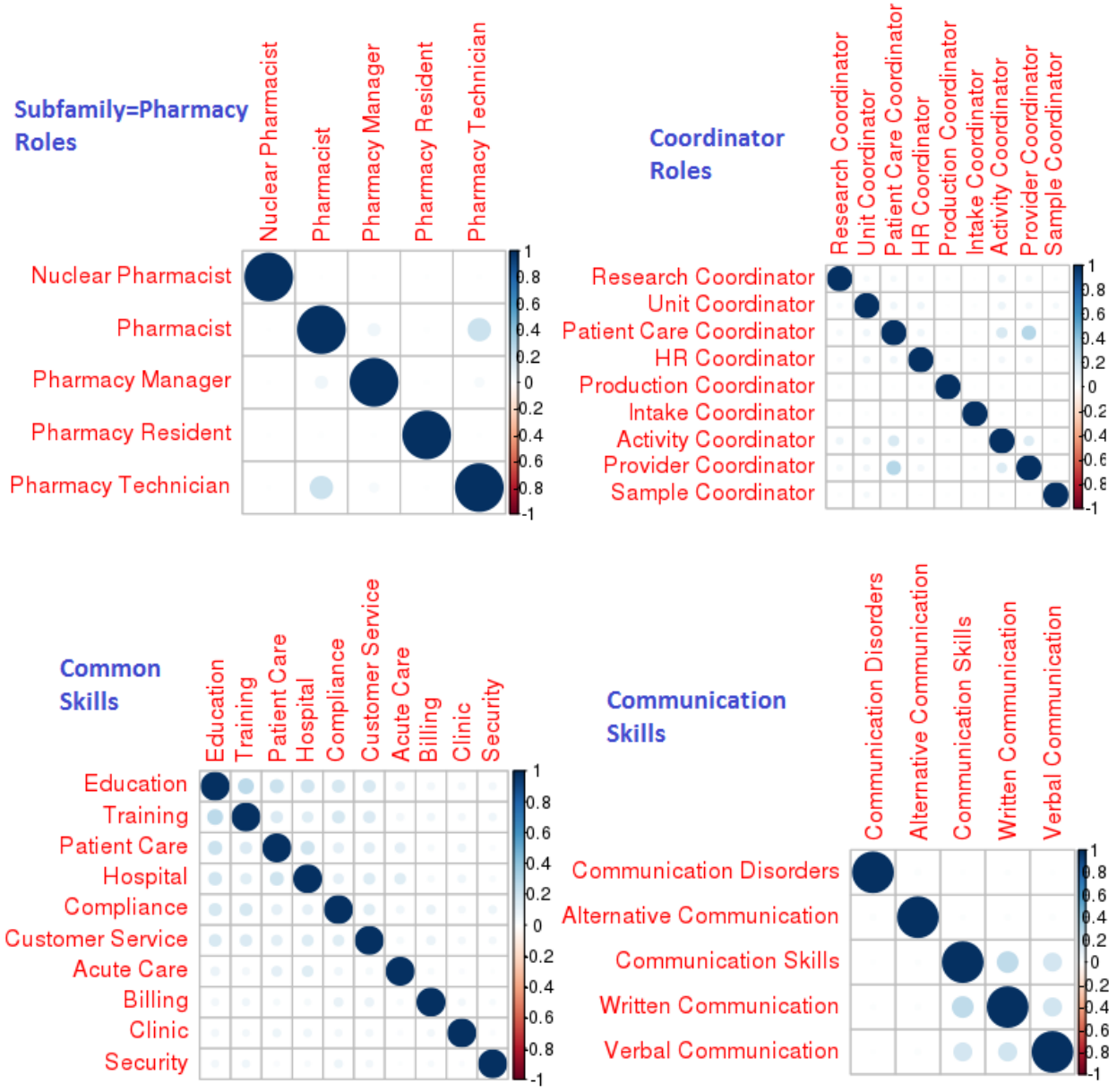
	roles	mc_first.role	mc_first.dist	mc_second.role	mc_second.dist	mc_third.role	mc_third.dist
1	RN	Facilities	0.9059226	LPN	0.9150609	Case Manager	0.9441403
2	Radiation Oncologist	Dosimetrist	0.8931298	Surgical Oncologist	0.9797980	Urologist	0.9909091
3	Sales/Marketing (all)	Executive	0.8981932	Facilities	0.9427278	Driver	0.9437368
4	Executive	Sales/Marketing (all)	0.8981932	Executive Assistant	0.9554211	Facilities	0.9600920
5	Administrator	Chief Nursing Officer	0.9568013	Executive	0.9664900	Nursing Home Administrator	0.9686223
6	CNA	Medical Assistant	0.9207515	Nursing Assistant	0.9337227	Technician/Technologist (all)	0.9367286
7	Driver	Sales/Marketing (all)	0.9437368	Facilities	0.9559499	Case Manager	0.9642867
8	Pathologist	Speech Pathologist	0.8882160	Therapist	0.8982190	Physical Therapist	0.9446728
9	Facilities	RN	0.9059226	Sales/Marketing (all)	0.9427278	Technician/Technologist (all)	0.9553417
10	Nurse Practitioner	Physician Assistant	0.8899542	Advanced Practice Nurse	0.9064998	Clinical Nurse Specialist	0.9262034
11	Technician/Technologist (all)	Medical Assistant	0.9319863	CNA	0.9367286	Medical Tech (unspec)	0.9393137
12	Logistics	Sales/Marketing (all)	0.9750751	Executive	0.9819722	Driver	0.9844437
13	Legal	Sales/Marketing (all)	0.9510879	Facilities	0.9648041	Driver	0.9681082
14	Public Relations	Sales/Marketing (all)	0.9791967	Executive	0.9801404	Direct Support Staff	0.9828974
15	Therapist	Respiratory Therapist	0.8345262	Physical Therapist	0.8466040	Occupational Therapist	0.8731884
16	Recruiter	RN	0.9594808	Case Manager	0.9606721	Facilities	0.9695681
17	Internal Audit	Controller	0.9842871	Legal	0.9879889	Business Services	0.9894322
18	Medical Assistant	Clinical Assistant	0.9202348	CNA	0.9207515	Office Assistant	0.9268394
19	Business Services	Practice Manager	0.9319227	Clinical Manager	0.9414569	Medical Director	0.9526333
20	LPN	RN	0.9150609	Licensed Vocational Nurse	0.9235407	CNA	0.9412068

	skills	mc_first.skill	mc_first.dist	mc_second.skill	mc_second.dist	mc_third.skill	mc_third.dist
1	Clerical	Customer Service	0.9387015	Medical Records	0.9426236	Clinic	0.9445216
2	Finance	Accounting	0.7493755	Engineering	0.9180248	Project Management	0.9370304
3	Acute Care	Hospital	0.8657516	Patient Care	0.8816546	Intensive Care	0.8818801
4	Quality Assurance	Compliance	0.9153468	Training	0.9388687	Education	0.9459632
5	Surgery	Cardiovascular	0.8966982	Hospital	0.9007762	Operating Room	0.9063368
6	Nutrition	Hospital	0.9710489	Purchasing	0.9716163	Training	0.9725676
7	Patient Care	Education	0.7879400	Hospital	0.7948995	Training	0.8432766
8	Operating Room	Surgery	0.9063368	Perioperative	0.9282730	Cardiovascular	0.9551438
9	Product Management	Data Management	0.9314010	Materials Management	0.9411499	Project Management	0.9430937
10	Security	Engineering	0.9191008	Compliance	0.9283886	Training	0.9383390
11	Hospice	Palliative Care	0.8482777	Rehabilitation	0.8945368	Nursing Home	0.9057499
12	Cardiovascular	Intensive Care	0.8437084	Telemetry	0.8872307	Surgery	0.8966982
13	Education	Training	0.7309776	Patient Care	0.7879400	Hospital	0.8032104
14	Training	Education	0.7309776	Compliance	0.8276504	Patient Care	0.8432766
15	Rehabilitation	Acute Care	0.8850180	Hospice	0.8945368	Training	0.9004258
16	Trauma	Intensive Care	0.8822413	Cardiovascular	0.9153554	Emergency Room	0.9206571
17	Project Management	Engineering	0.9044707	Data Management	0.9319156	Finance	0.9370304
18	Customer Service	Education	0.8323120	Training	0.8437458	Hospital	0.8571567
19	Process Improvement	Data Management	0.9408390	Medical Records	0.9430665	Quality Of Care	0.9438212
20	Hearing	Speech	0.8724421	Hand	0.9039351	Pathology	0.9079041

9.1 Correlation Plots

From the roles and skills relationship values, we can plot the correlation matrices for key roles and skills. For this we pick roles (or skills) in certain category and plot the correlations for those. Because the distances are so heavily biased, we don't frequently see color-filled non-diagonal entries in correlation plots. We observe that roles defined by same subfamily (according to provided role hierarchy file in the dataset) are even less closely related than certain combinations of roles from different families. **This calls for a more suitable**

classification of roles. We provide here few examples of the correlation plots, more can be seen in the code in appendix 12.0.2.20.



9.2 Diversity and Dexterity Index

Using the relationship between roles and skills, we define two new coefficients for any job posting as a measure of variability of roles and skills defining that job. First is diversity index, which is defined for a job posting as the maximum of the distance between any two roles governing that job. Farther apart the roles are, the more they are likely to correspond to different responsibilities and more diverse such a job will be. The second coefficient is dexterity index which is defined for a job as the maximum of the distance between any two skills governing that job. A job with high dexterity index will require a more dexterous person to handle unrelated skills. These two coefficients are purported to somehow represent one aspect of a job taking into account the multitudes of skills or roles defining that job.

The code for finding these indices along with functional dependencies is in appendix 12.0.2.21.

10 Limitations

The diversity and dexterity indices described in last section are only included for completeness's sakes and are not good candidates to dictate the salary for fixed role or fixed skill. Primary reason behind that is the limited nature of the dataset in that provided attributes as location, roles and skill do not uniquely govern the behavior of salary for the job postings. Two key examples listed below confirm this observation.

- In a particular city (thereby in constrained location variance), if we list the jobs containing exactly two given roles (e.g. RN and CNA), independent of the skills, the salary values for these jobs are seen to be in close proximity to average salary of CNA (<30,000) *as well as* average salary of RN (>60,000) (see picture below). This implies that in the postings defined by more than one roles, very likely there is a "primary"" role which dictates the value of salary for that posting. Because this information is unavailable in the current dataset, **this calls for a finer tabulation of roles and skills in job postings**.

	company	city	state	salary	role	skill
215323	Presence Health	Chicago	IL	27051	c("CNA", "RN")	c("Education", "Patient Care")
638517	Presence Health	Chicago	IL	27492	c("CNA", "RN")	NA
169812	Northwestern Medicine	Chicago	IL	27857	c("CNA", "RN")	c("Basic Life Support", "Cardiovascular", "Education", "...
209214	Northwestern Medicine	Chicago	IL	28117	c("CNA", "RN")	c("Clinical Health Assessment", "Education", "Family M...
328510	Maxim Healthcare Services	Chicago	IL	28382	c("CNA", "RN")	c("Customer Service", "Training")
353675	Northwestern Medicine	Chicago	IL	28812	c("CNA", "RN")	c("Cardiovascular", "Education", "Hearing", "Nursery", "...
395464	Maxim Healthcare Services	Chicago	IL	29848	c("CNA", "RN")	c("Customer Service", "Training")
538805	University of Chicago Medicine	Chicago	IL	30919	c("CNA", "RN")	c("Cardiovascular", "Hospital", "Night/Third Shift", "Pat...
358714	Maxim Healthcare Services	Chicago	IL	31870	c("CNA", "RN")	c("Customer Service", "Training")
263258	Prairie Manor Healthcare Center	Chicago	IL	36295	c("CNA", "RN")	Skilled Nursing Facility
321127	The University of Chicago Medicine	Chicago	IL	51459	c("CNA", "RN")	c("Emotional Intelligence", "High School Diploma", "Ho...
240003	The University of Chicago Medicine	Chicago	IL	51653	c("CNA", "RN")	c("Cardiovascular", "Hospital", "Patient Care", "Telemet...
450480	The University of Chicago Medicine	Chicago	IL	53111	c("CNA", "RN")	c("Hospital", "Patient Care")
495342	Fresenius Medical Care	Chicago	IL	58360	c("CNA", "RN")	c("Cardiovascular", "Clinic", "Compliance", "Customer ...
642366	Kindred Healthcare	Chicago	IL	60461	c("CNA", "RN")	c("Acute Care", "Critical Care", "Emergency Room", "H...
9472	Kindred Healthcare	Chicago	IL	61214	c("CNA", "RN")	c("Acute Care", "Critical Care", "Emergency Room", "H...
617778	Kindred Healthcare	Chicago	IL	62591	c("CNA", "RN")	c("Acute Care", "Critical Care", "Emergency Room", "H...
568269	Kindred Healthcare	Chicago	IL	64374	c("CNA", "RN")	c("Acute Care", "Critical Care", "Emergency Room", "H...

- As a more serious issue, if we list jobs in a particular city containing exactly one role and one skill (say RN, Education), the salary range can be as big as 100,000 (see picture below) which could be the because of the nature of job (beginner, advanced etc.). Because this information is simply not available, **this calls for a more comprehensive collection of data**.

	company	city	state	salary	role	skill
537454	Presence Health	Chicago	IL	27754	RN	Education
387950	Advocate Health Care	Chicago	IL	46448	RN	Education
533959	Advocate Health Care	Chicago	IL	47518	RN	Education
232687	EDU Healthcare	Chicago	IL	47692	RN	Education
519990	Advocate Health Care	Chicago	IL	47761	RN	Education
414072	Advocate Health Care	Chicago	IL	48071	RN	Education
400069	Advanced Resources	Chicago	IL	48090	RN	Education
120183	Advocate Health Care	Chicago	IL	48148	RN	Education
352560	Advocate Health Care	Chicago	IL	48542	RN	Education
634076	Home Dialysis Services	Chicago	IL	69251	RN	Education
38662	Senior Home Health Agency	Chicago	IL	69588	RN	Education
589669	Aetna	Chicago	IL	71272	RN	Education
84363	Aetna	Chicago	IL	72405	RN	Education
423809	Community First Healthcare of Illinois Inc	Chicago	IL	77633	RN	Education
59710	Legacy Healthcare	Chicago	IL	79265	RN	Education
179734	Healthcare Recruiting Specialists	Chicago	IL	85485	RN	Education
364342	Novasyte, LLC	Chicago	IL	88872	RN	Education
616225	Villa Healthcare	Chicago	IL	91759	RN	Education
285217	Planet Healthcare	Chicago	IL	105310	RN	Education
415070	Planet Healthcare	Chicago	IL	108248	RN	Education
283259	Prolink Healthcare	Chicago	IL	126366	RN	Education

