PART 6 -- Calculation of Median Rate of Annual Turnover

In [1]:
```
import numpy as np
import pandas as pd
```

Read all the *detail.csv.
Renamed "2015Q2-house-disburse-detail.csv" to "2015Q2-house-disburse-detail-old.csv"
Then renamed "2015Q2-house-disburse-detail-updated.csv" to "2015Q2-house-disburse-detail.csv". Then redirected all the filenames to "filename.txt" using the command: ls *detail.csv > filename.txt

In [3]:
```
# Create a list of filename called file_list
# Strip '\n' at the end of the filename
#Ref: https://stackoverflow.com/questions/42488579/
#remove-n-from-each-string-stored-in-a-python-list

file_list = []
with open('filename.txt', 'r', encoding='utf-8') as myfile:
    for line in myfile:
        st_line = line.rstrip()
        file_list.append(st_line)
file_list = file_list[2:30]
print(file_list)
```

```
['2010Q1-house-disburse-detail.csv', '2010Q2-house-disburse-detail.csv
', '2010Q3-house-disburse-detail.csv', '2010Q4-house-disburse-detail.c
sv', '2011Q1-house-disburse-detail.csv', '2011Q2-house-disburse-detail
.csv', '2011Q3-house-disburse-detail.csv', '2011Q4-house-disburse-deta
il.csv', '2012Q1-house-disburse-detail.csv', '2012Q2-house-disburse-de
tail.csv', '2012Q3-house-disburse-detail.csv', '2012Q4-house-disburse-
detail.csv', '2013Q1-house-disburse-detail.csv', '2013Q2-house-disburs
e-detail.csv', '2013Q3-house-disburse-detail.csv', '2013Q4-house-disbu
rse-detail.csv', '2014Q1-house-disburse-detail.csv', '2014Q2-house-dis
burse-detail.csv', '2014Q3-house-disburse-detail.csv', '2014Q4-house-d
isburse-detail.csv', '2015Q1-house-disburse-detail.csv', '2015Q2-house
-disburse-detail.csv', '2015Q3-house-disburse-detail.csv', '2015Q4-hou
se-disburse-detail.csv', '2016Q1-house-disburse-detail.csv', '2016Q2-h
ouse-disburse-detail.csv', '2016Q3-house-disburse-detail.csv', '2016Q4
-house-disburse-detail.csv']
```

In [19]:
```
year = 2010
combined_df = pd.DataFrame(columns = ['BIOGUIDE_ID', 'YEAR'])
for i in range(0, 7):  # represent the years from 2010 to 2016
#Create a dataframe for each of 2016 quarter files and concatenate the 4
    df1 = pd.read_csv(file_list[4*i], low_memory = False)
    df2 = pd.read_csv(file_list[4*i + 1], low_memory = False)
    df3 = pd.read_csv(file_list[4*i + 2], low_memory = False)
    df4 = pd.read_csv(file_list[4*i + 3], low_memory = False)
    df = pd.concat([df1, df2, df3, df4])
    #print(year)
    #Get rows that have only at 'PERSONNEL COMPENSATION' value in 'CATEG
    df = df[df['CATEGORY'] == 'PERSONNEL COMPENSATION']
    #Remove all the rows with 'BIOGUIDE_ID' = NaN
    df = df[df['BIOGUIDE_ID'].notnull()]
    #We want only columns 'BIOGUIDE_ID', 'PAYEE'
    rep_df = df[['BIOGUIDE_ID', 'PAYEE']]
    groupby_rep_df = rep_df.groupby('BIOGUIDE_ID').count()
    # Get the representative who have at staff size of 5 at least
    groupby_rep_df = groupby_rep_df[groupby_rep_df['PAYEE'] >= 5]
    rep_list = groupby_rep_df.index
    #Create a new dataframe with rep_list and year
    new_df = pd.DataFrame({'BIOGUIDE_ID': rep_list })
    new_df['YEAR'] = year
    year = year + 1 #increment year
    #print(new_df)
    combined_df = pd.concat([combined_df, new_df])
```

