DBA 5101

Analytics in Managerial Economics

Estimating the Effect of a Banking Regulation

Ke Ma, A0212524U
Mingzhe Xu, A0232022A
Nanhai Zhong, A0231953E
Xiao Liang, A0232007X

1. Introduction

1.1 The Volcker Rule and its Possible Influences

The Volcker Rule is a part of the Dodd-Frank Act (DFA, July 21, 2010), which was signed into law as an answer to the financial crisis in 2008. It aims to protect banks from non-banking risks by limiting equity investments into or sponsorship of hedge funds, venture capital, and private equity.
The Volcker Rule applies to “any banking entity”. Any insured bank or thrift, any bank holding company or any other company controlling an insured bank or thrift, and any affiliate or subsidiary of such a company will be influenced.

1.2 Problem Statement

In order to analyze the implications the Volcker Rule has on banks’ relevant business models, DiD (Difference in Differences) is used to specifically analyze changes in bank trading asset ratio, which is treated as the main business indicator. In this paper, the regulation is assumed to be the only strong endogenous shock. Also, based on experience, the previous differences between treatment banks and control banks are stable. Therefore, DiD is used in this paper to solve this problem.

2 Data preparation and analysis

Our dataset is constructed on the bank holding companies (BHCs) level provided by the professor. The selected dataset contains one timestamp column, the full set of BHCs (652 individual institutions after cleaning) and selected financial data, such as return on assets. Dataset across quarters from the third quarter of 2004 to the second quarter of 2015, 40 timestamps in total. ***Table 2, Appendix I*** provides the description of all variables in this research and ***Table 3, Appendix I*** exhibits a summary of the dataset.

2.1 Variable definition

Dependent variable To simplify the problem, our research regards all variables referred below with no fixed effect of time or company (*Jussi Keppo*, 2016). *Trading asset ratio* is the major indicator to measure how the Volcker Rule affected the non-banking business. Most BHCs do not have large non-banking business therefore this ratio is lower than 1%. However, for some banks this ratio can be up to 31% ***Table 3, Appendix I.*** For the purpose of splitting control and treated groups, a dummy variable *Affected BHCs* is introduced. As we assume BHCs with trading asset ratio above 3% are likely to be affected, this dummy is one so, zero otherwise.

Explanatory variables and controls *After DFA* variables are created, which is a dummy as well to measure whether the timestamp of a certain record is before or after the law announcement (Q2 2009, when the Obama Administration first announced). Variable *effect* is calculated as the average of trading asset ratio before DFA. We also introduce *affect_pre2007* variable for robustness test as the average trading asset ratio before 2007(Q4 2006). Besides main explanatory, 11 financial features are used as control covariants. They are also used in propensity score calculation by logistic regression as basic financial data that portraits banks’ characters.

2.2 Unbalanced panel data cleaning and imputation

2.2.1 Cleaning

First, delete the banks whose valid data ratio is less than 80% which equals to 32(40 quarters in total) regarding ***Table 2, Appendix I***. After this, the size of the dataset dropped to 25%. Besides data cleaning, there are some banks which have chunks of missing data in some covariates, deleting records of these banks as well. 652 banks are left for further imputation.

2.2.2 Data Imputation

Though we filter out banks with fewer than 32 valid records, some banks still do not have records for all 40 quarters which are unbalanced panel data in collection. To deal with this, we firstly complete the table for all 40 quarters and then use KNN-Imputer from scikit-learn package in python to impute the missing records. As we assume all the variables are neither time fixed nor company fixed, imputation strategy is linear imputing within 40 records of each bank by columns and the imputed values calculation is weighted by distance.

2.2.3 Dataset summary statistics

Here we have the dataset summary ***Table 1, Appendix I***. In brief, after screening the table, the standard deviation and mean of '*Total assets'* is quite huge compared with other features which means for different bank companies, the size is quite different. So, the Volcker Rule might have different effects for these companies.

2.2.4 Propensity score matching

As stated before, dummy variable *Affected BHC* is defined for propensity score matching. To use DiD approach, we assume the baseline bias before and after the event are equivalent. To secure this assumption, we conduct propensity score matching to construct an artificial control group by matching each treated unit with three similar non-treated units of similar characteristics. To avoid propositional effects, we choose the first quarter (Q3 2004) data and set *Affect BHC as* dependent variables, other control variables as independent variables and deploy logistic regression. For each treated band, select three control banks with the closest propensity score.

As shown in ***Table 2 & Table 3, Appendix I***, differences of covariates between treatment and control groups are reduced. Mean difference of covariates after matching becomes less significant. ***Figure 2.1*** shows the treated group and control group(before and after matching) trading asset ratio by quarters. The difference reduction between treated and control group of trading asset ratio after 2008 is more significantly visible after matching. ***Figure 2, Appendix I***

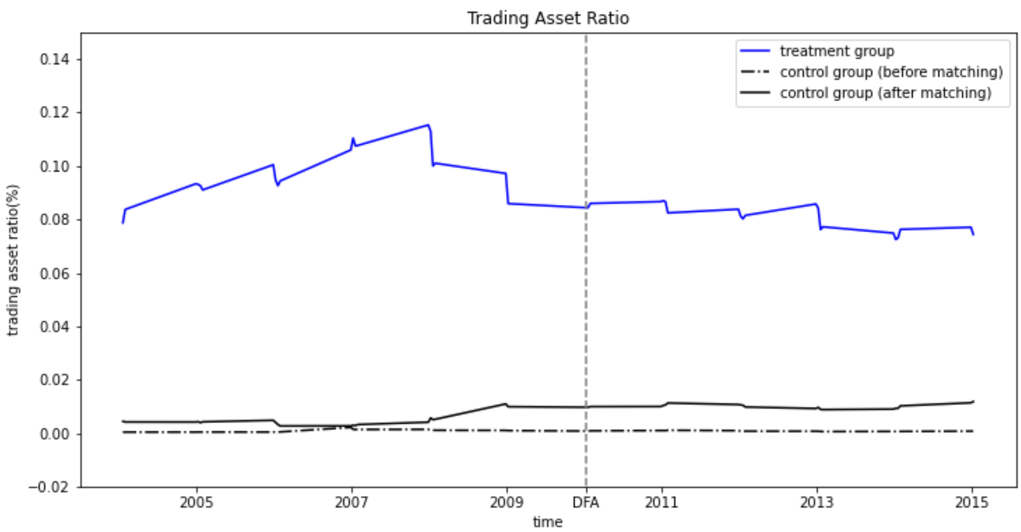


Figure 2.1

3 DiD model and identification

To test the effect of the new regulation, we use the DiD model ***Formula 1, Appendix I*** to scale the effect of the Volcker Rule. Because the baseline bias is not 0 ***Figure 2.1***, RCT is not applicable.

However, the baseline bias is relatively stable, so DiD fits more. With domain knowledge, we could know that the announcement of the new regulation is endogenous. In addition, in order to use DiD model, we still need to confirm that the regulation is the only strong shock, which will be confirmed in the Test 1 and Test 2. Thus, we construct the baseline model containing the interaction term:

$$\mathrm{Y}_{\mathrm{i,t}} = \boldsymbol{\alpha} + \boldsymbol{\beta_1} * \boldsymbol{after\ DFA_t} + \boldsymbol{\beta_2} * \boldsymbol{affected\ BHC_i} + \boldsymbol{\beta_3} * (\boldsymbol{after\ DFA_t} * \boldsymbol{affected\ BHC_i}) + \boldsymbol{X_{i,t}} + \boldsymbol{\epsilon_{i,t}}$$

4 Results and robustness

4.1 The Influence of the New Regulation on the Trading Assets

In the previous article, we observe that the trading assets ratio of the treatment group has a downward trend ***Figure 2.1***. Then, we hypothesize that the banks might decrease their trading assets after the announcement of the new regulation.

In order to confirm the hypothesis, we need to make sure that the trading assets didn't decrease because of other factors accidentally after the announcement of the new regulation. Therefore, in Test 1 and Test 2, the models only contain *after_DFA* but not *affected_BHC* and *after_DFA * affected_BHC.* The result in Test 1 ***Table 1, Appendix II*** shows that the coefficient of *after_DFA* is weakly significant without control variables, which means the time indicator is not a decisive factor. In addition, although the result in Test 2 shows that the coefficient of *after_DFA* is strongly significant with control variables, the R square is very low, which means the time indicator and other control variables can not interpret the decrease in trading assets ratio. In summary, the announcement of the new regulation has a huge effect.

From the results of the Test 3 and Test 4 models ***Table 1, Appendix II***, the coefficients of the interaction term *after_DFA * affected_BHC* are significant and negative. Moreover, the R squares of these two models are higher than in Test 1 and Test 2, which means the models perform better. It shows that the announcement of the new regulation did cause the decrease in the trading assets.

4.2 Differences in Response of Different Companies

The new regulation is intended to prohibit banks from engaging in certain non-banking activities. Hence, those banks who used to have those activities are most affected (which has already been regarded as the treatment group). Meanwhile, theoretically, those banks who are more likely to carry out non-banking activities could also be influenced a lot.

With the PSM, we identify those companies in the control group but with high probability of being affected. At the same time, some banks in the treatment group but with lower probability of being affected have also been identified. Then, regardless of the previous group, we try to separate all the banks (after matching) into two new groups: low probability group *(Propensity Score<0.5)* and high probability group *(Propensity Score>=0.5)* and build the DiD models respectively.

The results ***Table 4, Appendix II*** show that the coefficients of *after_DFA * affected_BHC* in different groups are all significant and negative. More important, the absolute value of the coefficient of *after_DFA * affected_BHC* is bigger in the high probability group, which means the banks with propensity score larger than 0.5 will respond more than the other group.

On the other hand, in the baseline model ***Table 2, Appendix II,*** the coefficients of *dep_roa1(return on assets)* and *dep_creditrisk_total3 (non-performing loan ratio)* are strongly significant with relatively bigger positive value, while *dep_leverage (leverage ratio)* is strongly significant with relatively bigger negative value. To detail this, if some banks have higher return on assets and non-performing loan ratio and lower leverage ratio, they are more likely to have larger scale of trading assets. So, although they may have a similar decrease in trading assets compared to other banks, the impact does not account for a large proportion of their overall value.

To sum up, the banks with higher propensity score, lower return on assets, lower non-performing loan ratio and higher leverage ratio will respond most, vise versa.

4.3 Robustness Analysis and Decision Making

4.3.1 Robustness analysis

In the robustness analysis, we mainly consider two key questions: how does the baseline model perform when baseline bias is lower and whether the model still works well in a more specific situation. To solve the first question, we use the sample after propensity score matching. To solve the second question, we use *affect_pre2007* and *affect* to replace *affected_BHC* in the baseline model to see how they work. *Affect_pre2007* is created to deal with the ***Prepositional Effect***——key information in the market always leaks partially before its announcement. As time goes closer to the announcement date, more banks tend to anticipate the event and respond accordingly. Then, on the event date, the market has fully responded. ***Figure 2, Appendix I***

The results of the robustness analysis show that: Firstly, all coefficients of the interaction term are significant and negative. Secondly, most R squares of models using samples after PSM are better than models using samples before PSM matching. Thirdly, the replacements of *affected_BHC* increase the R-squares, which shows the risk of overfitting.

Table 3, Appendix II

4.3.2 How can banks and the government use the result?

According to the research, banks with a high proportion of trading assets start to comply with the new law by reducing trading assets. Because the law was not fully implemented until late 2017, affected banks sustained their trading assets ratio to 8% in order to maintain profitability and control risks.

When implementing DFA or similar policies to strip banks of non-banking businesses, banks can predict the degree of impact according to our research results and respond by adjusting liabilities, operating costs and deposit reserve ratio, etc. At the same time, the government can assess the impact of relevant policies on the banking sector and warn high-risk enterprises in advance. Moreover, features that we identified above as important identifiers (return on assets, loan ratio, leverage ratio and propensity score) can be reviewed by both banks and the government as red flags of high risks in terms of policy.

Reference

Tom J Pollard, Alistair E W Johnson, Jesse D Raffa, Roger G Mark; tableone: An open source Python package for producing summary statistics for research papers, JAMIA Open, Volume 1, Issue 1, 1 July 2018, Pages 26–31, https://doi.org/10.1093/jamiaopen/ooy012

Chung, Sohhyun and Keppo, Jussi and Yuan, Xuchuan, The Impact of Volcker Rule Bank Profits and Default Probabilities (April 28, 2020). Available at SSRN: https://ssrn.com/abstract=2167773 or http://dx.doi.org/10.2139/ssrn.2167773

Appendix I

1 Data Processing Plot

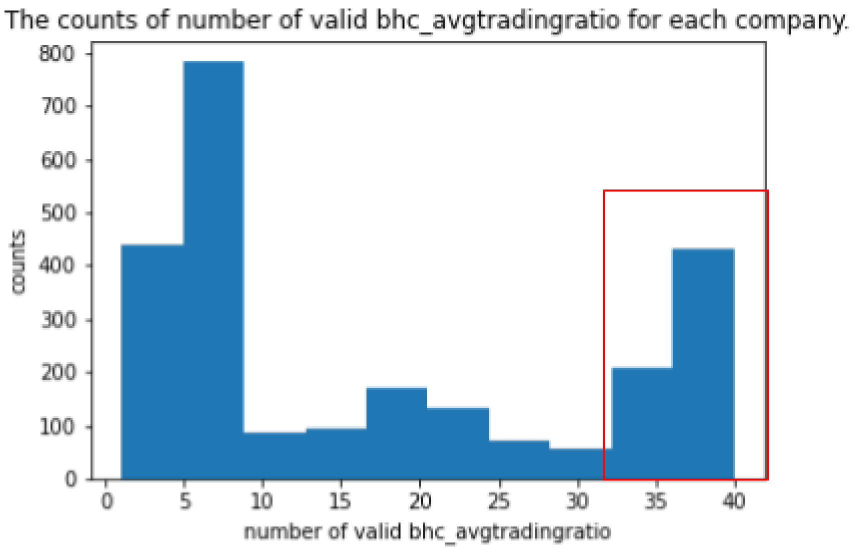


Figure 1

2 Variable description

Variable Table

Variable Name	Explanation
Company code	Banking holding company code
Time	Issue date of the financial records
Trading asset ratio	Ratio of trading assets to total assets
Affected BHC	Dummy variable: takes a value of one if the average trading asset ratio during the pre-DFA period (Q3 2004 - Q2 2009) was equal to or larger than 3%, and zero otherwise
After DFA	Dummy variable that equals one for all quarters between the third quarter of 2010 and the second quarter of 2015, and zero for all quarters from the third quarter of 2004 to the second quarter of 2009
Return on assets	Net operating income divided by average total assets
Leverage ratio	Average equity divided by average total assets
Total assets	Natural logarithm of total assets
Non-performing loan ratio	Past due and nonaccrual loans divided by total loans
Cost-income ratio	Operating expenses divided by total income
Deposit ratio	Average deposits divided by average total assets
Real estate loan ratio	Loans secured by real estate divided by total loans
Liquidity ratio	Cash and balances at other depository institutions divided by total assets
CPP recipient indicator	Capital Purchase Program indicator variable takes one if the bank is a current recipient of CPP funds in a given quarter, and zero otherwise

3 Summary statistic

Table 1: Summary Statistic

Variable	Mean	Min	Max	Std
Dependent variables				
bhc_avgtradingratio	0.002218403	0	0.31179884	0.014953638
Explanatory variable and controls				
dep_roa1	0.002030255	-0.38713744	0.93427694	0.007566278
dep_leverage	0.09408885	-0.13981718	1.1579652	0.037470641
dep_lnassets	14.25606073	5.8888779	21.669949	1.364146511
dep_creditrisk_total3	0.027872589	0	0.38342741	0.030243114
dep_cir	0.550531474	-12.478261	45.933334	0.44802255
dep_depositratio	0.667424369	0	0.99809349	0.121700151
dep_loans_REratio	0.731903865	0	1.0101086	0.155777123
dep_liquidity	0.052765885	0.001810492	0.83480453	0.051555283
dep_cpp_bankquarter	0.085143636	0	1	0.279083642