11 Conclusion

11.1 Key Findings

My individual analysis led me to some key results which are as follows:

- Some roles (and skills) are far more prevalent than others.
- Time based variation for postings is what would be anticipated barring few anomalies in filling dates.
- Bigger size of the state doesn't always correspond to more jobs for a given role.
- There is perceivable salary differences among states (even when all other factors are controlled)
- The salary profile for same type of roles can be heavily affected by the city in which the jobs are based.
- Clustering is more noticeable in high paying jobs compared to high frequency jobs.
- Based on standard similarity measures, some pair of roles are related but not too many.
- Missing data and incomplete information about postings is cause of some serious biases in the results.

11.2 Work Summary

In this group project, while my work has mainly focused on dependencies on roles and skills in multiple aspects (e.g. state-wise differences, clustering etc.), rest of my team has tried to come up with ways to relate salary information with other attributes. Combined, our complementary efforts have accomplished significant milestones to the overall goal of healthcare job market data analysis.

11.2.1 Group Contribution

Other teammates in this group effort have accomplished different aspects of this project. Wilson has performed principal component analysis and regression modeling for salary estimation. Zuohao has investigated the contributions from specific skills in pay scales for different jobs and company profiles based on keyword analysis. Huan has looked into correlations among urban/rural population, population density and average salary.

11.2.2 Individual Contribution

I have cleaned up the data, merged different tables together for ease of use and analyzed the salary variations among top roles and skills. Then I have also looked at the state-wise 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. Furthermore, I have put a city-wise spread of job postings in all four states on the map of the United States. I also did the time based analysis for job over the period of postings available in the database. Finally, I have delved into the relationship among different roles and skills and come up with measures for different variations in the jobs.

12 Appendix

12.0.2.1 Data Collection Code

```
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()
role_H_data <- read_excel("~/data/DARL_Fall2017/GREENWICH/role_hierarchy_hc.xls",skip=0)
```

12.0.2.2 Data Integration Code

```
#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
```

12.0.2.3 Data Classfication Code

```
#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")
```

12.0.2.4 Profile by Roles Code

```
#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)

#average salary for roles
avsal_roles <- freq_roles
for (i in 1:num_roles){
  print(i)
  temp <- arfd[which(sapply(lapply(arfd$role,as.character),
                            FUN=function(X) all_roles[i]  %in% X)),]
  avsal_roles[i] <- mean(temp$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)
```

12.0.2.5 Profile by Skills Code

```
#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)

#average salary for skills
avsal_skills <- freq_skills
for (i in 1:num_skills){
  print(i)
  temp <- arfd[which(sapply(lapply(arfd$skill,as.character),
                            FUN=function(X) all_skills[i]  %in% X)),]
  avsal_skills[i] <- mean(temp$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)
```

12.0.2.6 Creation of Columns Code

```
#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["fill_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$post_date))
arfd_new$fill_weekday <- weekdays(as.Date(arfd_new$fill_date))
```

12.0.2.7 Posting Trends Code

```
#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"))
```

12.0.2.8 Filling Trends Code

```
# Plot the # of jobs filling with weekday
temp <- as.data.frame(table(arfd$fill_weekday))
colnames(temp) <- c("weekday","freq")
temp <- temp[c(2,6,7,5,1,3,4),] #to put weekdays in order
temp$weekday <- factor(temp$weekday, levels=temp$weekday)
plot_ly(x = ~temp$weekday, y = ~temp$freq, type = 'bar',) %>%
  layout(yaxis = list(title = 'Count'), margin = list(b = 100),
         xaxis = list(title='Day',ticks = "outside",tickangle=45),
         barmode = 'group', title = '# of Jobs filled with day of the week')

# Plot the # of jobs filling with date
temp <- arfd[which(!is.na(arfd$fill_date)),]
temp1 <- data.frame(date = temp$fill_date)
temp1 <- separate(temp1, "date", c("Year", "Month", "Day"), sep = "-")
temp$fill_date <- as.numeric(temp1$Day)
temp <- as.data.frame(table(temp$fill_date))
colnames(temp) <- c("date","freq")
```

```
plot_ly(x = ~temp$date, y = ~temp$freq, type = 'bar',) %>%
  layout(yaxis = list(title = 'Count'), margin = list(b = 100),
         xaxis = list(title='Day',ticks = "outside",tickangle=45),
         barmode = 'group', title = '# of Jobs filled with date')
```

12.0.2.9 Frequency Contribution from States Code

```
#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')
```

12.0.2.10 Frequency Contribution within States Code

```
# 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")
```

12.0.2.11 Salary-wise Top Roles within States Code

```
#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()
```

12.0.2.12 Salary-wise Comparison among States Code

```
# 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 default 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')
```