In [20]:
```
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3344 entries, 0 to 443
Data columns (total 2 columns):
BIOGUIDE_ID    3344 non-null object
YEAR           3344 non-null object
dtypes: object(2)
memory usage: 78.4+ KB
```

In [28]:
```
#Find how many years the representative served
groupby_rep = combined_df.groupby('BIOGUIDE_ID')
groupby_rep_count = groupby_rep.count()
```

In [29]:
```
groupby_rep_count.head()
```

Out[29]:

BIOGUIDE_ID	YEAR
A000014	1
A000022	4
A000055	7
A000210	5
A000358	4

In [30]:
```
groupby_rep_count.columns
```

Out[30]:
```
Index(['YEAR'], dtype='object')
```

In [38]:
```
groupby_rep_count.count()
```

Out[38]:
```
BIOGUIDE_ID    700
dtype: int64
```

In [56]:
```
groupby_rep_count.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 700 entries, A000014 to Z000018
Data columns (total 1 columns):
BIOGUIDE_ID    700 non-null int64
dtypes: int64(1)
memory usage: 10.9+ KB
```

In [57]:
```
groupby_rep_count.index
```

Out[57]:
```
Index(['A000014', 'A000022', 'A000055', 'A000210', 'A000358', 'A000361
',
       'A000362', 'A000363', 'A000364', 'A000365',
       ...
       'W000822', 'Y000031', 'Y000033', 'Y000062', 'Y000063', 'Y000064
',
       'Y000065', 'Y000066', 'Z000017', 'Z000018'],
      dtype='object', name='BIOGUIDE_ID', length=700)
```

In [61]:
```
groupby_rep_count.dtypes
```

Out[61]:
```
BIOGUIDE_ID    int64
dtype: object
```

In [64]:
```
groupby_rep_count = groupby_rep_count.rename(index=str, columns = {"BIOG
groupby_rep_count.index
```

Out[64]:
```
Index(['A000014', 'A000022', 'A000055', 'A000210', 'A000358', 'A000361
',
       'A000362', 'A000363', 'A000364', 'A000365',
       ...
       'W000822', 'Y000031', 'Y000033', 'Y000062', 'Y000063', 'Y000064
',
       'Y000065', 'Y000066', 'Z000017', 'Z000018'],
      dtype='object', name='BIOGUIDE_ID', length=700)
```

In [65]:
```
groupby_rep_count.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 700 entries, A000014 to Z000018
Data columns (total 1 columns):
YEARS_SERVED    700 non-null int64
dtypes: int64(1)
memory usage: 10.9+ KB
```

In [74]:
```
groupby_rep_count = groupby_rep_count[groupby_rep_count['YEARS_SERVED']
groupby_rep_count.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 360 entries, A000055 to Y000064
Data columns (total 1 columns):
YEARS_SERVED    360 non-null int64
dtypes: int64(1)
memory usage: 5.6+ KB
```

In [75]:
```
#Create a list of representatives who have served at least 4 years and h
#a staff of at least 5 every year they served
rep_list = groupby_rep_count.index
print(len(rep_list))
```

```
360
```

In [80]:
```
year = 2010
final_df = pd.DataFrame(columns = ['BIOGUIDE_ID', 'PAYEE', 'YEAR'])
for i in range(0, 7):  # represent the years from 2010 to 2016
#Create a dataframe for each of 2016 quarter files and concatenate the 4
    df11 = pd.read_csv(file_list[4*i], low_memory = False)
    df12 = pd.read_csv(file_list[4*i + 1], low_memory = False)
    df13 = pd.read_csv(file_list[4*i + 2], low_memory = False)
    df14 = pd.read_csv(file_list[4*i + 3], low_memory = False)
    df_concat = pd.concat([df11, df12, df13, df14])

    #Get rows that have only at 'PERSONNEL COMPENSATION' value in 'CATEG
    df_concat = df_concat[df_concat['CATEGORY'] == 'PERSONNEL COMPENSATI
    #Remove all the rows with 'BIOGUIDE_ID' = NaN and PAYEE = Nan
    df_concat = df_concat[df_concat['BIOGUIDE_ID'].notnull()]
    df_concat = df_concat[df_concat['PAYEE'].notnull()]

    #We want only columns 'BIOGUIDE_ID', 'PAYEE'
    rep_payee_df = df_concat[['BIOGUIDE_ID', 'PAYEE']]
    rep_payee_df['YEAR'] = year
    year = year + 1
    rep_payee_df = rep_payee_df.loc[rep_payee_df['BIOGUIDE_ID'].isin(rep
    final_df = pd.concat([final_df, rep_payee_df])
```

```
/Users/Jayashri/anaconda/lib/python3.6/site-packages/ipykernel_launche
r.py:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
(http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-vi
ew-versus-copy)
```