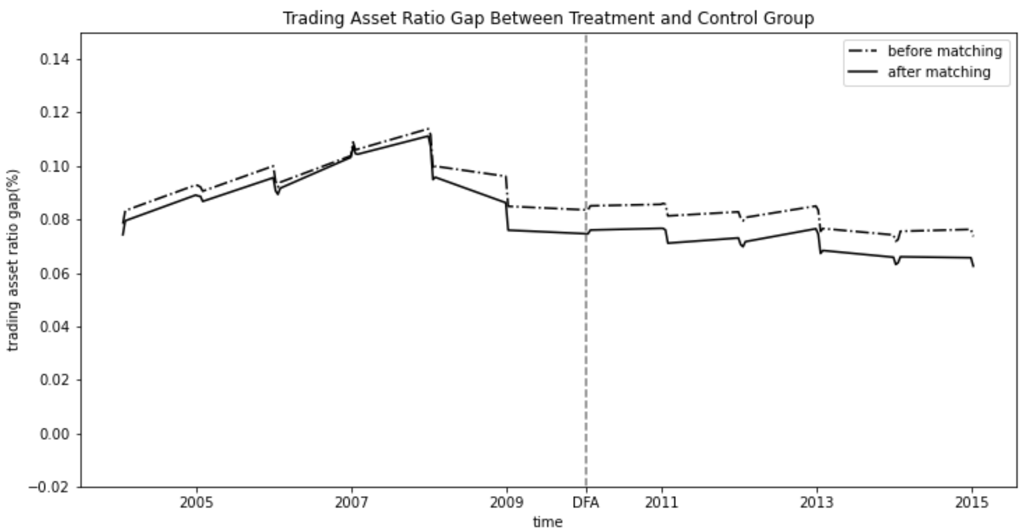


Figure 2

Table 2: Statistics in Q3 2004 before Matching

		Missing	Control	Treated	Diff	P-Value
dep_roa1, mean (SD)		0	0.0 (0.0)	0.0 (0.0)	0.0	0.059
dep_leverage, mean (SD)		0	0.1 (0.0)	0.1 (0.0)	0.0	0.002
dep_lnassets, mean (SD)		0	14.2 (1.2)	18.5 (2.5)	-4.3	<0.001
dep_creditrisk_total3, mean (SD)		0	0.0 (0.0)	0.0 (0.0)	0.0	0.008
dep_cir, mean (SD)		0	0.5 (0.4)	0.7 (1.2)	-0.2	0.033
dep_depositratio, mean (SD)		0	0.7 (0.1)	0.4 (0.2)	0.3	<0.001
dep_loans_REratio, mean (SD)		0	0.7 (0.1)	0.4 (0.2)	0.3	<0.001
dep_liquidity, mean (SD)		0	0.1 (0.1)	0.1 (0.1)	0.0	<0.001
dep_cpp_bankquarter, n (%)	0.0	0	23499 (91.5)	360 (90.0)	23139	0.557
	1.0		2180 (8.5)	40 (10.0)	2140	

Table 3: Statistics in Q3 2004 after Matching

		Missing	Control	Treated	Diff	P-Value
dep_roa1, mean (SD)		0	0.0 (0.0)	0.0 (0.0)	0.0	0.880
dep_leverage, mean (SD)		0	0.1 (0.0)	0.1 (0.0)	0.0	<0.001
dep_lnassets, mean (SD)		0	18.2 (1.9)	18.5 (2.5)	-0.3	0.025
dep_creditrisk_total3, mean (SD)		0	0.0 (0.0)	0.0 (0.0)	0.0	0.148
dep_cir, mean (SD)		0	0.5 (0.3)	0.7 (1.2)	-0.2	0.014
dep_depositratio, mean (SD)		0	0.6 (0.1)	0.4 (0.2)	0.2	<0.001
dep_loans_REratio, mean (SD)		0	0.6 (0.1)	0.4 (0.2)	0.2	<0.001
dep_liquidity, mean (SD)		0	0.0 (0.0)	0.1 (0.1)	-0.1	<0.001
dep_cpp_bankquarter, n (%)	0.0	0	990 (82.5)	360 (90.0)	630	<0.001
	1.0		210 (17.5)	40 (10.0)	170	

Fomula 1

$$\mathrm{ATT}_{\mathrm{DiD}} = E\left[y_{t1,a} - y_{t0,b}\right] - E\left[y_{c0,a} - y_{c0,b}\right] = E[\Delta(y_t)] - E[\Delta(y_c)]$$

$$\widehat{\mathrm{ATT}_{\mathrm{DiD}}} = \frac{1}{N_t}\sum\nolimits_{i\in t}(y_{i,a} - y_{i,b}) - \frac{1}{N_c}\sum\nolimits_{i\in c}(y_{i,a} - y_{i,b})$$
$$= \frac{1}{N_t}\sum\nolimits_{i\in t}\Delta y_i - \frac{1}{N_c}\sum\nolimits_{i\in c}\Delta y_i$$

Other Formular

BASELINE TEST

Test 1:

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \epsilon_{i,t}$$

Test 2:

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + X_{i,t} + \epsilon_{i,t}$$

Test 3:

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \beta_2 *\ affected\ BHC_i + \beta_3 * \ (after\ DFA_t * affected\ BHC_i) + \epsilon_{i,t}$$

Test 4:

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \beta_2 *\ affected\ BHC_i + \beta_3 * \ (after\ DFA_t * affected\ BHC_i) + X_{i,t} + \epsilon_{i,t}$$

ROBUSTNESS TEST

Test A:

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \beta_2 *\ affected\ BHC_i + \beta_3 * \ (after\ DFA_t * affected\ BHC_i) + \epsilon_{i,t}$$

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \beta_2 *\ affected\ BHC_i + \beta_3 * \ (after\ DFA_t * affected\ BHC_i) + X_{i,t} + \epsilon_{i,t}$$

Test B:

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \beta_2 *\ affect + \beta_3 * \ (after\ DFA_t * affect) + \epsilon_{i,t}$$

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \beta_2 *\ affect + \beta_3 * \ (after\ DFA_t * affect) + X_{i,t} + \epsilon_{i,t}$$

Test C:

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \beta_2 *\ affected\_pre2007 + \beta_3 * \ (after\ DFA_t * affected\_pre2007) + \epsilon_{i,t}$$

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \beta_2 *\ affected\_pre2007 + \beta_3 * \ (after\ DFA_t * affected\_pre2007) + X_{i,t} + \epsilon_{i,t}$$

WHICH COMPANY RESPONDS MOST

$$\mathrm{Y_{i,t}} = \alpha + \beta_1 * after\ DFA_t + \beta_2 *\ affected\_BHC + \beta_3 * \ (after\ DFA_t * affected\_BHC) + X_{i,t} + \epsilon_{i,t}$$

Appendix II

1 Baseline model result:

Table 1: Baseline Test

Dependent Variable	Test 1	Test 2	Test 3	Test 4
after_DFA	-0.0004	-0.002	-0.0001	-0.0011
(p value)	*	***	0.4492	***
affect_BHC			0.097	0.0844
(p value)			***	***
after_DFA * affect_BHC			-0.0173	-0.0168
(p value)			***	***
constant	YES	YES	YES	YES
controls	NO	YES	NO	YES
observations	26080	26080	26080	26080
R-square	0	0.272	0.533	0.588

*p<0.1,**p<0.05, ***p<0.01

Table 2: Detailed Results for Baseline Model

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.029	0.0011	-26.2233	0	-0.0311	-0.0268
after_DFA_1	-0.0011	0.0001	-7.8839	0	-0.0013	-0.0008
treat_3_b_avg	0.0844	0.0007	116.3033	0	0.0829	0.0858
after_DFA_1:treat_3_b_avg	-0.0168	0.001	-17.3601	0	-0.0187	-0.0149
dep_roa1	0.0202	0.0083	2.4227	0.0154	0.0039	0.0366
dep_leverage	-0.0184	0.0017	-10.7729	0	-0.0217	-0.015
dep_lnassets	0.0025	0.0001	45.0705	0	0.0023	0.0026
dep_creditrisk_total3	0.0236	0.0021	11.0242	0	0.0194	0.0278
dep_cir	0.0012	0.0002	7.9432	0	0.0009	0.0015
dep_depositratio	-0.0091	0.0006	-15.6338	0	-0.0103	-0.008
dep_loans_REratio	0.003	0.0004	6.8114	0	0.0021	0.0038
dep_liquidity	-0.0018	0.0013	-1.4028	0.1607	-0.0044	0.0007
dep_cpp_bankquarter	0	0.0002	0.0222	0.9823	-0.0004	0.0004

Table 3: Robustness Test

Dependent variable	Test A				Test B				Test C			
after_DFA	-0.0001	0.0047	-0.0011	0.0009	0	0.0048	0	0.0025	0	0.0055	-0.0003	0.0014
(p value)	0.4492	0.0309	***	0.6453	0.4365	***	0.7488	***	0.5911	***	***	0.1028
affect_BHC	0.097	0.0921	0.0844	0.0881								
(p value)	***	***	***	***								
affect					1.0913	1.0981	1.0908	1.1046				
(p value)					***	***	***	***				
affect_pre2007									1.0505	1.043	1.0148	1.0009
(p value)									***	***	***	***
after_DFA * affect_BHC	-0.0173	-0.0221	-0.0168	-0.0192								
(p value)	***	***	***	***								
after_DFA * affect					-0.1826	-0.1962	-0.1831	-0.2078				
(p value)					***	***	***	***				
after_DFA * affect_pre2007									-0.2147	-0.2367	-0.2147	-0.2472
(p value)									***	***	***	***
constant	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
controls	NO	NO	YES	YES	NO	NO	YES	YES	NO	NO	YES	YES
PSM	NO	YES	NO	YES	NO	YES	NO	YES	NO	YES	NO	YES
observations	26080	1600	26080	1600	26080	1600	26080	1600	26080	1600	26080	1600
R-square	0.533	0.464	0.588	0.677	0.91	0.945	0.91	0.947	0.872	0.914	0.876	0.931

*p<0.1,**p<0.05, ***p<0.01

Table 4: Model Results for Different Company Group

Dependent variable	Propensity Score<0.5	Propensity Score>=0.5
after_DFA	-0.0031	-0.0059
(p value)	***	**
affected_BHC	0.0425	0.1242
(p value)	***	***
after_DFA * affected_BHC	-0.0137	-0.0293
(p value)	***	***
constant	YES	YES
controls	YES	YES
observations	960	640
R-square	0.695	0.918

*p<0.1,**p<0.05, ***p<0.01

Group Project 2 (Part I)

Data cleaning, Imputation and propensity matching

In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
raw_df = pd.read_csv('DiD_data.csv',index_col=0)
raw_df.head()
```

Out[3]:

	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_ln
rssd9001							
1020180	20040930	0.0	0	0	0.002772	0.081957	15.
1020180	20041231	0.0	0	0	0.003045	0.082480	15.
1020180	20050331	0.0	0	0	0.002616	0.082074	15.
1020180	20050630	0.0	0	0	0.002647	0.081712	15.
1020180	20050930	0.0	0	0	0.002867	0.082944	15.

1 Data cleaning

Drop companies whose bhc_avg.. number of not-null-value in total is lower than 32 (80%)

In [4]:
```
raw_df.isnull().sum()
```

Out[4]:
```
rssd9999                    0
bhc_avgtradingratio     40118
treat_3_b_avg               0
after_DFA_1                 0
dep_roa1                24622
dep_leverage            24543
dep_lnassets            19789
dep_creditrisk_total3   37128
dep_cir                 39178
dep_depositratio         2388
dep_loans_REratio       37128
dep_liquidity           26401
dep_cpp_bankquarter         0
dtype: int64
```

In [5]:
```
temp = raw_df[raw_df['bhc_avgtradingratio'].isnull() == False]
temp1 = temp.groupby(temp.index).count()
index = temp1[temp1.rssd9999 >= 32].index
df_bhc32 = raw_df[raw_df.index.isin(index)]
```

In [6]:
```
df_bhc32['bhc_avgtradingratio'].max()
```

Out[6]:
```
0.31179884
```

In [7]:
```
df_bhc32.isnull().sum()
```

Out[7]:
```
rssd9999                    0
```

```
bhc_avgtradingratio      1710
treat_3_b_avg               0
after_DFA_1                 0
dep_roa1                 1772
dep_leverage             1738
dep_lnassets              661
dep_creditrisk_total3     851
dep_cir                  1588
dep_depositratio           25
dep_loans_REratio         851
dep_liquidity            2603
dep_cpp_bankquarter         0
dtype: int64
```

Drop company 1246216

In [8]:
```
df_bhc32.groupby(df_bhc32.index).dep_roa1.count().sort_values()
# 1246216 compant's columns, dep_roa1 and dep_leverage data are missing, drop this c
df_bhc32 = df_bhc32.drop(1246216)
```

In [9]:
```
# count() 自动count所有非空值的个数
df_bhc32.isnull().groupby(df_bhc32.index).sum()
```

Out[9]:

	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_ln
rssd9001							
1020180	0	0	0	0	0	0	
1020676	0	2	0	0	2	2	
1020902	0	0	0	0	0	0	
1021682	0	1	0	0	1	1	
1022764	0	0	0	0	0	0	
...	...	...	...	...	...	...	
3274996	0	7	0	0	7	7	
3280988	0	2	0	0	3	2	
3297481	0	3	0	0	3	3	
3309889	0	5	0	0	5	5	
3320978	0	5	0	0	5	5	

652 rows × 13 columns

In [10]:
```
# Empty null left
df_bhc32.isnull().sum()
```

Out[10]:
```
rssd9999                    0
bhc_avgtradingratio      1707
treat_3_b_avg               0
after_DFA_1                 0
dep_roa1                 1736
dep_leverage             1702
dep_lnassets              660
dep_creditrisk_total3     850
dep_cir                  1585
dep_depositratio           25
dep_loans_REratio         850
```

```
dep_liquidity           2600
dep_cpp_bankquarter        0
dtype: int64
```

2 First Imputation Attempt

In [157…

```python
# Impute by company name
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=3, weights="distance")

temp = df_bhc32.copy()
for i in df_bhc32.index.unique():
    temp.loc[i] = imputer.fit_transform(df_bhc32.loc[i])
df_fillna = temp
```

In [158…

```python
# test any nan left
df_fillna.isnull().sum()
```

Out[158…

```
rssd9999                  0
bhc_avgtradingratio       0
treat_3_b_avg             0
after_DFA_1               0
dep_roa1                  0
dep_leverage              0
dep_lnassets              0
dep_creditrisk_total3     0
dep_cir                   0
dep_depositratio          0
dep_loans_RERatio         0
dep_liquidity             0
dep_cpp_bankquarter       0
dtype: int64
```

In [159…

```python
df_fillna
```

Out[159…

	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_
rssd9001							
1020180	20040930.0	0.0	0.0	0.0	0.002772	0.081957	1
1020180	20041231.0	0.0	0.0	0.0	0.003045	0.082480	1
1020180	20050331.0	0.0	0.0	0.0	0.002616	0.082074	1
1020180	20050630.0	0.0	0.0	0.0	0.002647	0.081712	1
1020180	20050930.0	0.0	0.0	0.0	0.002867	0.082944	1
...	...	...	...	...	...	...	
3320978	20050630.0	0.0	0.0	0.0	0.001570	0.093805	1.
3320978	20110331.0	0.0	0.0	1.0	0.001251	0.091727	1.
3320978	20110630.0	0.0	0.0	1.0	0.001243	0.091627	1.
3320978	20150331.0	0.0	0.0	1.0	0.000156	0.101024	1.
3320978	20150630.0	0.0	0.0	1.0	0.000156	0.101025	1.

25900 rows × 13 columns

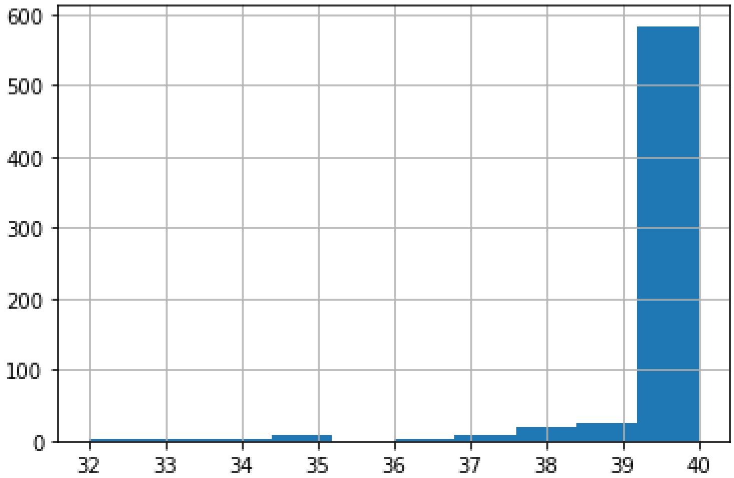
In [161…