12.0.2.13 Salary-wise Comparison among Cities Code

```
if(exists("given_state")){rm(given_state)}
if(exists("given_city")){rm(given_city)}
if(exists("given_role_index")){rm(given_role_index)}
extract_city_for_role <- function(given_state,given_city,given_role_index){
  given_role <- colnames(freq_roles_sorted)[given_role_index]
  temp <- arfd[which(arfd$state==given_state & arfd$city==given_city & arfd$role==given_role),
                     c("company","salary","city","state","role","skill","dexterity_index")]
  return(temp)
}

if(exists("given_df")){rm(given_df)}
give_skills_in_df <- function(given_df){
  #this function takes in a data frame and returns the list of all the skills in that data frame
  out <- NA #starting with list() fucks the whole thing up
  for(i in 1:lengths(df)[1]){
    this_list <- given_df[i,]$skill
    for (j in 1:length(this_list[[1]])){ #lengths is necessary instead of length
      this_skill <- as.character(lapply(this_list, `[[`, j))
      out <- c(out,this_skill)
    }
  }
  return(out)
}

role_chosen_index <- 14
chosen_cities <- c(2,91)
temp <- data.frame()
temp1 <- extract_city_for_role(table_cities[chosen_cities[[1]],]$state,
                               table_cities[chosen_cities[[1]],]$city,
                               role_chosen_index)
temp2 <- extract_city_for_role(table_cities[chosen_cities[[2]],]$state,
                               table_cities[chosen_cities[[2]],]$city,
                               role_chosen_index)
# do this loop if there are more than two cities
# for(i in chosen_cities){
#   print(i)
#   #temp <- extract_city_for_role(table_cities[i,]$state,table_cities[2,]$city,role_chosen_index)
#   temp1 <- extract_city_for_role(table_cities[i,]$state,table_cities[i,]$city,role_chosen_index)
#   temp <- rbind(temp,temp1)
# }
temp <- rbind(temp1,temp2)
ggplot(temp, aes(x=salary, colour=city)) + geom_density() +
  ggtitle(paste("Salary Distribution for ",colnames(freq_roles_sorted)[role_chosen_index]))
```

```
#to check high paying skills etc.
cat('\014')
temp1_low <- temp1[which(temp1$salary>20000 & temp1$salary<70000 & !is.na(temp1$skill)),]
temp1_high <- temp1[which(temp1$salary>40000 & temp1$salary<130000 & !is.na(temp1$skill)),]
low_skills <- give_skills_in_df(temp1_low)
high_skills <- give_skills_in_df(temp1_high)
sort(table(low_skills),decreasing=TRUE)
#sort(table(high_skills),decreasing=TRUE)
temp2_low <- temp2[which(temp2$salary<20000 & temp2$salary<125000 & !is.na(temp2$skill)),]
temp2_high <- temp2[which(temp2$salary>20000 & temp2$salary<130000 & !is.na(temp2$skill)),]
low_skills <- give_skills_in_df(temp2_low)
high_skills <- give_skills_in_df(temp2_high)
sort(table(low_skills),decreasing=TRUE)
sort(table(high_skills),decreasing=TRUE)
#low_skills <- c(give_skills_in_df(temp1_low),give_skills_in_df(temp2_low))
#high_skills <- c(give_skills_in_df(temp1_high),give_skills_in_df(temp2_high))
```

12.0.2.14 Clustering Most Frequent Code

```
#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)
```

12.0.2.15 Clustering Highest Paying Code

```
#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)
```

12.0.2.16 Cluster Against ttf Code

```
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)
```

12.0.2.17 Bulk Posting Behavior Code

```
#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))
```

12.0.2.18 Bulk Posting Effects Code

```
# we plot the jobs on time scale based on their time to fill
# the scatter plot will be in two categories, one for the jobs
# which were posted in bulk and second which weren't
#use temp and temp2 from above
#remove the rows with no time_to_fill
mytemp <- arfd[which(!is.na(arfd$time_to_fill)),
               c("job_id","post_date","time_to_fill","company")]
temp <- as.data.frame(table(mytemp$company))
colnames(temp) <- c("company","total_freq")
temp <- temp[with(temp, order(-total_freq)), ]
#nice things are observed when only 1 company is taken say 1st 1:1 or
#second bigest 2:2 and so on
temp <- temp[1:10,] #only keep top 200
#temp2 contains date with frequency for all companies from mytemp
temp2 <- as.data.frame(table(mytemp$company,mytemp$post_date))
colnames(temp2) <- c("company","post_date","freq")
#selecting only the ones which are in temp
temp2 <- temp2[which(temp2$company %in% temp$company &
```

```
                        temp2$freq > 0),]
#merging for later purposes (basically to keep the total as well)
temp3 <- merge(temp2, temp, all.x = "TRUE")

temp3$post_date <- as.character(temp3$post_date)
big_temp <- merge(mytemp,temp3,all.x = FALSE)
jobs_bulk <- big_temp[which(big_temp$freq>=big_temp$total_freq*0.05),
                      c("company","time_to_fill","total_freq")]
jobs_nonbulk <- big_temp[which(big_temp$freq<big_temp$total_freq*0.05),
                         c("company","time_to_fill","total_freq")]
jobs_bulk <- jobs_bulk[with(jobs_bulk,order(-total_freq)),]
jobs_nonbulk <- jobs_nonbulk[with(jobs_nonbulk,order(-total_freq)),]
#reduce the points in this file to same as bulk one
jobs_nonbulk <- jobs_nonbulk[1:length(jobs_bulk$company),]

jobs_bulk <- big_temp[which(big_temp$freq>=big_temp$total_freq*0.00),]
jobs_bulk <- jobs_bulk[with(jobs_bulk,order(freq)),]

plot_ly(data = big_temp, x = ~freq, y = ~time_to_fill,
        color = ~company, showlegend=FALSE,type = 'scatter',
        size=0.012, mode='markers', marker=list(size=~freq/80))%>%
  layout(title = paste('Time of fill for jobs',
                       '\n (Effects of bulk postings)'),
         yaxis = list(title="Time to fill",
                      zeroline = FALSE),
         xaxis = list(title="# of jobs posted on a day",
                      zeroline = FALSE))
```

12.0.2.19 Relation Roles & Skills Code

```
#roles relaionship table
temp <- arfd[which(!is.na(arfd$role)),]
roles_relation_table <- matrix(0,nrow=num_roles,ncol=num_roles)
roles_common_table <- matrix(0,nrow=num_roles,ncol=num_roles)
for (i in 1:num_roles){
  for (j in 1:num_roles){
    #Jaccard coefficient is defined by |S1 INTERSECTION S2|/|S1 UNION S2|
    #to count S1 INTERSECTION S2
    print(paste(as.character(i),"->",as.character(j)))
    temp <- arfd[which(sapply(lapply(arfd$role,as.character),
                              FUN=function(X) (all_roles[i]  %in% X) &&
                                (all_roles[j]  %in% X))),]
    num_s1ANDs2 <- length(temp$role)
    num_s1ORs2 <- as.numeric(freq_roles[i])+as.numeric(freq_roles[j])-num_s1ANDs2
    if(num_s1ORs2!=0){
      roles_relation_table[i,j] = num_s1ANDs2/num_s1ORs2
    }
    roles_common_table[i,j] <- num_s1ANDs2
  }
}
#to govern the distsance
roles_relation_table <- 1-roles_relation_table
#the diagonal should be zero for roles_relation_table after above line
```

```
rrt <- roles_relation_table+diag(271)
#the indices of 5 closest neighbors in each row
indices_nearest <- t(apply(rrt, 1, order)[ 1:5, ])
closest_roles_table <- data.frame(
  roles = all_roles,
  mc_first = list(index=indices_nearest[,1],role=all_roles[indices_nearest[,1]],dist=0),
  mc_second = list(index=indices_nearest[,2],role=all_roles[indices_nearest[,2]],dist=0),
  mc_third = list(index=indices_nearest[,3],role=all_roles[indices_nearest[,3]],dist=0),
  mc_fourth = list(index=indices_nearest[,4],role=all_roles[indices_nearest[,4]],dist=0),
  mc_fifth = list(index=indices_nearest[,5],role=all_roles[indices_nearest[,5]],dist=0))
for(i in 1:num_roles){
  closest_roles_table[i,]$mc_first.dist =
    roles_relation_table[i,indices_nearest[i,1]]
  closest_roles_table[i,]$mc_second.dist =
    roles_relation_table[i,indices_nearest[i,2]]
  closest_roles_table[i,]$mc_third.dist =
    roles_relation_table[i,indices_nearest[i,3]]
  closest_roles_table[i,]$mc_fourth.dist =
    roles_relation_table[i,indices_nearest[i,4]]
  closest_roles_table[i,]$mc_fifth.dist =
    roles_relation_table[i,indices_nearest[i,5]]
}

#skills relaionship table
temp <- arfd[which(!is.na(arfd$skill)),]
skills_relation_table <- matrix(0,nrow=num_skills,ncol=num_skills)
skills_common_table <- matrix(0,nrow=num_skills,ncol=num_skills)
for (i in 1:num_skills){
  for (j in 1:num_skills){
    #Jaccard coefficient is defined by |S1 INTERSECTION S2|/|S1 UNION S2|
    #to count S1 INTERSECTION S2
    print(paste(as.character(i),"->",as.character(j)))
    temp <- arfd[which(sapply(lapply(arfd$skill,as.character),
                              FUN=function(X) (all_skills[i] %in% X) &&
                                (all_skills[j]  %in% X))),]
    num_s1ANDs2 <- length(temp$skill)
    num_s1ORs2 <- as.numeric(freq_skills[i])+as.numeric(freq_skills[j])-num_s1ANDs2
    if(num_s1ORs2!=0){
      skills_relation_table[i,j] = num_s1ANDs2/num_s1ORs2
    }
    skills_common_table[i,j] <- num_s1ANDs2
  }
}

#to govern the distsance
skills_relation_table <- 1-skills_relation_table #do it only once
#the diagonal should be zero for skills_relation_table after above line
srt <- skills_relation_table+diag(341)
#the indices of 5 closest neighbors in each row
indices_nearest <- t(apply(srt, 1, order)[ 1:5, ])
closest_skills_table <- data.frame(
  skills = all_skills,
  mc_first = list(index=indices_nearest[,1],skill=all_skills[indices_nearest[,1]],dist=0),
```

```
  mc_second = list(index=indices_nearest[,2],skill=all_skills[indices_nearest[,2]],dist=0),
  mc_third = list(index=indices_nearest[,3],skill=all_skills[indices_nearest[,3]],dist=0),
  mc_fourth = list(index=indices_nearest[,4],skill=all_skills[indices_nearest[,4]],dist=0),
  mc_fifth = list(index=indices_nearest[,5],skill=all_skills[indices_nearest[,5]],dist=0))
for(i in 1:num_skills){
  closest_skills_table[i,]$mc_first.dist =
    skills_relation_table[i,indices_nearest[i,1]]
  closest_skills_table[i,]$mc_second.dist =
    skills_relation_table[i,indices_nearest[i,2]]
  closest_skills_table[i,]$mc_third.dist =
    skills_relation_table[i,indices_nearest[i,3]]
  closest_skills_table[i,]$mc_fourth.dist =
    skills_relation_table[i,indices_nearest[i,4]]
  closest_skills_table[i,]$mc_fifth.dist =
    skills_relation_table[i,indices_nearest[i,5]]
}
```

12.0.2.20 Correlation Plots Code

```
if(exists("given_role_type")){rm(given_role_type)}
if(exists("custom_roles")){rm(custom_roles)}
corrmat_for_role_type <- function(given_role_type,custom_roles){
  if(missing(custom_roles)){
    temp <- role_H_data[which(role_H_data$job_sub_family==given_role_type),"role"]
    temp <- unique(temp$role)
    custom_roles <- temp[which(temp %in% all_roles)]
  }
  temp <- 1-roles_relation_table[custom_roles,custom_roles]
  #we put the option roles_common_table[...] in above line
  #another option for another line is temp <- 1-roles_relation_table[...] (much lighter)
  dtemp <- diag(temp)
  temp <- temp/dtemp[row(temp)] #dividing each row by diagonal entry
  return(temp)
}

if(exists("custom_skills")){rm(custom_skills)}
corrmat_for_skills <- function(custom_skills){
  temp <- 1-skills_relation_table[custom_skills,custom_skills]
  #we put the option skills_common_table[...] in above line
  #another option for another line is temp <- 1-skills_relation_table[...] (much lighter)
  dtemp <- diag(temp)
  temp <- temp/dtemp[row(temp)] #dividing each row by diagonal entry
  return(temp)
}

#Because the skills_relationship_table and skill_relationship_table have very
#uneven entries, and we don't see any strong relationship other than any role (and skill) to itself
#we use anohter way we look at the roles_common_table (and skills_common_table) and for each row
#and consider each row as containing it's "commonality" with other roles
#we obtain these commonalities by dividing each row by the diagonal entry because diagonal
#entry shows the number of rows in which that role appears.

#just doing with first 10 roles
```

```
temp <- 1-roles_relation_table
dtemp <- diag(temp)
temp <- temp/dtemp[row(temp)] #dividing each row by diagonal entry
corrplot(temp[1:10,1:10])

#doing with respect to one category of roles
corrplot(corrmat_for_role_type("Administration"))
corrplot(corrmat_for_role_type("Healthcare Management"))
corrplot(corrmat_for_role_type("Lab/Research"))
corrplot(corrmat_for_role_type("Pharmacy"))
corrplot(corrmat_for_role_type("Other"))
corrplot(corrmat_for_role_type("Executive"))
corrplot(corrmat_for_role_type("Nursing"))
corrplot(corrmat_for_role_type("IT"))
corrplot(corrmat_for_role_type("Dental"))
corrplot(corrmat_for_role_type("Business Services"))

#custom list of roles
corrplot(corrmat_for_role_type("",c("RN","Facilities","LPN","CNA","Case Manager")))
temp <- c("Business Analyst","Healthcare Data Analyst","Medicare Analyst","Data Specialist","Business S
corrplot(corrmat_for_role_type("",temp))
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Coordinator", all_roles))]))
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Manager", all_roles))]))
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Therapist", all_roles))]))
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Specialist", all_roles))])) #not much useful
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Data", all_roles))])) #not much useful
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Surg", all_roles))]))
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Technician", all_roles))])) #not much useful
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Nurse", all_roles))])) #not much useful
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Medical", all_roles))])) #not much useful
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Analyst", all_roles))])) #not much useful
corrplot(corrmat_for_role_type("",all_roles[which(grepl("Lab", all_roles))])) #not much useful

#custom list of skills
corrplot(corrmat_for_skills(c("Healthcare Law","Medical Law")))
corrplot(corrmat_for_skills(all_skills[which(grepl("Care", all_skills))]p))
corrplot(corrmat_for_skills( all_skills[which(grepl("Communica", all_skills))]))
corrplot(corrmat_for_skills(all_skills[which(grepl("Medical", all_skills))]))
corrplot(corrmat_for_skills(all_skills[which(grepl("Manage", all_skills))])) #does not help
corrplot(corrmat_for_skills(all_skills[which(grepl("Health", all_skills))]))
corrplot(corrmat_for_skills(all_skills[which(grepl("Thera", all_skills))]))
temp <- c("Education","Training","Patient Care","Hospital","Compliance",
          "Customer Service","Acute Care","Billing","Clinic","Security")
corrplot(corrmat_for_skills(temp))
corrplot(corrmat_for_skills(all_skills[which(grepl("Medicine", all_skills))])) #doesn't help much
```