In [81]:
```
final_df.head()
```

Out[81]:

	BIOGUIDE_ID	PAYEE	YEAR
7969	A000055	ABERNATHY, PAMELA M	2010
7970	A000055	BIESZKA,MARK J	2010
7971	A000055	BOWLING,WILSON J	2010
7972	A000055	BROWN,STEPHANIE	2010
7973	A000055	BUSCHING,MARK	2010

In [82]:
```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 225077 entries, 7969 to 76391
Data columns (total 3 columns):
BIOGUIDE_ID    225077 non-null object
PAYEE          225077 non-null object
YEAR           225077 non-null object
dtypes: object(3)
memory usage: 6.9+ MB
```

In [86]:
```
final_df = final_df[['PAYEE', 'YEAR']]
final_df.head()
```

Out[86]:

	PAYEE	YEAR
7969	ABERNATHY, PAMELA M	2010
7970	BIESZKA,MARK J	2010
7971	BOWLING,WILSON J	2010
7972	BROWN,STEPHANIE	2010
7973	BUSCHING,MARK	2010

In [89]:
```
final_df.describe()
```

Out[89]:

	PAYEE	YEAR
count	225077	225077
unique	21621	7
top	ANFINSON, SUSAN	2012
freq	719	35610

In [110]:
```
payee_set_count_list = []  #Number of payees for each year from 2011 to
payee_set_diff_list = []   #Number of payees who left each year from 201
year_list = []
for year in range(2011, 2017):
    # For previous year, i.e. year - 1
    final_df1 = final_df[final_df['YEAR'] == year - 1]
    payee_list1 = final_df1['PAYEE'].tolist()
    payee_set1 =  set(payee_list1)
    # For next year, i.e. year
    final_df2 = final_df[final_df['YEAR'] == year]
    payee_list2 = final_df2['PAYEE'].tolist()
    payee_set2 =  set(payee_list2)
    payee_set_count_list.append(len(payee_set2))
    payee_set_diff_list.append(len(payee_set1 - payee_set2))
    year_list.append(year)
```

In [111]:
```
print(payee_set_count_list)
print(payee_set_diff_list)
print(year_list)
```

```
[7922, 7528, 7596, 7222, 7082, 10866]
[1358, 1982, 1721, 1955, 1731, 2242]
[2011, 2012, 2013, 2014, 2015, 2016]
```

In [112]:
```
# Create a dataframe called turnover_df with columns -- year, payee_coun
turnover_df = pd.DataFrame({'year' : year_list, 'payee_count' : payee_se
                             'payee_left' : payee_set_diff_list})
turnover_df
```

Out[112]:

	payee_count	payee_left	year
0	7922	1358	2011
1	7528	1982	2012
2	7596	1721	2013
3	7222	1955	2014
4	7082	1731	2015
5	10866	2242	2016

In [114]:
```
turnover_df['turnover_rate'] = turnover_df['payee_left']/turnover_df['pa
turnover_df
```

Out[114]:

	payee_count	payee_left	year	turnover_rate
0	7922	1358	2011	0.171421
1	7528	1982	2012	0.263284
2	7596	1721	2013	0.226567
3	7222	1955	2014	0.270701
4	7082	1731	2015	0.244422
5	10866	2242	2016	0.206332

In [115]:
```
#Get median of turnover_rate column
median_rate = turnover_df['turnover_rate'].median()
print(median_rate)
```

```
0.23549454676646503
```

The median annual turnover rate is 23.549454676646503

In [ ]: