STA 141C Final Report

Jixian Fu, Liya Li, Chengchen Luo

{mrfu, lyali, dorluo}@ucdavis.edu

1. Introduction

In this project, we aim to find the relationship between time, agencies and recipients with funding taxpayers paid to the U.S. Government. The difference between action date and last modified date indicates the efficacy of the government, and we want to know if the speed is associated with any agency, any party in the office or the amount of transaction funding. The second question we want to know is which agencies have the money and which agencies ended up spend the money. Moreover, we want to further analyze the recipients of each transaction.

In this project, we use the techniques we have learned in class. We use cluster to grab the data, apply parallelism processing to import and further clean the data. Both Python and R are used for further analysis including applying group by computation, apply family functions and object oriented programming, etc.

2. Data Importing

We use the original US transaction dataset **transaction.csv** from our class cluster. This dataset is formed based on the raw data behind usaspending.gov and it shows how the US government spends taxpayer money. Since our tasks are mainly about the general aspects of the dataset, we use partial data of original. We pull out the first 5 million rows as a sample to be analyzed. After all, the chosen dataset fits as “small” enough for we to draw graphs in our local machine while as “big” enough to represent the whole dataset.

3. Dataset Analysis

➢ Missing Values Removal

There are lots of missing values including NA's and empty strings "" especially under those "name columns". For the reason that we are only interested into the more complete data rows that indicate maximum information of each transaction we need, we tend to raise less bias and so we remove some rows that contains one or more missing value based on our need in each sub analysis.

➢ Outstanding Recipient Detection

Applying group-by techniques and removing empty strings in the *recipient_name* column leads to several findings about outstanding recipients and their locations. The top 10 recipients who frequently receive government fundings in counts are shown in Figure 1. The top 10 recipients who receive the most government fundings in amount are shown in Figure 2, where we add the industry column for more direct comparison.

Industry Category	Recipient Name	Funding Receiving Counts
Operation, Technology and Manufacturing	NATIONAL INDUSTRIES FOR THE BLIND	675367
Retail	KAUFMAN COMPANY INC.	439084
Industrial supply distribution	W.W. GRAINGER INC.	270128
Scientific tools and reagents	FISHER SCIENTIFIC COMPANY L.L.C.	132965
Distribution	GRAYBAR ELECTRIC COMPANY INC.	96325
Manufactured Goods	JROTC DOG TAGS INC.	59320
Furniture manufacturing	HERMAN MILLER INC.	44968
Telecom	LYNN ELECTRONICS CORP.	39968
Manufacturing, Distribution, and Retail	L C INDUSTRIES INC.	38643
Aerospace & Defense	LOCKHEED MARTIN CORPORATION	38605

Figure 1: Who get government fundings highly in count?

From Figure 1, we notice that NATIONAL INDUSTRIES FOR THE BLIND has the most times receiving government fundings and it is about 20 times the receiving times

for Aerospace & Defense Industry, which is surprising. And if we further look into the top 10 recipients who receive the most fundings and their industry categories:

Industry Category	Recipient Name	Total Obligation
Defense, Shipbuilding	HUNTINGTON INGALLS INCORPORATED	4.506517e+13
Aerospace & Defense	NORTHROP GRUMMAN SPACE & MISSION SYSTEMS CORP.	2.846893e+13
Aerospace & Defense	LOCKHEED MARTIN CORPORATION	1.481231e+13
Education	THE REGENTS OF THE UNIVERSITY OF CALIFORNIA	3.784553e+12
Education	MASSACHUSETTS INSTITUTE OF TECHNOLOGY	3.293463e+12
Aircraft manufacturing	UNITED TECHNOLOGIES CORPORATION	3.096351e+12
Defense, Ship Repairing	BAE SYSTEMS NORFOLK SHIP REPAIR INC.	2.700294e+12
Aerospace and defense	NORTHROP GRUMMAN SYSTEMS CORPORATION	1.130140e+12
Aerospace and defense	ATK LAUNCH SYSTEMS INC.	1.068994e+12
Aerospace and defense	RAYTHEON COMPANY	7.792496e+11

Figure 2: Who get government fundings highly in amount?