12.0.2.21 Diversity & Dexeterity Index Code

```
#next we translate the roles and skills columns into a numeric value by
#making use of these distances. we define two new columns,
#"diversity-index" and "dexterity-index", as the maximum distance between
#any pair of roles (and skills respectively) which define a job
temp_nozero <- arfd[which(!is.na(arfd$role)),]
```

```
temp_nozero$diversity_index <- 0
temp_nozero_noone <- temp_nozero[which(lengths(temp_nozero$role)>1),]

lentemp <- length(temp_nozero_noone$job_id)
for (i in 1:lentemp){
  if(ceiling(i %% 100 ==0)) {print(i)}
  temp_nozero_noone[i,]$diversity_index <-
    find_div_index(temp_nozero_noone[i,]$role)
}

temp_nozero_noone <-
  temp_nozero_noone[,c("job_id","diversity_index")]
temp_nozero <-
  merge(temp_nozero,temp_nozero_noone,by="job_id",all.x=TRUE)
temp_nozero$diversity_index <-
  rowSums(temp_nozero[, c("diversity_index.x","diversity_index.y")],
          na.rm = TRUE)
temp_nozero <- temp_nozero[,c("job_id","diversity_index")]
mytemp <- merge(arfd,temp_nozero,by="job_id",all.x=TRUE)
arfd <- mytemp

temp_nozero <- arfd[which(!is.na(arfd$skill)),]
temp_nozero$dexterity_index <- 0
temp_nozero_noone <-
  temp_nozero[which(lengths(temp_nozero$skill)>1),]

lentemp <- length(temp_nozero_noone$job_id)
for (i in 1:lentemp){
  if(ceiling(i %% 100 ==0)) {print(i)}
  temp_nozero_noone[i,]$dexterity_index <-
    find_dex_index(temp_nozero_noone[i,]$skill)
}

temp_nozero_noone <-
  temp_nozero_noone[,c("job_id","dexterity_index")]
temp_nozero <-
  merge(temp_nozero,temp_nozero_noone,by="job_id",all.x=TRUE)
temp_nozero$dexterity_index <-
  rowSums(temp_nozero[, c("dexterity_index.x",
                          "dexterity_index.y")],
          na.rm = TRUE)
temp_nozero <- temp_nozero[,c("job_id","dexterity_index")]
mytemp <- merge(arfd,temp_nozero,by="job_id",all.x=TRUE)
arfd <- mytemp

if(exists("role_list")){ rm(role_list) }
find_div_index <- function(role_list){
  #this function finds the diversity index as the maximum of the distances
  #between any two roles defining a job
  div_index <- 0 #just keep track of max
  length_role_list <- length(role_list[[1]])
  for (i in 1:(length_role_list-1)){ #lengths is necessary instead of length
    for (j in (i+1):length_role_list){
```

```
      this_role_first <- lapply(role_list, `[[`, i)
      this_role_second <- lapply(role_list, `[[`,j)
      index_this_role_first <- match(this_role_first,all_roles)
      index_this_role_second <- match(this_role_second,all_roles)
      temp <- roles_relation_table[cbind(index_this_role_first,
                                         index_this_role_second)]
      if(temp>div_index){
        div_index <- temp
      }
    }}
  return(div_index)
}

if(exists("skill_list")){ rm(skill_list) }
find_dex_index <- function(skill_list){
  #this function finds the dexterity index as the maximum of the distances
  #between any two skills defining a job
  dex_index <- 0 #just keep track of max
  length_skill_list <- length(skill_list[[1]])
  for (i in 1:(length_skill_list-1)){
    for (j in (i+1):length_skill_list){
      this_skill_first <- lapply(skill_list, `[[`, i)
      this_skill_second <- lapply(skill_list, `[[`,j)
      index_this_skill_first <- match(this_skill_first,all_skills)
      index_this_skill_second <- match(this_skill_second,all_skills)
      temp <- skills_relation_table[cbind(index_this_skill_first,
                                          index_this_skill_second)]
      if(temp>dex_index){
        dex_index <- temp
      }
    }}
  return(dex_index)
}
```