```python
# check records number for companies
```

```python
df_fillna.groupby(df_fillna.index).rssd9999.count().hist()
# Almost all company have 40 not-null rows now.
```

Out[161…

<AxesSubplot:>

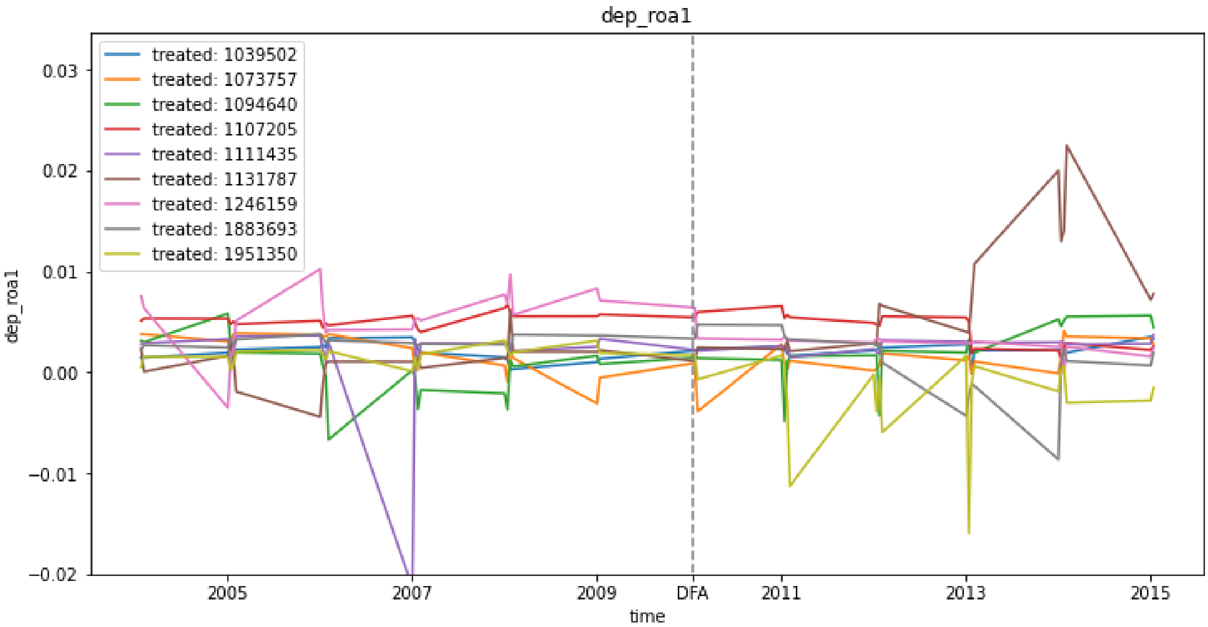


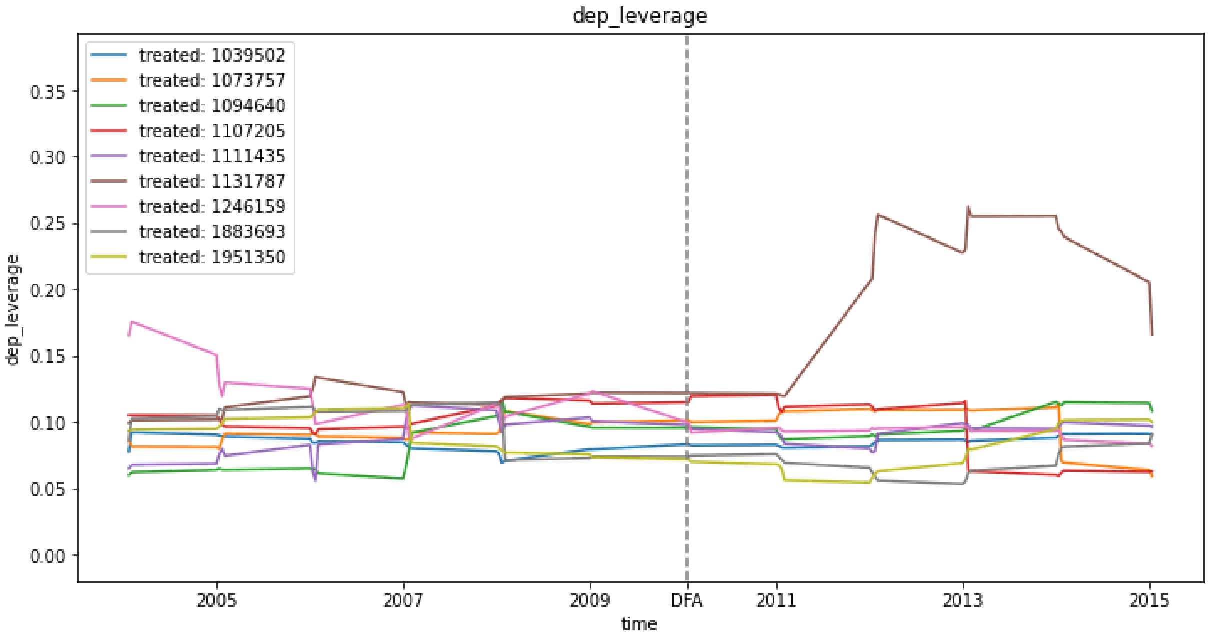
In [162…

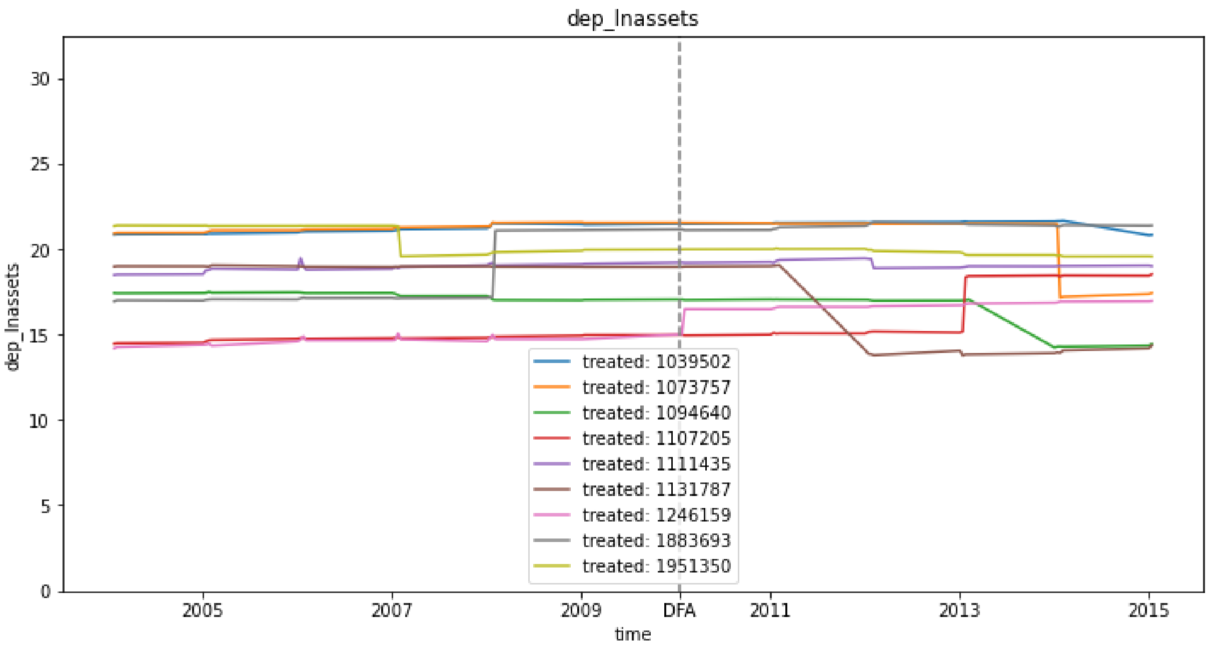
```python
# Out put data to csv
df_fillna.to_csv('fillna_data.csv')
```

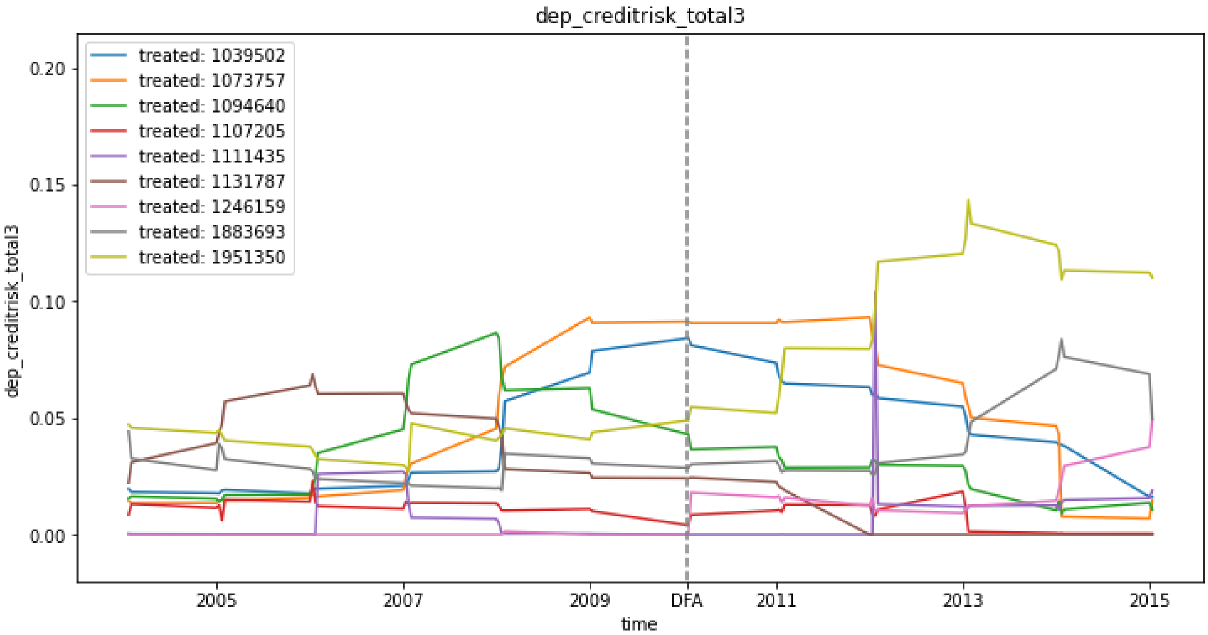
In [165…

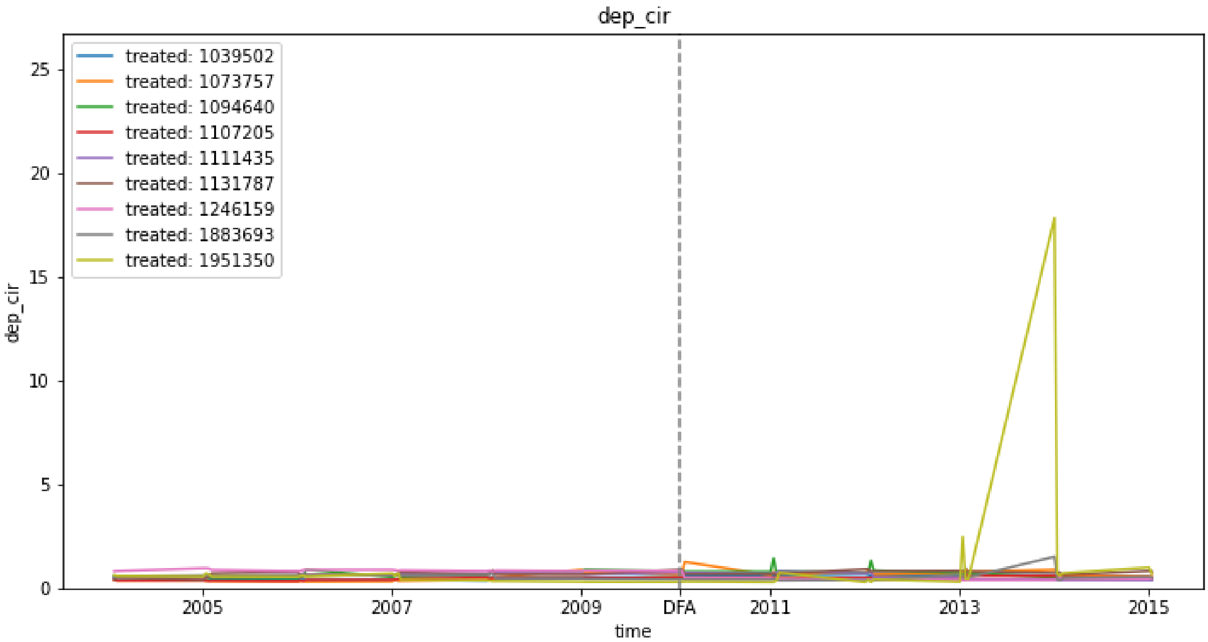
```python
# Identify fixed effect
## All covariates plot for treatment group companies
df1 = df_treated
for column in df_treated.columns[4:-1]:
    feature = df1[column]
    quarter = df_treated["rssd9999"].unique()
    company1 = df1.index.unique()

    y_max = feature.max()
    plt.figure(figsize=(12, 6))
    for i in range(len(company1)-1): # 删去-1
        plt.plot(quarter,feature[i*40:i*40+40],label="treated: "+str(company1[i]))
    plt.plot([20100721,20100721],[-0.1,y_max*1.5],c="grey",linestyle="--")
    plt.ylim(ymin=-0.02,ymax=y_max*1.5)
    plt.xticks(ticks=[20050331,20070331,20090331,20100721,20110331,20130331,20150331
    plt.legend()
    plt.xlabel("time")
    plt.ylabel(column)
    plt.title(column)
    plt.show()
```

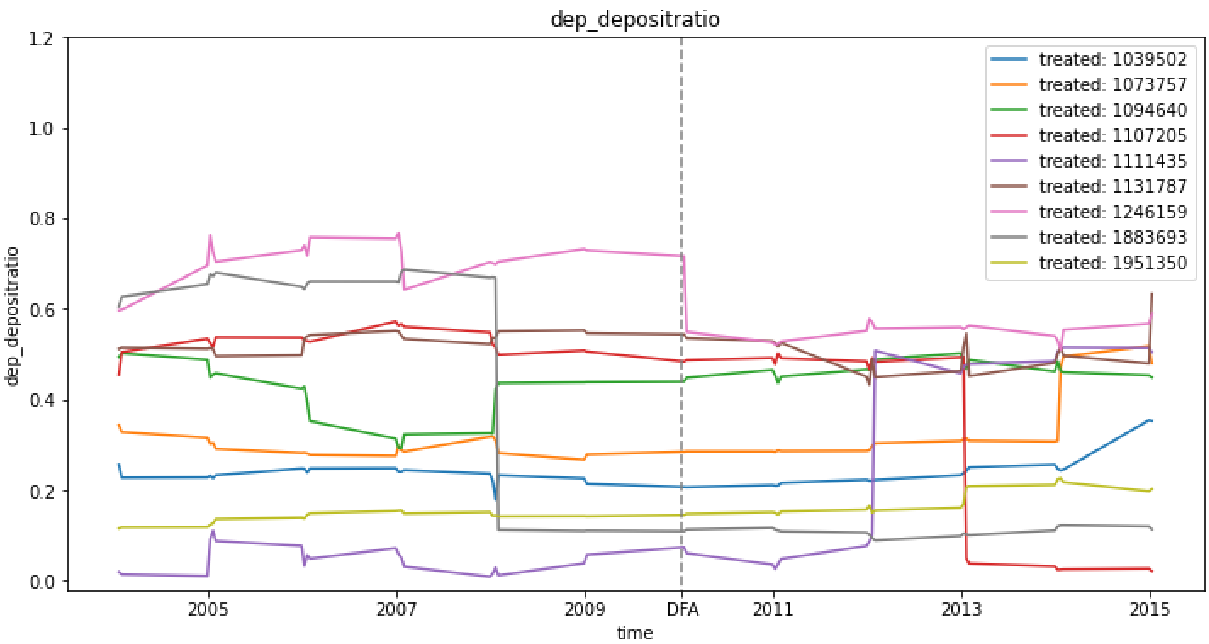

dep_roa1
treated: 1039502
treated: 1073757
treated: 1094640
treated: 1107205
treated: 1111435
treated: 1131787
treated: 1246159
treated: 1883693
treated: 1951350
0.03
0.02
0.01
0.00
−0.01
−0.02
dep_roa1
2005
2007
2009
DFA
2011
2013
2015
time

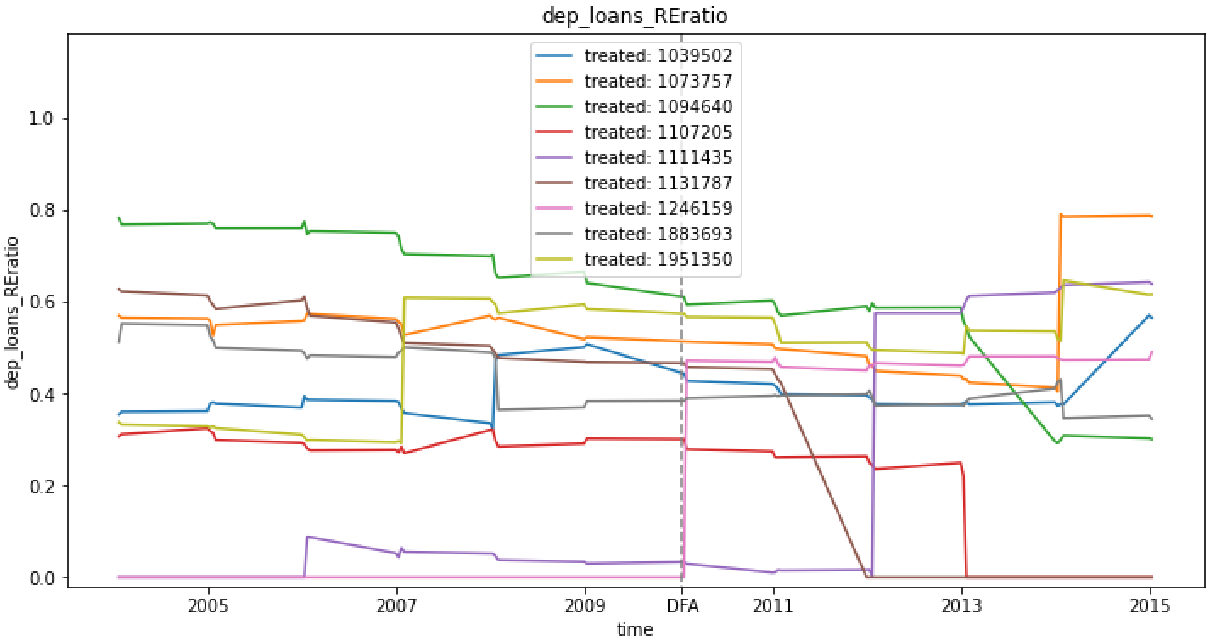

dep_leverage
treated: 1039502
treated: 1073757
treated: 1094640
treated: 1107205
treated: 1111435
treated: 1131787
treated: 1246159
treated: 1883693
treated: 1951350
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
dep_leverage
2005
2007
2009
DFA
2011
2013
2015
time

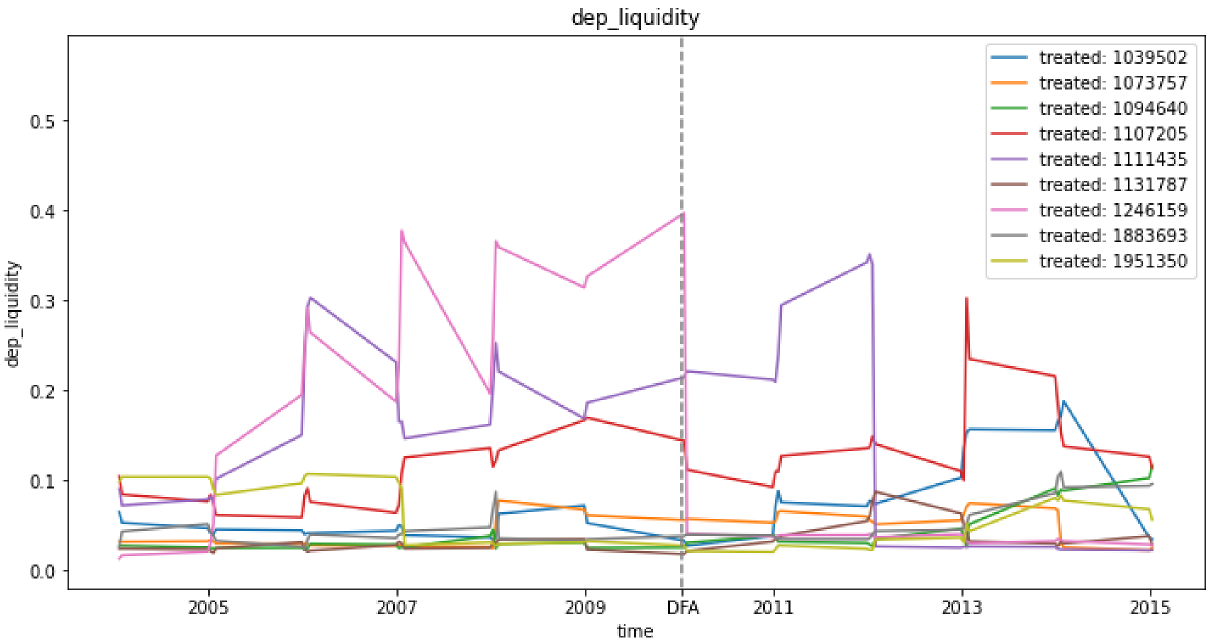

dep_lnassets
treated: 1039502
treated: 1073757
treated: 1094640
treated: 1107205
treated: 1111435
treated: 1131787
treated: 1246159
treated: 1883693
treated: 1951350
30
25
20
15
10
5
0
dep_lnassets
2005
2007
2009
DFA
2011
2013
2015
time


dep_creditrisk_total3
treated: 1039502
treated: 1073757
treated: 1094640
treated: 1107205
treated: 1111435
treated: 1131787
treated: 1246159
treated: 1883693
treated: 1951350
dep_creditrisk_total3
0.20
0.15
0.10
0.05
0.00
2005
2007
2009
DFA
2011
2013
2015
time


dep_cir
treated: 1039502
treated: 1073757
treated: 1094640
treated: 1107205
treated: 1111435
treated: 1131787
treated: 1246159
treated: 1883693
treated: 1951350
dep_cir
25
20
15
10
5
0
2005
2007
2009
DFA
2011
2013
2015
time


dep_depositratio
treated: 1039502
treated: 1073757
treated: 1094640
treated: 1107205
treated: 1111435
treated: 1131787
treated: 1246159
treated: 1883693
treated: 1951350
dep_depositratio
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2005
2007
2009
DFA
2011
2013
2015
time





Specify the methods for imputation according to fixed effect

- After delete all the campanies with fewer than 32 not-null target value records, companies left should be imputed.
- Each of the companies must have 40 records
- Specify the methods for imputation by variable classification we identified above.
  - Time fixed effect: use linear imputation method
  - Company fixed effect: use sample mean imputation method
  - Other variables: use linear imputation method

Find out that no variable is fixed effect

In [169…

```
# create an empty table with complete company codes and quaters
df_spefill = pd.DataFrame(columns = df_bhc32.columns, index = [i for i in df_bhc32.i
df_spefill['rssd9999'] = [j for i in range(652) for j in df_bhc32["rssd9999"].unique
df_spefill.head()
```

Out[169…

	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_ln
1020180	20040930	NaN	NaN	NaN	NaN	NaN	
1020180	20041231	NaN	NaN	NaN	NaN	NaN	
1020180	20050331	NaN	NaN	NaN	NaN	NaN	
1020180	20050630	NaN	NaN	NaN	NaN	NaN	
1020180	20050930	NaN	NaN	NaN	NaN	NaN	

In [170…

```
# reset index
df_spefill = df_spefill.reset_index()
df_bhc32 = df_bhc32.reset_index()
df_spefill = df_spefill.rename({'index': 'rssd9001'},axis = 1)
```

In [171…

```
df_spefill = df_spefill.iloc[:,:2]
```

In [172…

```
# Fill the table above by match up with df_bhc32 by company code and quater value.
df_spefill = df_spefill.merge(df_bhc32, on = ['rssd9001', 'rssd9999'], how = 'left')
df_spefill = df_spefill.set_index('rssd9001')
```

In [173…

```
df_spefill
```

Out[173…

	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_ln
rssd9001							
1020180	20040930	0.0	0.0	0.0	0.002772	0.081957	15.
1020180	20041231	0.0	0.0	0.0	0.003045	0.082480	15.
1020180	20050331	0.0	0.0	0.0	0.002616	0.082074	15.
1020180	20050630	0.0	0.0	0.0	0.002647	0.081712	15.
1020180	20050930	0.0	0.0	0.0	0.002867	0.082944	15.
...	...	...	...	...	...	...	
3320978	20140630	0.0	0.0	1.0	0.001117	0.101113	13.
3320978	20140930	0.0	0.0	1.0	-0.000371	0.102202	13.
3320978	20141231	0.0	0.0	1.0	-0.000236	0.099802	13.
3320978	20150331	NaN	0.0	1.0	NaN	NaN	
3320978	20150630	NaN	0.0	1.0	NaN	NaN	13.

26080 rows × 13 columns

In [174…

```
# Impute by company name
imputer = KNNImputer(n_neighbors=3, weights="distance")
temp = df_spefill.copy()
for i in df_spefill.index.unique():
    temp.loc[i] = imputer.fit_transform(df_spefill.loc[i])
temp
```

Out[174...

	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep
rssd9001							
1020180	20040930.0	0.0	0.0	0.0	0.002772	0.081957	1
1020180	20041231.0	0.0	0.0	0.0	0.003045	0.082480	1
1020180	20050331.0	0.0	0.0	0.0	0.002616	0.082074	1
1020180	20050630.0	0.0	0.0	0.0	0.002647	0.081712	1
1020180	20050930.0	0.0	0.0	0.0	0.002867	0.082944	1
...	...	...	...	...	...	...	
3320978	20140630.0	0.0	0.0	1.0	0.001117	0.101113	1
3320978	20140930.0	0.0	0.0	1.0	-0.000371	0.102202	1
3320978	20141231.0	0.0	0.0	1.0	-0.000236	0.099802	1
3320978	20150331.0	0.0	0.0	1.0	0.000156	0.101024	1
3320978	20150630.0	0.0	0.0	1.0	0.000156	0.101025	1

26080 rows × 13 columns

In [175...

```
df_spefill = temp
df_spefill.to_csv('df_before_PSM.csv')
```

In [197...

```
df_spefill['bhc_avgtradingratio'].loc[df_spefill['rssd9999'] <= 20071231].mean()
```

Out[197...

```
0.002258778045495232
```

Propensity Matching

In [176...

```
df_fillna = df_spefill
```

In [177...

```
from sklearn.linear_model import LogisticRegression
X = df_fillna[df_fillna['rssd9999'] == 20040930].iloc[:,4:]
y = df_fillna[df_fillna['rssd9999'] == 20040930].iloc[:,2]
lr = LogisticRegression()
propen_score = lr.fit(X,y).predict_proba(X)[:,1]
```

In [178...

```
lr.classes_
```

Out[178...

```
array([0., 1.])
```

In [179...

```
df_match = df_fillna[df_fillna['rssd9999'] == 20040930]
df_match['propen_score'] = propen_score
```

```
<ipython-input-179-3ff8d2a6f655>:2: 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: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df_match['propen_score'] = propen_score
```

In [180...

```
df_match
```

Out[180...

rssd9001	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_
1020180	20040930.0	0.0	0.0	0.0	0.002772	0.081957	1
1020676	20040930.0	0.0	0.0	0.0	0.001248	0.048543	1.
1020902	20040930.0	0.0	0.0	0.0	0.002235	0.081153	1
1021682	20040930.0	0.0	0.0	0.0	0.005972	0.065503	1.
1022764	20040930.0	0.0	0.0	0.0	0.002161	0.106747	1
...	...	...	...	...	...	...	
3274996	20040930.0	0.0	0.0	0.0	0.003501	0.077832	1.
3280988	20040930.0	0.0	0.0	0.0	0.003097	0.072055	1.
3297481	20040930.0	0.0	0.0	0.0	0.002234	0.093897	1.
3309889	20040930.0	0.0	0.0	0.0	0.002465	0.080434	1.
3320978	20040930.0	0.0	0.0	0.0	0.001488	0.094232	1.

652 rows × 14 columns

In [181...

```
# return treated companies number and control company number
df_match.groupby('treat_3_b_avg').count()
```

Out[181...

treat_3_b_avg	rssd9999	bhc_avgtradingratio	after_DFA_1	dep_roa1	dep_leverage	dep_lnassets	de
0.0	642	642	642	642	642	642	
1.0	10	10	10	10	10	10	

In [182...

```
df_control = df_match.drop(df_match[df_match['treat_3_b_avg'] == 1].index)
df_treated = df_match[df_match['treat_3_b_avg'] == 1]
```

In [183...