Figure 2 shows that in this dataset, the industry of Defense, Shipbuilding gets most of the government fundings while Aerospace & Defense Industry occupies half of the ranks, meaning that the government supports Aerospace & Defense Industry very much which makes sense in reality. And we can see that the UC Regents and MIT are ranked at 4th and 5th in the list, meaning that High Level Education also catches the government's eyes while giving fundings.

The comparison between two tables above shows that recipient named LOCKHEED MARTIN CORPORATION receives large amount of government fundings though its rank in the number of times receiving fundings is just the 10th.

Moreover, when we look into the total transaction numbers among countries, we find that the United States has the main portion which is way much more that the other countries. See Figure 3.

Country	Total Obligation
UNITED STATES	4269966
UNITED STATES OF AMERICA	410378
GERMANY	3405
AFGHANISTAN	2447
IRAQ	1715
JAPAN	1683
CANADA	1122
KOREA REPUBLIC OF	1086
ITALY	990
UNITED KINGDOM	942

Figure 3: Where do the fundings go?

➢ Agency analysis

One question we come up with is how the funding transfer through each agency. We first look at the agencies that have fundings and find that Department of Defense gets the most fundings, 1639251148020 which is 1.6 trillion dollars. It is 500 billion more than the second agency. Then we plot a barplot for the top 10 agencies; however, the sum difference is too large, we then exclude the Department of Defense to make other agencies show. See in Figure 4.

	funding_toptier_agency_name	total_obligation	transaction_count	obligation_per_trans
25	Department of Energy	1639251148020	6444	254384101
62	National Aeronautics and Space Administration	1113408993223	19832	56142043
23	Department of Defense	103987674234274	2923655	35567697
7	Architect of the Capitol	158256144	6	26376024
5	Agency for International Development	51208473548	2977	17201368

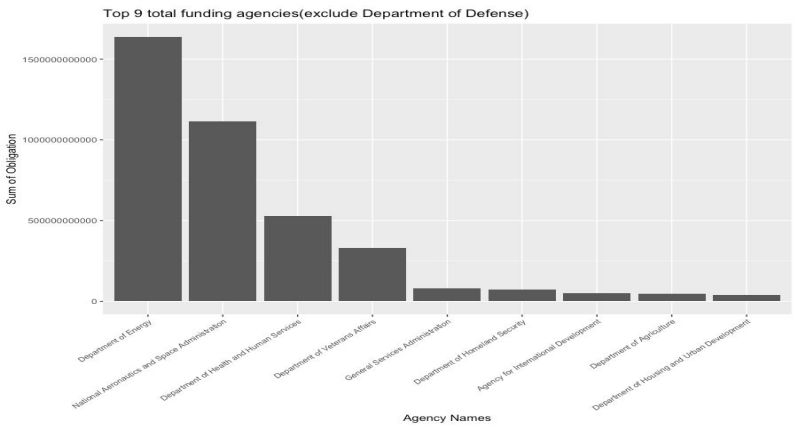


Figure 4

Some agency has only a few transactions but has really high amount of spending per transaction. Such as Architect of the Capitol, they only has 6 transactions in 5 billion transactions, but they got 26376024 per transaction which makes they the top five average funding agencies in the 99 agencies. See in Figure 5.

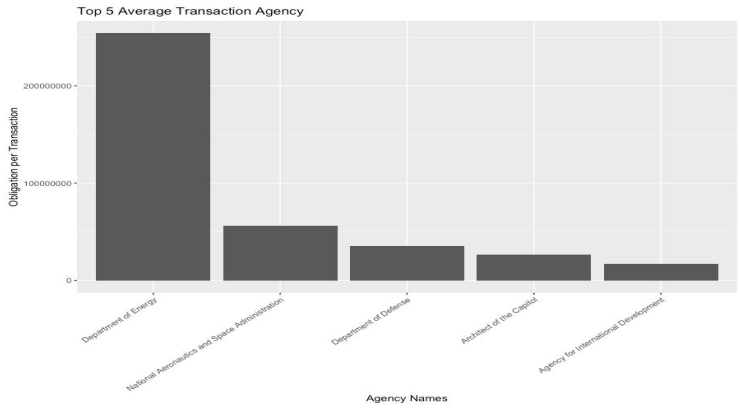


Figure 5

Compare to Department of Defense (DOD) , Department of Energy (DOE) as the second largest funding agency they spend much more per transaction. Which is interesting to know for us, soon we realized that this is making sense from the recipient analysis. DOD has much more small transactions with the small business and individuals than DOE, whereas DOE are mainly working with large institutions.

There are 73 Awarding agencies in the dataset. In most of the cases, the funding agencies is the same agency who receives the funds. However, there are exceptions. See Figure 6.

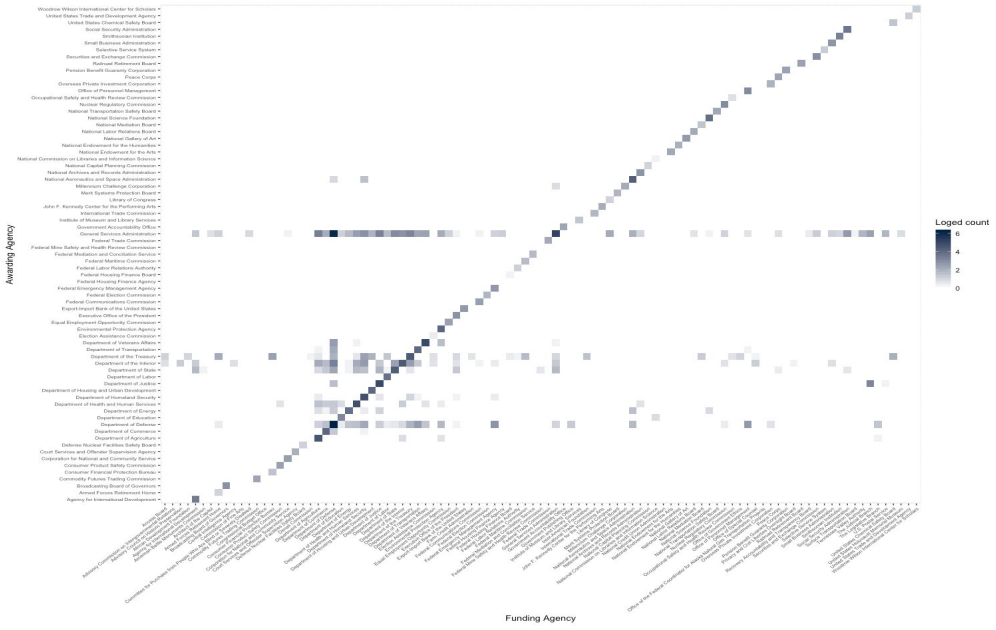


Figure 6: Heatmap

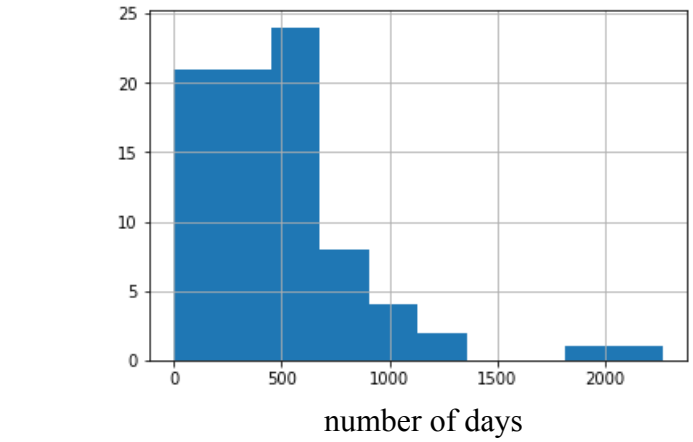
This heat map is a visual representation of the contingency table of funding and awarding agency. For better visualization, we take a log for the counts in the contingency table, since some number is very big and some are very small. The very obvious line in the middle is General Service Admission, this agency received funding from almost every agency. This agency is an independent agency of the United States government, was established in 1949 to help manage and support the basic functioning of federal agencies. Other agency such as Department of Treasury, Department of Interior are receiving funds from other agencies as well.

➢ Deep Into Transaction Time Length

When browsing through the data, we see that sometimes there is a gap between the ‘action_date’, which represents the time that a transaction beings , and the ‘last_modified_date’, which roughly represents the time that a transaction ends. We called it ‘time difference’. We think that, there might be a relationship between the ‘time difference’ and the difference of agencies, the execution style of the two political parties, and the amount of total obligations.

Therefore, we calculated the time difference to see if there is indeed some relations between them.

The Distribution of the Mean of Time Difference by Agencies



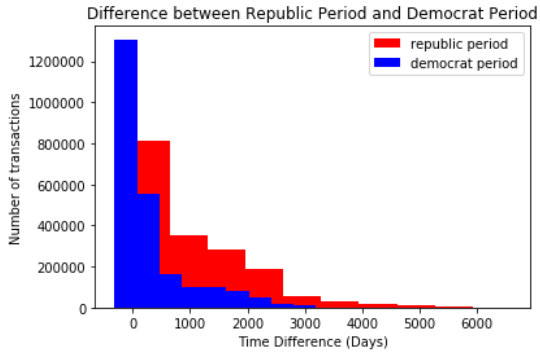
From this figure, we can see that most of the mean of time difference grouped by Agencies is close to 0, while a few is very large. The whole distribution is seriously right skewed. Hence, the agencies with a long transaction processing time is fewer than those with shorter ones. To see which kind of agencies have the longest and shortest mean of processing time, we get the following results. See Figure 7.

```
In [360]:
1 #agencies with the largest time difference
2 by_agency.sort_values(ascending=False).head()

Out[360]: awarding_toptier_agency_name
          United States Holocaust Memorial Museum    2264.000000
          Commission on Civil Rights                 2034.937500
          Armed Forces Retirement Home               1239.500000
          Federal Emergency Management Agency        1199.328638
          Executive Office of the President          1107.915344
          Name: difference, dtype: float64

In [361]:
1 #agencies with the smallest time difference
2 by_agency.sort_values(ascending=True).head()

Out[361]: awarding_toptier_agency_name
          Federal Mine Safety and Health Review Commission     0.000000
          Woodrow Wilson International Center for Scholars    20.000000
          National Science Foundation                         34.804541
          Consumer Financial Protection Bureau                37.142857
          Federal Labor Relations Authority                   38.375000
          Name: difference, dtype: float64
```

Figure 7: Python Output

We can not tell there is an obvious reason to explain the difference within the information provided. However, it is interesting to see that the executive office of the president have a large mean of transaction processing time.

Then we move on to the topic of parties. For convenience, we select the end of 2008 as the ending date of the execution of president Bush, which separates data into those in 2001-2008 and those in 2009-2016. These two roughly represents the execution period of the republicans and democrats. Figure 8 compares them.

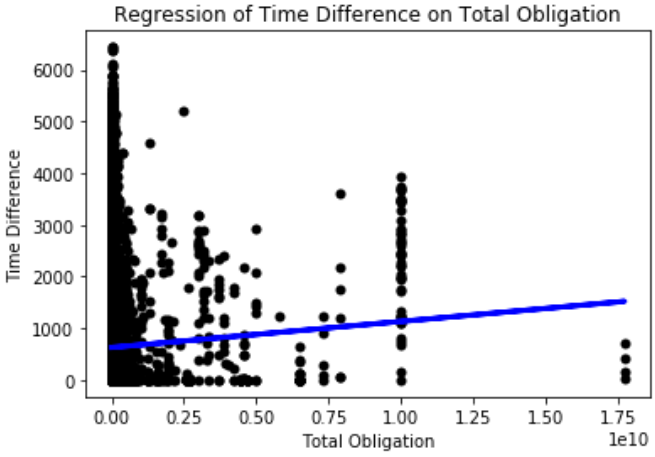


From the figure, we can see that the democrats and republicans both have a right skew distribution of time difference. However democrats have a more obvious right skewed, which means democrats finish a transaction faster than republicans in general.

Next, since time difference and total obligation are both numeral, we apply the method of linear regression to check their relationship. The result of the model goes below

```
R^2 is 0.00032221898690121265
co-efficient is [[5.03296967e-08]]
intercept is [629.87993373]
```

The R square is very small. And, if we test the model again actual data we can see that



The model can not give a good prediction of the actual data represented by the black points. Therefore, we do not think the total obligation can explain the time for processing transactions.

3. Methods

➢ Cluster And SLURM Usage

We obtained a csv file called **selected_columns.csv** from the cluster, which contains 13 needed columns. Using bash pipes and sbatch process on the cluster, then moving the target csv file from the remote home directory to local machine increases the running time of data obtaining step efficiently. Bash scripts **selected_columns.sh** and **submit1.sh** are attached as Appendix 1.

➢ Parallelism With Local Processors

When computing the time difference between the last modified date and the action date, I used multiprocessing with 8 processors in Python. Therefore, the computation time is much shorter.

➢ Group By Computation

During the preprocessing stage, we apply several group by techniques in R to merge and rearrange statistics and data. Then, ordered data tables are created for comparison.

- Efficiency Improvement With Apply family

Applying the apply family functions in python code enhances the increase of running time while leaving our code neat.

- Object Oriented Programming

Since most tools in Python is from open source packages, we import many of them and use the built-in methods of those objects. Object called function leads to direct and clear logic of computation while making code lines efficient and easy to read.

4. Conclusion

The analysis on outstanding recipients lead to the conclusion that the government has been using the fundings to support not only the defence industries but also education institutes and etc. A very large portion of fundings go directly to Aerospace and Defence Industries which makes sense in reality. And, the United States receives most of the fundings compared to the other countries, which encourages us to dig deeper into transactions related to the United States.

Based on the analysis of the funding agencies and awarding agencies we can conclude that most of the funding is going to the fields that is not straightly related to citizen's everyday life such as military, energy, space technology; Mose agency receive fundings from themselves with some exceptions.

The linear regression model try doesn't lead us to conclude the relationship between total obligation and the time for processing transactions, thus we decide not to use a linear regression model to explain our data at this point.

5. Appendix

➢ **1. Bash Scripts**

selected_column.sh

```
# get data
CSVFILE=/scratch/transaction.csv

# obtain the needed col indexes from colname_index.txt
 Action_date =  3
 Last_modified_date: = 4
 Total_obligation = 8
 Awarding_agency_id = 17
 Pop_country_code = 27
 Pop_country_name = 28
 Recipient_location_country_code = 34
 Recipient_location_country_name = 35
 Recipient_location_state_code = 36
 Recipient_name = 51
 Awarding_toptier_agency_name =  53
 Funding_toptier_agency_name = 54
 Business_categories = 61

# extract needed cols
cat ${CSVFILE} | cut --delimiter ',' --fields=
${Action_date},${Last_modified_date}, ${Total_obligation},  ${Awarding_agency_id}, ${Pop_country_code},
${Pop_country_name}, ${Recipient_location_country_code}, ${Recipient_location_country_name},
${Recipient_location_state_code}, ${Recipient_name}, ${Awarding_toptier_agency_name},
${Funding_toptier_agency_name}, ${Business_categories}
 > selected_columns.csv


# run these code in local mac command to upload the files to remote
# scp /Users/liliya/Desktop/Winter\ Quarter\ 2019/STA\ 141C/Final/submit1.sh
s141c-23@gauss.cse.ucdavis.edu:/home/s141c-23/STA141C-HW5
# scp /Users/liliya/Desktop/Winter\ Quarter\ 2019/STA\ 141C/Final/selected_columns.sh
s141c-23@gauss.cse.ucdavis.edu:/home/s141c-23/STA141C-HW5

# dowanload selected_columns.csv to local
# scp s141c-23@gauss.cse.ucdavis.edu:/home/s141c-23/STA141C-HW5/selected_columns.csv
/Users/liliya/Desktop/Winter\ Quarter\ 2019/STA\ 141C/Final/selected_columns.csv
```

submit1.sh

```
#!/bin/bash -l

# Use the staclass partition. Only applies if you are in STA141C
#SBATCH --partition=staclass
```

```
# Use two cores to get some pipeline parallelism
#SBATCH --cpus-per-task=2

# Give the job a name
#SBATCH --job-name=hw5_step1

# Email me a result
#SBATCH --mail-type=ALL
#SBATCH --mail-user=lyali@ucdavis.edu

bash selected_columns.sh
(This code file is modified from previous HW5 script and job name accidently remains)
```

➢ **Python Script**

```
# In[35]:

import pandas as pd #for dataframe
import numpy as np #for numpy array
import datetime #for counting the date_difference
from multiprocessing import Pool #for multiple processing
import multiprocessing
import time #for count the time
import matplotlib.pyplot as plt #for plot
from sklearn.linear_model import LinearRegression #for linear regression


# In[3]:


#read the data by chunk, and making a sample using the first 5 million rows
raw = pd.read_csv('selected_columns.csv',chunksize= 1000)
data = raw.get_chunk(5000000)

#for group members to use
data.to_csv('finaldata.csv')


# In[4]:


#clean the data by dropping duplicates and missing values
dropped_dup= data.drop_duplicates(['action_date', 'total_obligation'], keep= 'first')
dropped_dup.dropna(subset=['business_categories','awarding_toptier_agency_name', 'action_date',
'last_modified_date', 'total_obligation'], inplace=True)
dropped_dup.reset_index(inplace=True)


# In[292]:
```

```
def get_date(string):
  '''for transform string into time dtype'''
  return datetime.datetime.strptime(string,"%Y-%m-%d")

def modify(data):
  '''for make the columns of time strings into time dtype'''
  data['start_time'] = data.action_date.apply(get_date)
  data['end_time'] = data.last_modified_date.apply(get_date)
  return data


##using parallel method to do the transformation
start_time = time.time()

pool = multiprocessing.Pool(processes=8) # making 8 processes
result = pool.map(modify, (dropped_dup,))
pool.close()
pool.join()

print("--- {} seconds ---".format(time.time() - start_time))


# In[385]:


#calculate the time difference
new_table = result[0].copy()
new_table['time_diff'] = new_table['end_time'] - new_table['start_time']


# In[351]:


# transform into integer
new_table['days'] = new_table.time_diff.apply(str)
def get_int(day):
  return int(day.split(' ')[0])
new_table['difference'] = new_table.days.apply(get_int)


# In[362]:


# group by agency_name to see if there is a difference between different agencies
by_agency= new_table.difference.groupby(new_table.awarding_toptier_agency_name).mean()
by_agency.hist()
```

In[360]:

```
#agencies with the largest time difference
by_agency.sort_values(ascending=False).head()
```

In[361]:

```
#agencies with the smallest time difference
by_agency.sort_values(ascending=True).head()
```

Is there a obvious finishing time difference between the execution periods of each of the two parties?

In[366]:

```
#sort the table by
sort_table = new_table.sort_values(by=['start_time'],ascending=True).copy()
```

In[402]:

```
#separate the data by the time point 2008-12-31
end_day = datetime.datetime.strptime('2008-12-31',"%Y-%m-%d")
republic_period = sort_table.loc[sort_table['start_time']<=end_day ].copy()
democrat_period = sort_table.loc[sort_table['start_time']>end_day ].copy()
```

In[430]:

```
#plot the distribution of time difference of two parties
plt.hist(republic_period.difference,bins= 10,color='red')
plt.hist(democrat_period.difference,bins = 10,color= 'blue')
plt.title('Difference between Republic Period and Democrat Period')
plt.legend(['republic period','democrat period'])
```

In[544]:

```
#remove negative total obligation
```

```
effect_table = sort_table.loc[sort_table.total_obligation >0].copy()
effect_table.reset_index(inplace=True)


#make numpy arrays
total_ob = np.array(effect_table.total_obligation)
diff = np.array(effect_table.difference)

#make parameters for linear regression
x = total_ob.reshape(-1,1)
y = diff.reshape(-1,1)


#build the regression model
reg = LinearRegression().fit(x, y)


# In[551]:


# see the parameters
print('R^2 is {}\nco-efficient is {}\n intercept is {}\n'.format(reg.score(y,x),reg.coef_,reg.intercept_))


# In[552]:


#get 10 thousands random indexes
import random
index = []
for x in range(100000):
  index.append(random.randint(1,4000000))


# In[553]:


#test the model
test_x = total_ob[index]
test_x = test_x.reshape(-1,1)

test_y = diff[index]
pred_y = reg.predict(test_x)


# In[558]:
```

```
#show the predicting results
plt.scatter(test_x, test_y,  color='black',linewidths=0.0001)
plt.plot(test_x, pred_y, color='blue', linewidth=3)
plt.title('Regression of Time Difference on Total Obligation')
plt.xlabel('Total Obligation')
plt.ylabel('Time Difference')
plt.show()
```

➢ **R Script 1**

```
library(data.table)
library(tidyverse)
library(ggplot2)

selected = fread("~/Desktop/141final/finaldata.csv" )
########################################
#select data has funding in it and remove the negetive transactions
clean_data = selected[selected$funding_toptier_agency_name != "" ]
clean_data = clean_data[clean_data$total_obligation >= 0 ]

#make a contingency table with funding and awarding agency
cont = table(clean_data$funding_toptier_agency_name, clean_data$awarding_toptier_agency_name)

#log the contangency table for better visualization
contingency = log10(cont)
map = as.data.frame(contingency)
map$Freq[map$Freq == -Inf]  = 0       #log 0 is -Inf so I change it to 0


ggplot(data = map, mapping = aes(x = Var1, y = Var2, fill = Freq)) +
    geom_tile()+ xlab(label = "Funding Agency") + ylab(label = "Awarding Agency")+
   scale_fill_gradient(name = "Loged count", low = "#FFFFFF", high = "#012345")+
   theme(axis.text.x = element_text(angle = 44 ,size = 6, hjust = 1),
         axis.text.y = element_text(size = 6))

#sum the obligation based on agency
funding_flow = aggregate(total_obligation~funding_toptier_agency_name, data = clean_data, FUN = sum)

#count transaction for each funding agencies
names_funding = as.data.frame(table(clean_data$funding_toptier_agency_name))
funding_flow$transaction_count = names_funding$Freq
funding_flow$obligation_per_trans = funding_flow$total_obligation / funding_flow$transaction_count

#top 5 angency total amount
head(funding_flow[order(-funding_flow$total_obligation),],5)

#choose not to show the bar for Department of Defense beacuse it's too long, and the other's can't be seen\
bar = funding_flow[order(-funding_flow$total_obligation),][2:10,1:2]
ggplot(bar, aes(x = reorder(funding_toptier_agency_name, -total_obligation), y = total_obligation)) +
```

```
  geom_bar(stat = "identity") + xlab("Agency Names") + ylab("Sum of Obligation")+
  ggtitle("Top 9 total funding agencies(exclude Department of Defense)")+
  theme(axis.text.x = element_text(angle = 40, hjust = 1, size = 8))

#top 5 angency average per transaction
head(funding_flow[order(-funding_flow$obligation_per_trans),], 5)

bar_t = funding_flow[order(-funding_flow$obligation_per_trans),][1:5,c(1,4)]
ggplot(bar_t, aes(x = reorder(funding_toptier_agency_name, -obligation_per_trans), y = obligation_per_trans)) +
 geom_bar(stat = "identity") + xlab("Agency Names") + ylab("Obligation per Transaction")+
 ggtitle("Top 5 Average Transaction Agency")+
 theme(axis.text.x = element_text(angle = 40, hjust = 1))


########################################
#since the department of defense has such big funding, let's study it
defense_department = clean_data[clean_data$funding_toptier_agency_name == "Department of Defense"]
#who do they pay?
table(defense_department$awarding_toptier_agency_name)
```

➢ **R Script 2**

```
path <- "~/Desktop/Winter Quarter 2019/STA 141C/Final/finaldata.csv"
data <- fread(path)

# drop the rows that has no recipient names
data <- data[which(data$recipient_name!=''),]

### find top 10 appearance of recipients: who receive fundings many times
appearance_rep <- table(data$recipient_name)
top10_appearance <- head(appearance_rep[order(appearance_rep, decreasing = T)],10)
top10_appearance

### find top 10 sum obligation by recipient: who receive fundings the most
#top10_sum_data <- data[which(data$recipient_name %in% top10_appearance_names),]
#sum(is.na(top10_sum_data)) # check NaN values
top10_sum_data <- aggregate(data$total_obligation ~ data$recipient_name, data = data, sum)
top10_sum <- head(top10_sum_data[order(top10_sum_data$`data$total_obligation`, decreasing = T),],10)
top10_sum

######## what portions of fundings does each country receive and find the top 10
data_l <- data[which(data$pop_country_name!=""),]
q1 <- aggregate(data_l$total_obligation ~ data_l$pop_country_name, data = data_l, length)
sorted_q1 <- q1[order(q1$`data_l$total_obligation`, decreasing = T),]  # sort q1 by fundings
first10 <- head(sorted_q1,10) # obtain the first 10
first10
```