```
df_treated
```

Out[183...

rssd9001	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_
1039502	20040930.0	0.235039	1.0	0.0	0.001450	0.077595	2
1073757	20040930.0	0.122903	1.0	0.0	0.003536	0.091040	2
1094640	20040930.0	0.034627	1.0	0.0	0.004087	0.070220	1
1107205	20040930.0	0.039912	1.0	0.0	0.005250	0.115001	1
1111435	20040930.0	0.038143	1.0	0.0	0.001909	0.063153	1
1131787	20040930.0	0.014920	1.0	0.0	0.003164	0.091551	1
1246159	20040930.0	0.016353	1.0	0.0	0.002966	0.206398	1.
1883693	20040930.0	0.028502	1.0	0.0	0.003358	0.091927	1

	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_
rssd9001							
1951350	20040930.0	0.169691	1.0	0.0	0.003747	0.071185	2
3232316	20040930.0	0.087236	1.0	0.0	0.001762	0.084244	1

In [184…

```
match_control_index = np.empty(1)
for i in df_treated.index:
    print(abs(df_treated.loc[i]['propen_score'] - df_control['propen_score']).sort_v
    match_control_index = np.append(match_control_index, abs(df_treated.loc[i]['prop
match_control_index = match_control_index[1:]
match_control_index
```

```
rssd9001
1120754    0.301453
1119794    0.436904
1132449    0.555485
Name: propen_score, dtype: float64
rssd9001
1120754    0.228146
1119794    0.363596
1132449    0.482178
Name: propen_score, dtype: float64
rssd9001
1027004    0.005074
1078846    0.007484
1249196    0.008306
Name: propen_score, dtype: float64
rssd9001
1094828    0.000002
1029464    0.000013
1102312    0.000024
Name: propen_score, dtype: float64
rssd9001
1119794    0.016361
1120754    0.119089
1132449    0.134943
Name: propen_score, dtype: float64
rssd9001
1132449    0.013651
2277860    0.018764
1070345    0.022795
Name: propen_score, dtype: float64
rssd9001
1136661    0.000046
2490575    0.000302
2706735    0.000344
Name: propen_score, dtype: float64
rssd9001
1049341    0.002019
1020902    0.004399
1027518    0.005827
Name: propen_score, dtype: float64
rssd9001
1120754    0.331769
1119794    0.467220
1132449    0.585801
Name: propen_score, dtype: float64
rssd9001
1120754    0.041367
1119794    0.094084
1132449    0.212665
Name: propen_score, dtype: float64
```

Out[184… array([1120754., 1119794., 1132449., 1120754., 1119794., 1132449.,
       1027004., 1078846., 1249196., 1094828., 1029464., 1102312.,
       1119794., 1120754., 1132449., 1132449., 2277860., 1070345.,
       1136661., 2490575., 2706735., 1049341., 1020902., 1027518.,
       1120754., 1119794., 1132449., 1120754., 1119794., 1132449.])

In [185…

```
df_after_PSM = df_fillna.loc[np.concatenate((df_treated.index.values, match_control_
df_after_PSM
```

Out[185…

	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_
rssd9001							
1039502	20040930.0	0.235039	1.0	0.0	0.001450	0.077595	2
1039502	20041231.0	0.251247	1.0	0.0	0.001451	0.092131	2
1039502	20050331.0	0.254006	1.0	0.0	0.001939	0.090340	2
1039502	20050630.0	0.251873	1.0	0.0	0.000846	0.089686	2
1039502	20050930.0	0.249962	1.0	0.0	0.002129	0.089087	2
...	...	...	...	...	...	...	
1132449	20140630.0	0.004883	0.0	1.0	0.002429	0.151388	1
1132449	20140930.0	0.004372	0.0	1.0	0.001443	0.148722	1
1132449	20141231.0	0.004843	0.0	1.0	0.001485	0.146115	1
1132449	20150331.0	0.005708	0.0	1.0	0.001545	0.143872	1
1132449	20150630.0	0.005523	0.0	1.0	0.001388	0.142639	1

1600 rows × 13 columns

In [186…

```
df_after_PSM.groupby(df_after_PSM.index).count()
# As shown in the table, some control companies are choosen for more than one time.
```

Out[186…

	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_lr
rssd9001							
1020902	40	40	40	40	40	40	
1027004	40	40	40	40	40	40	
1027518	40	40	40	40	40	40	
1029464	40	40	40	40	40	40	
1039502	40	40	40	40	40	40	
1049341	40	40	40	40	40	40	
1070345	40	40	40	40	40	40	
1073757	40	40	40	40	40	40	
1078846	40	40	40	40	40	40	
1094640	40	40	40	40	40	40	
1094828	40	40	40	40	40	40	
1102312	40	40	40	40	40	40	

rssd9001	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	dep_lr
1107205	40	40	40	40	40	40	
1111435	40	40	40	40	40	40	
1119794	200	200	200	200	200	200	
1120754	200	200	200	200	200	200	
1131787	40	40	40	40	40	40	
1132449	240	240	240	240	240	240	
1136661	40	40	40	40	40	40	
1246159	40	40	40	40	40	40	
1249196	40	40	40	40	40	40	
1883693	40	40	40	40	40	40	
1951350	40	40	40	40	40	40	
2277860	40	40	40	40	40	40	
2490575	40	40	40	40	40	40	
2706735	40	40	40	40	40	40	
3232316	40	40	40	40	40	40	

In [187…

```
# output data after propensity score matching. In total, 10 treated companies and 30
df_after_PSM.to_csv('df_after_PSM.csv')
```

Company division by propensity score

In [ ]:

```
# divide in to two groups: PS<0.5 PS>=0.5
treat_cmp = df_treated['propen_score'].sort_values(ascending = True).index.tolist()
treat_cmp11 = df_treated[df_treated.index.isin(treat_cmp[0:6])]
treat_cmp22 = df_treated[df_treated.index.isin(treat_cmp[6:10])]

match_control_index = np.empty(1)
for i in treat_cmp11.index:
    match_control_index = np.append(match_control_index,abs(treat_cmp11.loc[i]['prop
match_control_index = match_control_index[1:]
cmp11 = df_fillna.loc[np.concatenate((treat_cmp11.index.values, match_control_index)
cmp11.to_csv('cmp11.csv')

match_control_index = np.empty(1)
for i in treat_cmp22.index:
    match_control_index = np.append(match_control_index,abs(treat_cmp22.loc[i]['prop
match_control_index = match_control_index[1:]
cmp22 = df_fillna.loc[np.concatenate((treat_cmp22.index.values, match_control_index)
cmp22.to_csv('cmp22.csv')
```

In [3]:
```python
import numpy as np
import pandas as pd
```

In [62]:
```python
df1=pd.read_csv('df_before.csv')
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26080 entries, 0 to 26079
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   rssd9001              26080 non-null  int64
 1   rssd9999              26080 non-null  float64
 2   bhc_avgtradingratio   26080 non-null  float64
 3   treat_3_b_avg         26080 non-null  float64
 4   after_DFA_1           26080 non-null  float64
 5   dep_roa1              26080 non-null  float64
 6   dep_leverage          26080 non-null  float64
 7   dep_lnassets          26080 non-null  float64
 8   dep_creditrisk_total3 26080 non-null  float64
 9   dep_cir               26080 non-null  float64
 10  dep_depositratio      26080 non-null  float64
 11  dep_loans_REratio     26080 non-null  float64
 12  dep_liquidity         26080 non-null  float64
 13  dep_cpp_bankquarter   26080 non-null  float64
dtypes: float64(13), int64(1)
memory usage: 2.8 MB
```

In [5]:
```python
df2=pd.read_csv('df_after.csv')
df2.head()
```

Out[5]:

	rssd9001	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	c
0	1039502	20040930.0	0.235039	1.0	0.0	0.001450	0.077595	
1	1039502	20041231.0	0.251247	1.0	0.0	0.001451	0.092131	
2	1039502	20050331.0	0.254006	1.0	0.0	0.001939	0.090340	
3	1039502	20050630.0	0.251873	1.0	0.0	0.000846	0.089686	
4	1039502	20050930.0	0.249962	1.0	0.0	0.002129	0.089087	

In [6]:
```python
from tableone import TableOne, load_dataset
import pandas as pd
```

In [7]:
```python
data=pd.read_csv('df_before.csv')
columns = ['bhc_avgtradingratio', 'after_DFA_1', 'dep_roa1','dep_leverage','dep_lnas
categorical = ['after_DFA_1','dep_cpp_bankquarter']
groupby = ['treat_3_b_avg']

labels={'treat_3_b_avg': 'treat_3_b_avg'}
mytable1 = TableOne(data, columns=columns, categorical=categorical, groupby=groupby,
print(mytable1.tabulate(tablefmt = "github"))
```

```
|                                  |                     | Missing    | Overall       |
0.0           | 1.0          | P-Value    |
|----------------------------------|---------------------|-----------|--------------|
```

```
--------------|------------|-----------|
| n                                  |                     |            | 26080          |
25680         | 400        |           |
| bhc_avgtradingratio, mean (SD)     |                     | 0          | 0.0 (0.0)      |
0.0 (0.0)     | 0.1 (0.1)  | <0.001    |
| after_DFA_1, n (%)                 | 0.0                 | 0          | 13040 (50.0)   |
12840 (50.0)  | 200 (50.0) | 1.000     |
|                                    | 1.0                 |            | 13040 (50.0)   |
12840 (50.0)  | 200 (50.0) |           |
| dep_roa1, mean (SD)                |                     | 0          | 0.0 (0.0)      |
0.0 (0.0)     | 0.0 (0.0)  | 0.059     |
| dep_leverage, mean (SD)            |                     | 0          | 0.1 (0.0)      |
0.1 (0.0)     | 0.1 (0.0)  | 0.002     |
| dep_lnassets, mean (SD)            |                     | 0          | 14.3 (1.4)     |
14.2 (1.2)    | 18.5 (2.5) | <0.001    |
| dep_creditrisk_total3, mean (SD)   |                     | 0          | 0.0 (0.0)      |
0.0 (0.0)     | 0.0 (0.0)  | 0.008     |
| dep_cir, mean (SD)                 |                     | 0          | 0.6 (0.4)      |
0.5 (0.4)     | 0.7 (1.2)  | 0.033     |
| dep_depositratio, mean (SD)        |                     | 0          | 0.7 (0.1)      |
0.7 (0.1)     | 0.4 (0.2)  | <0.001    |
| dep_loans_RERatio, mean (SD)       |                     | 0          | 0.7 (0.2)      |
0.7 (0.1)     | 0.4 (0.2)  | <0.001    |
| dep_liquidity, mean (SD)           |                     | 0          | 0.1 (0.1)      |
0.1 (0.1)     | 0.1 (0.1)  | <0.001    |
| dep_cpp_bankquarter, n (%)         | 0.0                 | 0          | 23859 (91.5)   |
23499 (91.5)  | 360 (90.0) | 0.557     |
|                                    | 0.5460339955167122  |            | 1 (0.0)        |
1 (0.0)       |            |           |
|                                    | 1.0                 |            | 2220 (8.5)     |
2180 (8.5)    | 40 (10.0)  |           |
```

In [8]:

```python
mytable1.to_excel('mytable1.xlsx')
```

In [10]:

```python
data=pd.read_csv('df_after.csv')
columns = ['bhc_avgtradingratio', 'after_DFA_1', 'dep_roa1','dep_leverage','dep_lnas
categorical = ['after_DFA_1','dep_cpp_bankquarter']
groupby = ['treat_3_b_avg']

labels={'treat_3_b_avg': 'treat_3_b_avg'}
mytable2 = TableOne(data, columns=columns, categorical=categorical, groupby=groupby,
print(mytable2.tabulate(tablefmt = "github"))
```

```
|                                    |     | Missing   | Overall      | 0.0         | 1.
0         | P-Value   |
|------------------------------------|-----|-----------|-------------|------------|---
---------|-----------|
| n                                  |     |           | 1600         | 1200        | 40
0         |           |
| bhc_avgtradingratio, mean (SD)     |     | 0         | 0.0 (0.1)    | 0.0 (0.0)   | 0.
1 (0.1)   | <0.001    |
| after_DFA_1, n (%)                 | 0.0 | 0         | 800 (50.0)   | 600 (50.0)  | 20
0 (50.0)  | 1.000     |
|                                    | 1.0 |           | 800 (50.0)   | 600 (50.0)  | 20
0 (50.0)  |           |
| dep_roa1, mean (SD)                |     | 0         | 0.0 (0.0)    | 0.0 (0.0)   | 0.
0 (0.0)   | 0.880     |
| dep_leverage, mean (SD)            |     | 0         | 0.1 (0.0)    | 0.1 (0.0)   | 0.
1 (0.0)   | <0.001    |
| dep_lnassets, mean (SD)            |     | 0         | 18.3 (2.0)   | 18.2 (1.9)  | 1
8.5 (2.5) | 0.025     |
| dep_creditrisk_total3, mean (SD)   |     | 0         | 0.0 (0.0)    | 0.0 (0.0)   | 0.
0 (0.0)   | 0.148     |
| dep_cir, mean (SD)                 |     | 0         | 0.6 (0.6)    | 0.5 (0.3)   | 0.
7 (1.2)   | 0.014     |
| dep_depositratio, mean (SD)        |     | 0         | 0.5 (0.2)    | 0.6 (0.1)   | 0.
```

```
4 (0.2)  | <0.001    |
| dep_loans_REratio, mean (SD)      |     | 0         | 0.5 (0.2)   | 0.6 (0.1)   | 0.
4 (0.2)  | <0.001    |
| dep_liquidity, mean (SD)          |     | 0         | 0.1 (0.1)   | 0.0 (0.0)   | 0.
1 (0.1)  | <0.001    |
| dep_cpp_bankquarter, n (%)        | 0.0 | 0         | 1350 (84.4) | 990 (82.5) | 36
0 (90.0) | <0.001    |
|                                   | 1.0 |           | 250 (15.6)  | 210 (17.5) | 40
(10.0)  |            |
```

In [ ]:

```
mytable2.to_excel('mytable2.xlsx')
```

In [ ]:

```
from statsmodels.formula.api import ols
regout = ols('bhc_avgtradingratio ~ treat_3_b_avg + after_DFA_1 + treat_3_b_avg*afte
regout.summary2()
```

In [ ]:

```
regout = ols('bhc_avgtradingratio ~ treat_3_b_avg + after_DFA_1 + treat_3_b_avg*afte
regout.summary2()
```

Robust test

In [65]:

```
df11=df1.describe().loc[['mean', 'min', 'max','std']]
df11=df11.drop(['rssd9001','rssd9999'],axis=1)
df11=df11.T
```

In [66]:

```
df11.to_excel('mytable3.xlsx')
```

In [67]:

```
df1['bhc_avgtradingratio'].std
```

Out[67]:

```
<bound method NDFrame._add_numeric_operations.<locals>.std of 0        0.0
1        0.0
2        0.0
3        0.0
4        0.0
        ...
26075    0.0
26076    0.0
26077    0.0
26078    0.0
26079    0.0
Name: bhc_avgtradingratio, Length: 26080, dtype: float64>
```

In [41]:

Out[41]:

	rssd9001	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	c
0	1020180	20040930.0	0.0	0.0	0.0	0.002772	0.081957	
1	1020180	20041231.0	0.0	0.0	0.0	0.003045	0.082480	
2	1020180	20050331.0	0.0	0.0	0.0	0.002616	0.082074	
3	1020180	20050630.0	0.0	0.0	0.0	0.002647	0.081712	
4	1020180	20050930.0	0.0	0.0	0.0	0.002867	0.082944	

Group Project 2 (Part II)

Data Feature and Model Regression

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

Data Feature Before Data Cleaning

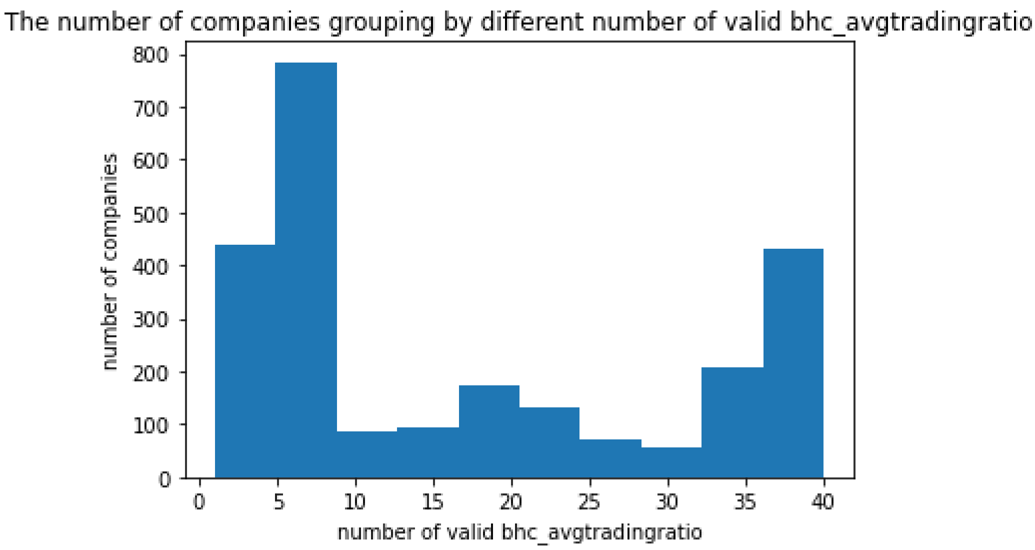
In [2]:

```
df_raw = pd.read_csv("DiD_data1.csv")
rssd = df_raw["rssd9001"].unique()
print(rssd[0:5])
```

```
[1020180. 1020201. 1020340. 1020395. 1020582.]
```

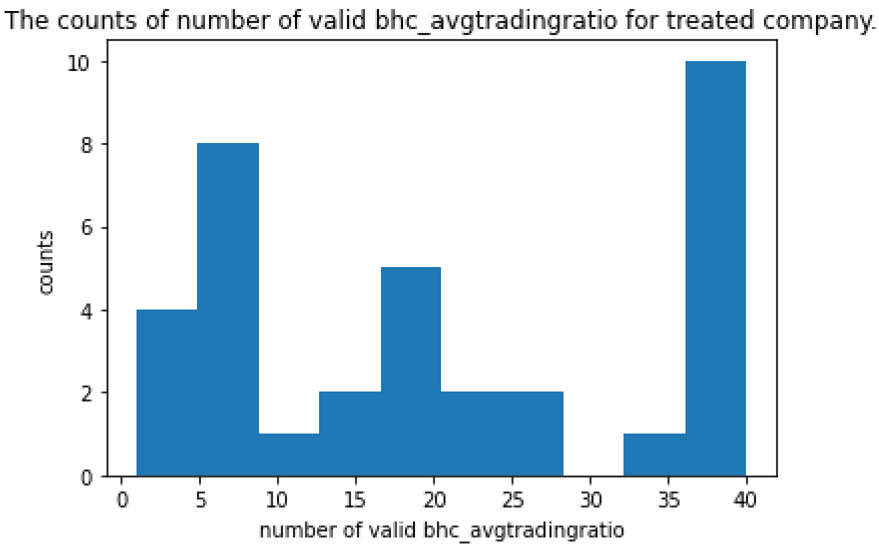
In [8]:

```
plt.hist(df_raw["rssd9001"].value_counts().tolist())
plt.xlabel("number of valid bhc_avgtradingratio")
plt.ylabel("number of companies")
plt.title("The number of companies grouping by different number of valid bhc_avgtradingratio")
plt.savefig("The number of companies grouping by different number of valid bhc_avgtradingratio")
```



In [6]:

```
df = df_raw[df_raw['treat_3_b_avg']==1]
plt.hist(df["rssd9001"].value_counts().tolist())
plt.xlabel("number of valid bhc_avgtradingratio")
plt.ylabel("counts")
plt.title("The counts of number of valid bhc_avgtradingratio for treated company.")
plt.savefig("The counts of number of valid bhc_avgtradingratio for treated company.")
```



In [51]:

```
count = df_raw.groupby("rssd9001").agg('count')
rssd_40 = count[count['rssd9999']==40].index.tolist()
print(rssd_40)
df_40 = df_raw[df_raw["rssd9001"].isin(rssd_40)]
```

```
[1020180.0, 1020902.0, 1022764.0, 1023239.0, 1025309.0, 1025541.0, 1025608.0, 1026
801.0, 1027004.0, 1027518.0, 1029222.0, 1029464.0, 1030170.0, 1030947.0, 1031449.
0, 1032464.0, 1037003.0, 1039502.0, 1048764.0, 1048773.0, 1048812.0, 1048867.0, 10
48894.0, 1049341.0, 1049828.0, 1050646.0, 1050712.0, 1052220.0, 1053272.0, 105349
6.0, 1053580.0, 1054514.0, 1055007.0, 1056161.0, 1058398.0, 1059715.0, 1060328.0,
1060627.0, 1061679.0, 1062621.0, 1064278.0, 1064728.0, 1066209.0, 1066713.0, 10680
25.0, 1068191.0, 1069778.0, 1070345.0, 1070420.0, 1070448.0, 1070578.0, 1070644.0,
1070765.0, 1070804.0, 1070831.0, 1071191.0, 1071276.0, 1071306.0, 1071397.0, 10716
69.0, 1073757.0, 1074156.0, 1075612.0, 1075694.0, 1075984.0, 1076002.0, 1076217.0,
1076262.0, 1076431.0, 1076691.0, 1078529.0, 1078846.0, 1079562.0, 1080595.0, 10811
18.0, 1081239.0, 1081538.0, 1081716.0, 1082067.0, 1082209.0, 1082777.0, 1083783.0,
1085013.0, 1085170.0, 1085509.0, 1086131.0, 1086533.0, 1086654.0, 1090987.0, 10943
14.0, 1094640.0, 1094828.0, 1095674.0, 1096505.0, 1097025.0, 1097089.0, 1097182.0,
1097306.0, 1097566.0, 1097614.0, 1097771.0, 1098303.0, 1098620.0, 1098648.0, 10987
32.0, 1098796.0, 1098844.0, 1099917.0, 1102312.0, 1102367.0, 1103766.0, 1103878.0,
1104231.0, 1104923.0, 1106879.0, 1107205.0, 1107522.0, 1108097.0, 1108163.0, 11083
50.0, 1135972.0]
```

In [55]:

```
df_40_bhc_t_b = df_40[(df_40["bhc_avgtradingratio"]>0) & (df_40["treat_3_b_avg"]==1) & (df_40["a
fter_DFA_1"]==0)]
rssd_40_bhc_t_b =df_40_bhc_t_b["rssd9001"].unique()
d1 = df_40_bhc_t_b[df_40_bhc_t_b["rssd9001"]==rssd_40_bhc_t_b[0]]
print(d1)
plt.plot(range(len(d1)),d1["rssd9999"])
```

	rssd9001	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	\
1451	1039502.0	20040930.0	0.235039	1.0	0.0	
1452	1039502.0	20041231.0	0.251247	1.0	0.0	
1453	1039502.0	20050331.0	0.254006	1.0	0.0	
1454	1039502.0	20050630.0	0.251873	1.0	0.0	
1455	1039502.0	20050930.0	0.249962	1.0	0.0	
1456	1039502.0	20051231.0	0.241241	1.0	0.0	
1457	1039502.0	20060331.0	0.249551	1.0	0.0	
1458	1039502.0	20060630.0	0.257517	1.0	0.0	
1459	1039502.0	20060930.0	0.258023	1.0	0.0	
1460	1039502.0	20061231.0	0.269020	1.0	0.0	
1461	1039502.0	20070331.0	0.280388	1.0	0.0	
1462	1039502.0	20070630.0	0.282337	1.0	0.0	
1463	1039502.0	20070930.0	0.291326	1.0	0.0	
1464	1039502.0	20071231.0	0.293686	1.0	0.0	
1465	1039502.0	20080331.0	0.311799	1.0	0.0	
1466	1039502.0	20080630.0	0.296531	1.0	0.0	
1467	1039502.0	20080930.0	0.249427	1.0	0.0	
1468	1039502.0	20081231.0	0.231852	1.0	0.0	
1469	1039502.0	20090331.0	0.214859	1.0	0.0	
1470	1039502.0	20090630.0	0.206057	1.0	0.0	

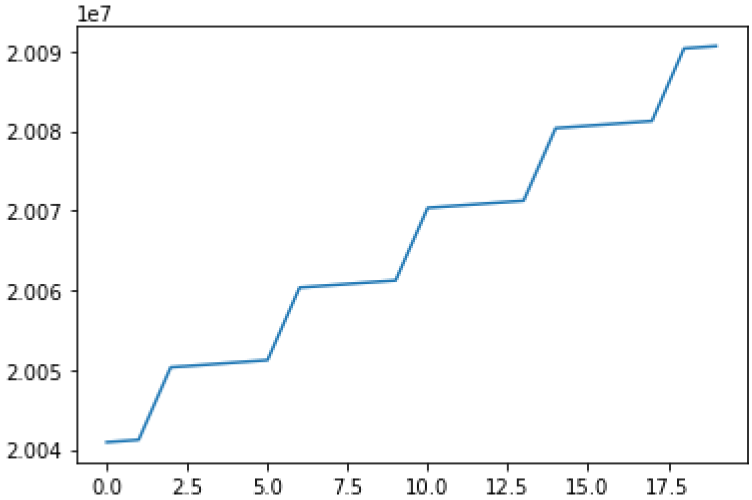
	dep_roa1	dep_leverage	dep_lnassets	dep_creditrisk_total3	dep_cir	\
1451	0.001450	0.077595	20.852951	0.019520	0.559893	
1452	0.001451	0.092131	20.869310	0.018247	0.539865	
1453	0.001939	0.090340	20.887342	0.017743	0.498619	
1454	0.000846	0.089686	20.881365	0.017612	0.581872	
1455	0.002129	0.089087	20.908112	0.018822	0.447572	
1456	0.002246	0.088821	20.904705	0.019156	0.395284	
1457	0.002492	0.087188	20.964863	0.017566	0.413715	
1458	0.002722	0.084197	21.006941	0.018620	0.374949	
1459	0.002473	0.084112	21.014463	0.019761	0.389554	
1460	0.003366	0.085275	21.024496	0.019643	0.362881	
1461	0.003468	0.084586	21.066088	0.020909	0.360539	
1462	0.002954	0.082636	21.100361	0.020833	0.372119	
1463	0.002296	0.081423	21.115021	0.023659	0.341449	
1464	0.001953	0.079954	21.169327	0.026557	0.372885	
1465	0.001481	0.077643	21.219706	0.027068	0.332510	
1466	0.001172	0.075706	21.297443	0.028217	0.461308	
1467	0.000262	0.069285	21.534849	0.038316	0.489998	
1468	0.000317	0.070648	21.500319	0.057120	0.452440	
1469	0.001007	0.079233	21.455244	0.069405	0.452835	
1470	0.001325	0.079146	21.429646	0.078649	0.473000	

	dep_depositratio	dep_loans_REratio	dep_liquidity	dep_cpp_bankquarter
1451	0.256591	0.354119	0.064694	0.0
1452	0.227403	0.360078	0.052704	0.0
1453	0.228123	0.361415	0.046781	0.0
1454	0.231952	0.377931	0.041456	0.0
1455	0.226194	0.380548	0.039657	0.0
1456	0.232841	0.377691	0.045451	0.0
1457	0.247609	0.368695	0.044260	0.0
1458	0.246906	0.394229	0.039956	0.0
1459	0.238976	0.386063	0.041114	0.0
1460	0.247613	0.385896	0.041204	0.0
1461	0.248303	0.383203	0.043788	0.0
1462	0.240614	0.380782	0.050356	0.0
1463	0.240472	0.366978	0.048954	0.0
1464	0.244193	0.357250	0.038842	0.0
1465	0.236185	0.334731	0.036494	0.0
1466	0.220393	0.322525	0.033784	0.0

1467	0.178673	0.471863	0.036255	0.0
1468	0.232767	0.483076	0.062709	0.0
1469	0.225804	0.500646	0.071796	1.0
1470	0.214116	0.506732	0.052391	1.0

Out[55]:

```
[<matplotlib.lines.Line2D at 0x20a5ad28a88>]
```



Data Feature After Data Cleaning

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import make_interp_spline
```

In [6]:

```
df_before_PSM = pd.read_csv("df_before_PSM.csv")
print(len(df_before_PSM))
print(df_before_PSM.head())
```

```
26080
   rssd9001    rssd9999  bhc_avgtradingratio  treat_3_b_avg  after_DFA_1  \
0   1020180  20040930.0                  0.0            0.0          0.0
1   1020180  20041231.0                  0.0            0.0          0.0
2   1020180  20050331.0                  0.0            0.0          0.0
3   1020180  20050630.0                  0.0            0.0          0.0
4   1020180  20050930.0                  0.0            0.0          0.0

   dep_roal  dep_leverage  dep_lnassets  dep_creditrisk_total3   dep_cir  \
0  0.002772      0.081957     15.601202               0.013304  0.463811
1  0.003045      0.082480     15.630583               0.009732  0.456392
2  0.002616      0.082074     15.644925               0.011830  0.444011
3  0.002647      0.081712     15.679702               0.013654  0.433771
4  0.002867      0.082944     15.661868               0.012456  0.400985

   dep_depositratio  dep_loans_REratio  dep_liquidity  dep_cpp_bankquarter
0          0.561805           0.593738       0.024337                  0.0
1          0.557617           0.601763       0.025446                  0.0
2          0.556980           0.600700       0.025153                  0.0
3          0.571642           0.601042       0.023670                  0.0
4          0.577408           0.581438       0.029793                  0.0
```

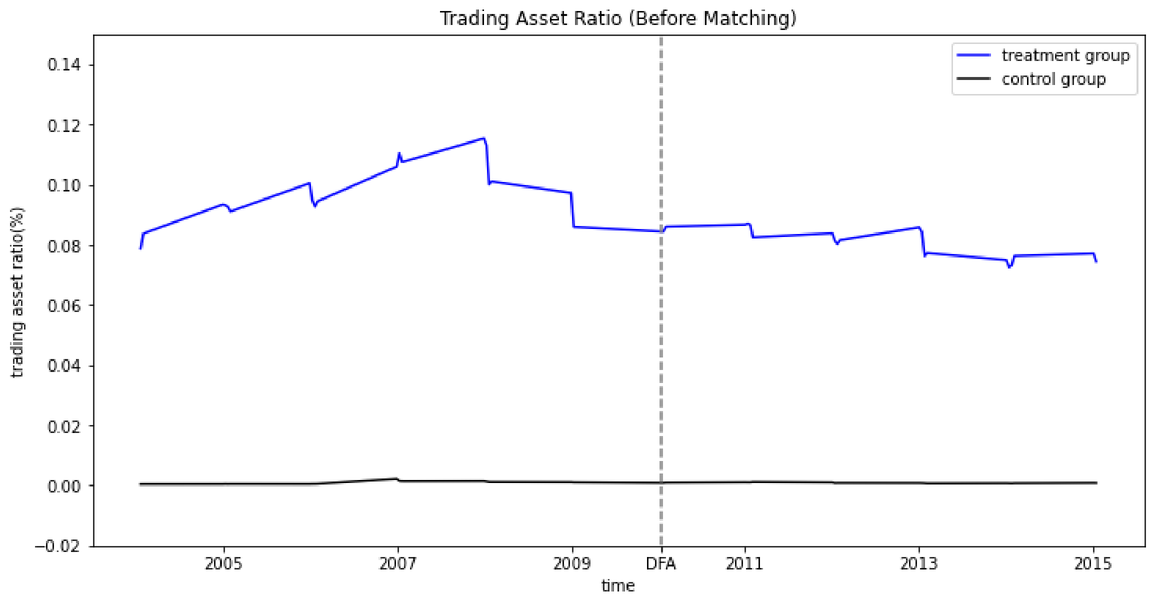
In [5]:

```
df_after_PSM = pd.read_csv("df_after_PSM.csv")
print(len(df_after_PSM))
print(df_after_PSM.head())
```

```
1600
    rssd9001    rssd9999  bhc_avgtradingratio  treat_3_b_avg  after_DFA_1  \
0  1039502.0  20040930.0             0.235039            1.0          0.0
1  1039502.0  20041231.0             0.251247            1.0          0.0
2  1039502.0  20050331.0             0.254006            1.0          0.0
3  1039502.0  20050630.0             0.251873            1.0          0.0
4  1039502.0  20050930.0             0.249962            1.0          0.0

   dep_roal  dep_leverage  dep_lnassets  dep_creditrisk_total3   dep_cir  \
0  0.001450      0.077595     20.852951               0.019520  0.559893
1  0.001451      0.092131     20.869310               0.018247  0.539865
2  0.001939      0.090340     20.887342               0.017743  0.498619
3  0.000846      0.089686     20.881365               0.017612  0.581872
4  0.002129      0.089087     20.908112               0.018822  0.447572

   dep_depositratio  dep_loans_REratio  dep_liquidity  dep_cpp_bankquarter
0          0.256591           0.354119       0.064694                  0.0
1          0.227403           0.360078       0.052704                  0.0
2          0.228123           0.361415       0.046781                  0.0
3          0.231952           0.377931       0.041456                  0.0
4          0.226194           0.380548       0.039657                  0.0
```

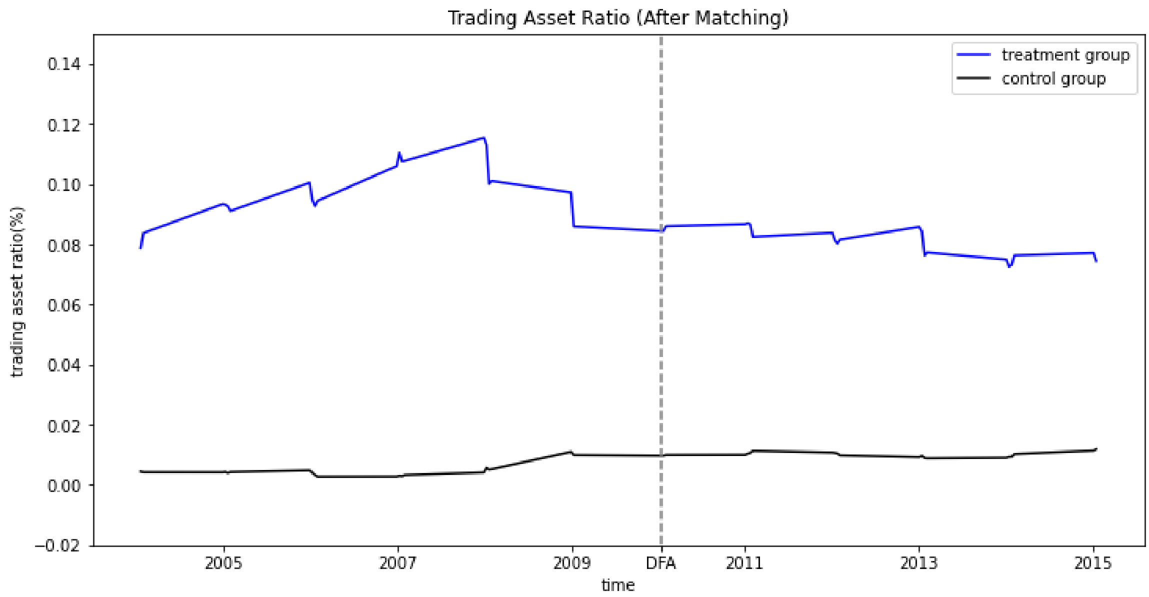
In [42]:

```
## before matching
df0 = df_before_PSM[df_before_PSM["treat_3_b_avg"] == 0]
df1 = df_before_PSM[df_before_PSM["treat_3_b_avg"] == 1]
t_bhc = df1.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
c_bhc = df0.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
quarter = df_before_PSM["rssd9999"].unique()

plt.figure(figsize=(12, 6))
plt.plot(quarter, t_bhc, c="blue", label="treatment group")
plt.plot(quarter, c_bhc, c="black", label="control group")
plt.plot([20100721, 20100721], [-0.1, 0.2], c="grey", linestyle="--")
plt.ylim(ymin = -0.02, ymax=0.15)
plt.xticks(ticks=[20050331, 20070331, 20090331, 20100721, 20110331, 20130331, 20150331], labels=['2005'
,'2007','2009','DFA','2011','2013','2015'])
plt.legend()
plt.xlabel("time")
plt.ylabel("trading asset ratio(%)")
plt.title("Trading Asset Ratio (Before Matching)")
plt.savefig("Trading Asset Ratio (Before Matching)")
plt.show()
```



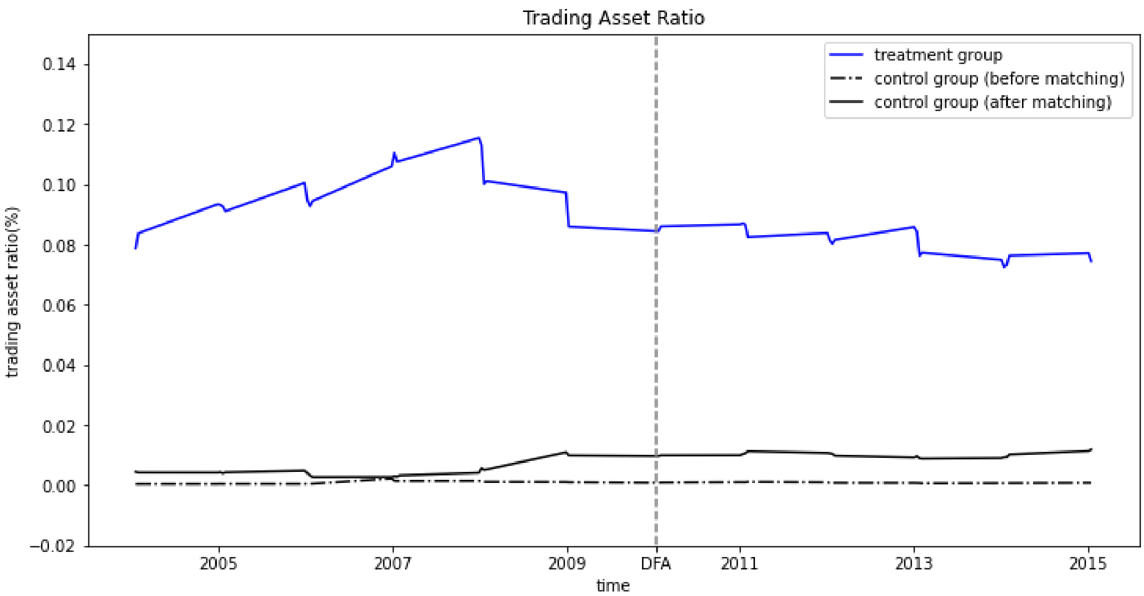
In [45]:

```
## after matching
df0_m = df_after_PSM[df_after_PSM["treat_3_b_avg"] == 0]
df1_m = df_after_PSM[df_after_PSM["treat_3_b_avg"] == 1]
t_bhc_m = df1_m.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
c_bhc_m = df0_m.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
quarter = df_after_PSM["rssd9999"].unique()

plt.figure(figsize=(12, 6))
plt.plot(quarter,t_bhc_m,c="blue",label="treatment group")
plt.plot(quarter,c_bhc_m,c="black",label="control group")
plt.plot([20100721,20100721],[-0.1,0.2],c="grey",linestyle="--")
plt.ylim(ymin = -0.02,ymax=0.15)
plt.xticks(ticks=[20050331,20070331,20090331,20100721,20110331,20130331,20150331],labels=['2005'
,'2007','2009','DFA','2011','2013','2015'])
plt.legend()
plt.xlabel("time")
plt.ylabel("trading asset ratio(%)")
plt.title("Trading Asset Ratio (After Matching)")
plt.savefig("Trading Asset Ratio (After Matching)")
plt.show()
```



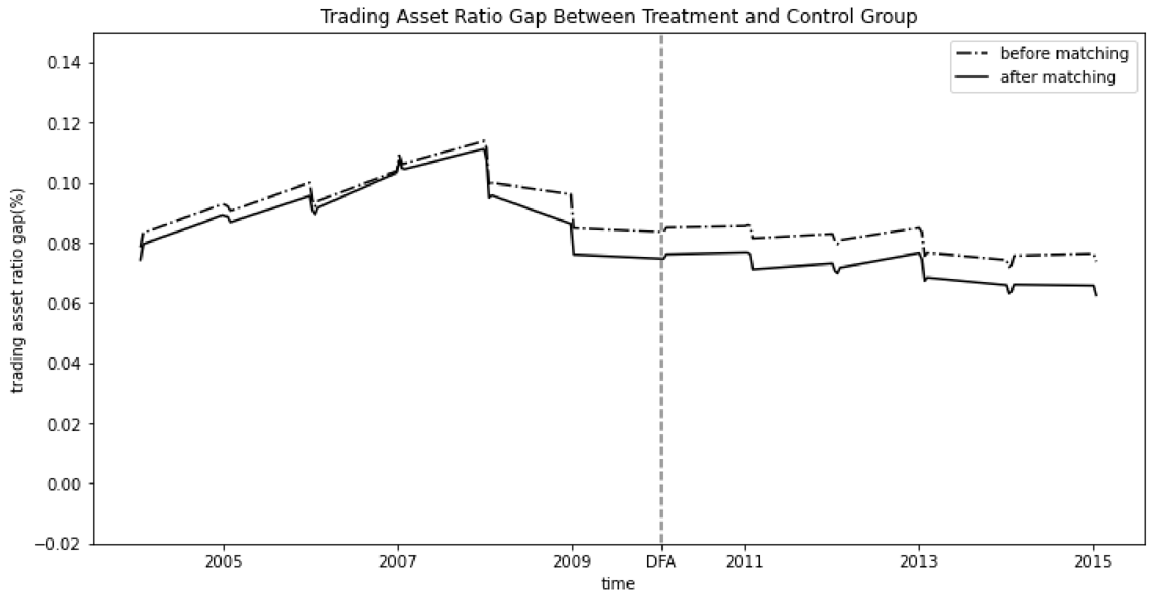
In [46]:

```
## before & after matching
df0 = df_before_PSM[df_before_PSM["treat_3_b_avg"] == 0]
df1 = df_before_PSM[df_before_PSM["treat_3_b_avg"] == 1]
t_bhc = df1.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
c_bhc = df0.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
df0_m = df_after_PSM[df_after_PSM["treat_3_b_avg"] == 0]
df1_m = df_after_PSM[df_after_PSM["treat_3_b_avg"] == 1]
t_bhc_m = df1_m.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
c_bhc_m = df0_m.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
quarter = df_after_PSM["rssd9999"].unique()

plt.figure(figsize=(12, 6))
plt.plot(quarter,t_bhc,c="blue",label="treatment group")
plt.plot(quarter,c_bhc,c="black",linestyle = "-.",label="control group (before matching)")
plt.plot(quarter,c_bhc_m,c="black",label="control group (after matching)")
plt.plot([20100721,20100721],[-0.1,0.2],c="grey",linestyle="--")
plt.ylim(ymin = -0.02,ymax=0.15)
plt.xticks(ticks=[20050331,20070331,20090331,20100721,20110331,20130331,20150331],labels=['2005'
,'2007','2009','DFA','2011','2013','2015'])
plt.legend()
plt.xlabel("time")
plt.ylabel("trading asset ratio(%)")
plt.title("Trading Asset Ratio")
plt.savefig("Trading Asset Ratio")
plt.show()
```



In [47]:

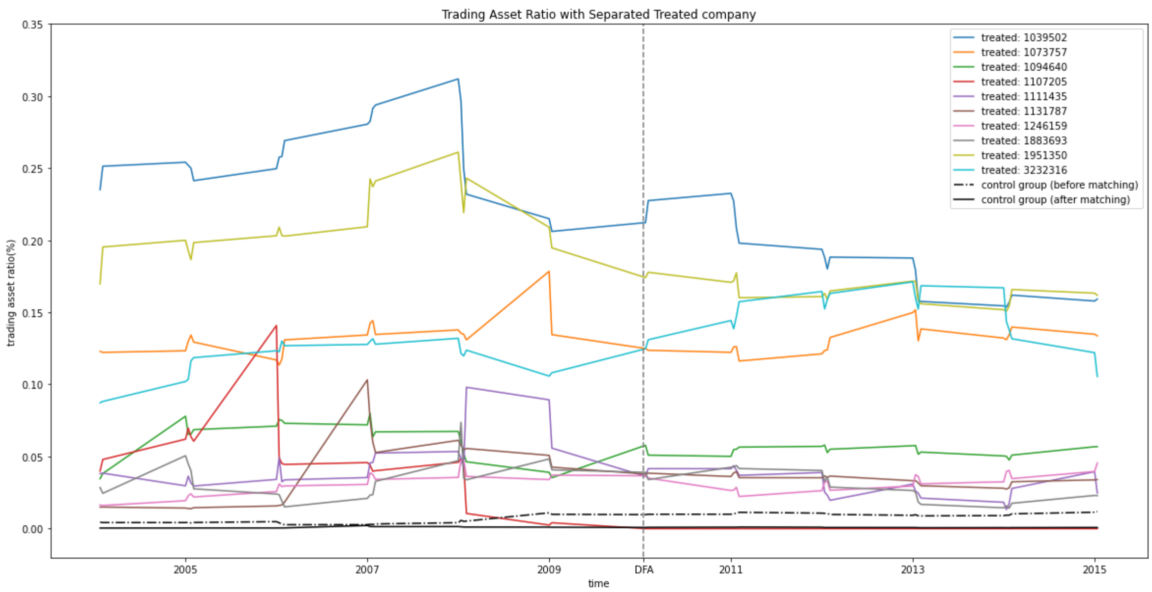
```
# gap changes
gap0 = t_bhc - c_bhc
# gap1 = t_bhc_m - c_bhc_m
# print(t_bhc)
plt.figure(figsize=(12, 6))
plt.plot(quarter,gap0,c="black",linestyle = "-.",label="before matching")
plt.plot(quarter,gap1,c="black",label="after matching")
plt.ylim(ymin = -0.02,ymax=0.15)
plt.plot([20100721,20100721],[-0.1,0.2],c="grey",linestyle="--")
plt.xticks(ticks=[20050331,20070331,20090331,20100721,20110331,20130331,20150331],labels=['2005'
,'2007','2009','DFA','2011','2013','2015'])
plt.xlabel("time")
plt.ylabel("trading asset ratio gap(%)")
plt.title("Trading Asset Ratio Gap Between Treatment and Control Group")
plt.legend()
plt.savefig("Trading Asset Ratio Gap Between Treatment and Control Group")
plt.show()
```



In [48]:

```
## separate treated company
df0 = df_before_PSM[df_before_PSM["treat_3_b_avg"] == 0]
df0_m = df_after_PSM[df_after_PSM["treat_3_b_avg"] == 0]
df1 = df_before_PSM[df_before_PSM["treat_3_b_avg"] == 1]
df1_m = df_after_PSM[df_before_PSM["treat_3_b_avg"] == 1]
t_bhc = df1.groupby(["rssd9001","rssd9999"])["bhc_avgtradingratio"].mean()
t_bhc_m = df1_m.groupby(["rssd9001","rssd9999"])["bhc_avgtradingratio"].mean()
c_bhc = df0.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
c_bhc_m = df0_m.groupby(["rssd9999"])["bhc_avgtradingratio"].mean()
# t_bhc_count = df1.groupby(["rssd9001"])["rssd9999"].count()
quarter = df_before_PSM["rssd9999"].unique()
company1 = df1["rssd9001"].unique()
# print(t_bhc_count)
# print(len(t_bhc))
# print(len(company1))

plt.figure(figsize=(20, 10))
for i in range(len(company1)):
    plt.plot(quarter,t_bhc[i*40:i*40+40],label="treated: "+str(company1[i]))
plt.plot(quarter,c_bhc_m,c="black",linestyle = "-.",label="control group (before matching)")
plt.plot(quarter,c_bhc,c="black",linestyle = "-",label="control group (after matching)")
plt.plot([20100721,20100721],[-0.1,0.4],c="grey",linestyle="--")
plt.ylim(ymin=-0.02,ymax=0.35)
plt.xticks(ticks=[20050331,20070331,20090331,20100721,20110331,20130331,20150331],labels=['2005'
,'2007','2009','DFA','2011','2013','2015'])
plt.legend()
plt.xlabel("time")
plt.ylabel("trading asset ratio(%)")
plt.title("Trading Asset Ratio with Separated Treated company")
plt.savefig("Trading Asset Ratio with Separated Treated company")
plt.show()
```

```
C:\Users\DELL\AppData\Roaming\Python\Python37\site-packages\ipykernel_launcher.py:
5: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  """
```



In [148]:

```
quarter_change = pd.DataFrame(df["rssd9001"].unique(),columns={"rssd9001"})
dep_name = ["dep_roa1","dep_leverage","dep_lnassets","dep_creditrisk_total3","dep_cir","dep_depo
sitratio","dep_loans_REratio","dep_liquidity","dep_cpp_bankquarter"]
quarter_mean_std = []

for i in range(len(dep_name)):
    group_mean = df.groupby(["rssd9001"])[dep_name[i]].mean()
    group_std = df.groupby(["rssd9001"])[dep_name[i]].std()
    quarter_change[dep_name[i]+"_mean"] = group_mean.tolist()
    quarter_change[dep_name[i]+"_std"] = group_std.tolist()
    quarter_change[dep_name[i]+"_mean/std"] = quarter_change[dep_name[i]+"_mean"]/ quarter_chang
e[dep_name[i]+"_std"]
    quarter_mean_std.append(np.mean(quarter_change[dep_name[i]+"_mean/std"]))

print(quarter_change)
print(quarter_mean_std)
```

```
     rssd9001  dep_roa1_mean  dep_roa1_std  dep_roa1_mean/std  \
0     1020180       0.002606      0.000368           7.090959
1     1020676       0.001147      0.001034           1.108885
2     1020902       0.001947      0.001616           1.204805
3     1021682       0.004587      0.001562           2.937073
4     1022764       0.001487      0.007436           0.199981
..        ...            ...           ...                ...
647   3274996      -0.000697      0.003422          -0.203644
648   3280988       0.002050      0.002548           0.804685
649   3297481       0.000853      0.002769           0.308098
650   3309889       0.001290      0.001129           1.142303
651   3320978       0.000929      0.000675           1.375479

     dep_leverage_mean  dep_leverage_std  dep_leverage_mean/std  \
0             0.092624          0.007609              12.172140
1             0.050215          0.004718              10.642694
2             0.087760          0.010112               8.678355
3             0.080450          0.011729               6.859013
4             0.114385          0.026739               4.277903
..                 ...               ...                    ...
647           0.055080          0.021153               2.603878
648           0.095772          0.020689               4.629046
649           0.095810          0.005312              18.035113
650           0.102238          0.011991               8.525938
651           0.094219          0.004719              19.967319

     dep_lnassets_mean  dep_lnassets_std  dep_lnassets_mean/std  ...  \
0            15.850387          0.131358             120.665350  ...
1            13.542366          0.109572             123.593496  ...
2            16.522048          0.114909             143.783358  ...
3            13.278138          0.158557              83.743873  ...
4            15.397809          0.113571             135.578549  ...
..                 ...               ...                    ...  ...
647          13.232869          0.259220              51.048737  ...
648          13.764835          0.406646              33.849637  ...
649          13.926375          0.084137             165.519804  ...
650          13.309338          0.109216             121.862399  ...
651          13.445053          0.103883             129.425501  ...

     dep_depositratio_mean/std  dep_loans_REratio_mean  dep_loans_REratio_std  \
0                    18.718127                0.608275               0.016155
1                    10.669983                0.695834               0.040358
2                    27.929902                0.385171               0.068371
3                    11.136061                0.716535               0.027208
4                    16.102885                0.805067               0.051688
..                         ...                     ...                    ...
647                  18.483324                0.902087               0.022366
648                   2.641324                0.432518               0.117917
649                  15.685670                0.951462               0.012307
650                  22.263053                0.948699               0.006308
651                  24.438418                0.847947               0.023519

     dep_loans_REratio_mean/std  dep_liquidity_mean  dep_liquidity_std  \
0                     37.652485            0.048482           0.030293
1                     17.241345            0.167067           0.139485
2                      5.633565            0.069253           0.018998
3                     26.335213            0.042515           0.015309
4                     15.575497            0.043746           0.050790
..                          ...                 ...                ...
647                   40.333388            0.027630           0.012769
648                    3.667981            0.027193           0.009864
```

```
649                  77.313475           0.019961           0.006480
650                 150.401515           0.019150           0.005507
651                  36.053073           0.066403           0.033873

     dep_liquidity_mean/std  dep_cpp_bankquarter_mean  \
0                  1.600425                  0.000000
1                  1.197744                  0.000000
2                  3.645259                  0.000000
3                  2.777143                  0.000000
4                  0.861318                  0.100000
..                      ...                       ...
647                2.163785                  0.000000
648                2.756726                  0.081081
649                3.080566                  0.000000
650                3.477505                  0.000000
651                1.960362                  0.000000

     dep_cpp_bankquarter_std  dep_cpp_bankquarter_mean/std
0                   0.000000                           NaN
1                   0.000000                           NaN
2                   0.000000                           NaN
3                   0.000000                           NaN
4                   0.303822                      0.329140
..                       ...                           ...
647                 0.000000                           NaN
648                 0.276725                      0.293003
649                 0.000000                           NaN
650                 0.000000                           NaN
651                 0.000000                           NaN

[652 rows x 28 columns]
[2.204778516906602, 9.879335796707878, 89.31864796482411, 1.9405069936796333, 5.00
0216168476343, 16.395761963604617, 29.454430279520796, 2.6576140491123437, 0.59191
90154855003]
```

In [151]:

```
company_change = pd.DataFrame(df["rssd9999"].unique(),columns={"rssd9999"})
dep_name = ["dep_roa1","dep_leverage","dep_lnassets","dep_creditrisk_total3","dep_cir","dep_depo
sitratio","dep_loans_REratio","dep_liquidity","dep_cpp_bankquarter"]
company_mean_std = []

for i in range(len(dep_name)):
    group_mean = df.groupby(["rssd9999"])[dep_name[i]].mean()
    group_std = df.groupby(["rssd9999"])[dep_name[i]].std()
    company_change[dep_name[i]+"_mean"] = group_mean.tolist()
    company_change[dep_name[i]+"_std"] = group_std.tolist()
    company_change[dep_name[i]+"_mean/std"] = company_change[dep_name[i]+"_mean"]/ company_chang
e[dep_name[i]+"_std"]
    company_mean_std.append(np.mean(company_change[dep_name[i]+"_mean/std"]))

print(company_change)
print(company_mean_std)
```

	rssd9999	dep_roa1_mean	dep_roa1_std	dep_roa1_mean/std	\
0	20040930.0	0.002950	0.001981	1.489408	
1	20041231.0	0.002869	0.002296	1.249325	
2	20050331.0	0.003045	0.003594	0.847248	
3	20050630.0	0.003147	0.003098	1.015797	
4	20050930.0	0.003122	0.002520	1.239092	
5	20051231.0	0.002869	0.002514	1.141253	
6	20060331.0	0.003001	0.003188	0.941191	
7	20060630.0	0.003143	0.003303	0.951600	
8	20060930.0	0.003073	0.002563	1.199185	
9	20061231.0	0.002730	0.002355	1.159430	
10	20070331.0	0.002729	0.003333	0.818995	
11	20070630.0	0.002799	0.002555	1.095590	
12	20070930.0	0.002715	0.002581	1.052225	
13	20071231.0	0.001875	0.003644	0.514584	
14	20080331.0	0.002305	0.003666	0.628740	
15	20080630.0	0.001582	0.005143	0.307543	
16	20080930.0	0.000697	0.007052	0.098795	
17	20081231.0	-0.000788	0.006443	-0.122294	
18	20090331.0	0.000920	0.004003	0.229860	
19	20090630.0	0.000052	0.005121	0.010227	
20	20100930.0	0.000913	0.005208	0.175238	
21	20101231.0	0.000004	0.005462	0.000795	
22	20110331.0	0.001254	0.003422	0.366564	
23	20110630.0	0.001206	0.003585	0.336344	
24	20110930.0	0.001471	0.003537	0.415783	
25	20111231.0	0.000575	0.005318	0.108220	
26	20120331.0	0.001960	0.002634	0.744044	
27	20120630.0	0.001964	0.002690	0.730246	
28	20120930.0	0.001947	0.003226	0.603497	
29	20121231.0	0.001688	0.004869	0.346684	
30	20130331.0	0.002131	0.002627	0.811268	
31	20130630.0	0.002342	0.003388	0.691297	
32	20130930.0	0.002298	0.003179	0.722702	
33	20131231.0	0.002420	0.008413	0.287693	
34	20140331.0	0.003543	0.036764	0.096367	
35	20140630.0	0.001751	0.015615	0.112143	
36	20140930.0	0.002320	0.002826	0.821075	
37	20141231.0	0.002253	0.003894	0.578594	
38	20150331.0	0.002551	0.006722	0.379590	
39	20150630.0	0.002388	0.002176	1.097715	

	dep_leverage_mean	dep_leverage_std	dep_leverage_mean/std	\
0	0.089923	0.029436	3.054798	
1	0.090714	0.030360	2.987957	
2	0.090374	0.031424	2.875975	
3	0.090424	0.031823	2.841453	
4	0.090330	0.031291	2.886810	
5	0.089559	0.030564	2.930228	
6	0.089174	0.030574	2.916631	
7	0.088891	0.031397	2.831160	
8	0.089839	0.031570	2.845660	
9	0.091037	0.031867	2.856747	
10	0.091310	0.031635	2.886341	
11	0.091265	0.031625	2.885818	
12	0.091695	0.031850	2.878945	
13	0.092133	0.031556	2.919610	
14	0.091831	0.031079	2.954714	
15	0.090749	0.031040	2.923605	
16	0.088946	0.030965	2.872511	
17	0.088608	0.030052	2.948500	

18	0.090004	0.029051	3.098134
19	0.090884	0.028775	3.158457
20	0.093345	0.035648	2.618499
21	0.092507	0.036196	2.555699
22	0.093682	0.035517	2.637665
23	0.094761	0.036787	2.575910
24	0.095750	0.037037	2.585267
25	0.095903	0.037379	2.565703
26	0.095637	0.037577	2.545121
27	0.096622	0.037965	2.545031
28	0.098030	0.038397	2.553051
29	0.097559	0.038421	2.539242
30	0.097411	0.038802	2.510486
31	0.097496	0.039399	2.474548
32	0.096946	0.039587	2.448907
33	0.097409	0.038392	2.537239
34	0.098139	0.038546	2.545998
35	0.102356	0.050107	2.042748
36	0.104153	0.052185	1.995848
37	0.104818	0.054960	1.907183
38	0.105430	0.053407	1.974098
39	0.104041	0.040626	2.560952

	dep_lnassets_mean	dep_lnassets_std	dep_lnassets_mean/std	... \
0	13.892325	1.315163	10.563199	...
1	13.901569	1.307781	10.629889	...
2	13.948178	1.324555	10.530460	...
3	13.976943	1.321791	10.574245	...
4	14.003626	1.318442	10.621343	...
5	14.031236	1.320163	10.628415	...
6	14.053815	1.321024	10.638580	...
7	14.078143	1.323661	10.635758	...
8	14.097758	1.322348	10.661159	...
9	14.122941	1.324957	10.659173	...
10	14.140310	1.324545	10.675601	...
11	14.152956	1.326943	10.665839	...
12	14.173942	1.333582	10.628471	...
13	14.199583	1.334980	10.636555	...
14	14.223872	1.338311	10.628229	...
15	14.237983	1.334905	10.665913	...
16	14.250309	1.339278	10.640289	...
17	14.278910	1.348275	10.590501	...
18	14.294549	1.341662	10.654360	...
19	14.298175	1.337033	10.693957	...
20	14.333446	1.338248	10.710604	...
21	14.331095	1.336419	10.723507	...
22	14.301451	1.301099	10.991821	...
23	14.337858	1.343630	10.670989	...
24	14.350014	1.348791	10.639166	...
25	14.356482	1.351045	10.626203	...
26	14.368189	1.353755	10.613585	...
27	14.368182	1.354676	10.606359	...
28	14.373151	1.359391	10.573228	...
29	14.394710	1.364128	10.552315	...
30	14.391942	1.364099	10.550511	...
31	14.390480	1.369805	10.505498	...
32	14.399547	1.375941	10.465234	...
33	14.418616	1.381242	10.438881	...
34	14.419457	1.421270	10.145474	...
35	14.445605	1.429060	10.108463	...
36	14.458022	1.444429	10.009508	...

37	14.484823	1.451941	9.976181	...
38	14.532966	1.457482	9.971286	...
39	14.547235	1.426783	10.195831	...

	dep_depositratio_mean/std	dep_loans_REratio_mean	dep_loans_REratio_std	\
0	5.948086	0.704876	0.154629	
1	5.546803	0.708214	0.155144	
2	5.480818	0.712567	0.154646	
3	5.417913	0.714156	0.154508	
4	5.439159	0.716212	0.155992	
5	5.442989	0.717576	0.155023	
6	5.565569	0.722524	0.153050	
7	5.664867	0.721998	0.153623	
8	5.554901	0.723585	0.153159	
9	5.587013	0.724879	0.151578	
10	5.592820	0.726393	0.151641	
11	5.436935	0.726000	0.151172	
12	5.658311	0.727115	0.151102	
13	5.592262	0.727923	0.150226	
14	5.511178	0.730704	0.149987	
15	5.470133	0.730142	0.150869	
16	5.332041	0.732667	0.151261	
17	5.437302	0.735124	0.149984	
18	5.579339	0.740377	0.148135	
19	5.593590	0.741894	0.148943	
20	5.715534	0.749269	0.150668	
21	5.648949	0.748807	0.150820	
22	6.479091	0.750378	0.149992	
23	6.114608	0.746384	0.153481	
24	5.648080	0.745772	0.155150	
25	5.467545	0.744706	0.156421	
26	5.508641	0.746799	0.154964	
27	5.606267	0.743866	0.156622	
28	5.530871	0.743310	0.157486	
29	5.401983	0.740595	0.159406	
30	5.451352	0.741240	0.159328	
31	5.490608	0.737858	0.160367	
32	5.488669	0.736349	0.160899	
33	5.453770	0.734809	0.160717	
34	5.174326	0.735961	0.160673	
35	5.212493	0.732812	0.160838	
36	5.215755	0.731405	0.160881	
37	5.149940	0.728721	0.162689	
38	5.289406	0.727057	0.163847	
39	5.425958	0.727155	0.161820	

	dep_loans_REratio_mean/std	dep_liquidity_mean	dep_liquidity_std	\
0	4.558507	0.036987	0.029410	
1	4.564874	0.036303	0.029253	
2	4.607732	0.035539	0.028196	
3	4.622140	0.035778	0.026878	
4	4.591333	0.036823	0.027460	
5	4.628837	0.037272	0.027927	
6	4.720838	0.035497	0.027505	
7	4.699801	0.034009	0.027483	
8	4.724395	0.032727	0.026819	
9	4.782223	0.032898	0.027448	
10	4.790207	0.032810	0.027902	
11	4.802493	0.030990	0.026370	
12	4.812094	0.030063	0.024964	
13	4.845536	0.031132	0.027101	

14	4.871774	0.033084	0.029629
15	4.839572	0.033128	0.028501
16	4.843736	0.032020	0.028469
17	4.901343	0.034870	0.033821
18	4.998001	0.042638	0.042306
19	4.981047	0.047295	0.044460
20	4.972997	0.067532	0.056837
21	4.964915	0.067463	0.057686
22	5.002796	0.071155	0.059097
23	4.863049	0.073235	0.060064
24	4.806770	0.073066	0.060833
25	4.760905	0.074844	0.062529
26	4.819177	0.075874	0.062812
27	4.749430	0.075354	0.063034
28	4.719841	0.072893	0.062310
29	4.645982	0.077249	0.063977
30	4.652284	0.080213	0.065866
31	4.601069	0.073010	0.062260
32	4.576473	0.067834	0.059363
33	4.572071	0.067772	0.061189
34	4.580504	0.068694	0.060326
35	4.556209	0.066721	0.062832
36	4.546260	0.061640	0.060246
37	4.479226	0.061306	0.058692
38	4.437406	0.062867	0.060312
39	4.493597	0.061950	0.052791

	dep_liquidity_mean/std	dep_cpp_bankquarter_mean	dep_cpp_bankquarter_std	\
0	1.257647	0.000000	0.000000	
1	1.241004	0.000000	0.000000	
2	1.260418	0.000000	0.000000	
3	1.331109	0.000000	0.000000	
4	1.340957	0.000000	0.000000	
5	1.334644	0.000000	0.000000	
6	1.290546	0.000000	0.000000	
7	1.237440	0.000000	0.000000	
8	1.220294	0.000000	0.000000	
9	1.198542	0.000000	0.000000	
10	1.175928	0.000000	0.000000	
11	1.175200	0.000000	0.000000	
12	1.204240	0.000000	0.000000	
13	1.148752	0.000000	0.000000	
14	1.116619	0.000000	0.000000	
15	1.162372	0.000000	0.000000	
16	1.124750	0.000000	0.000000	
17	1.031010	0.000000	0.000000	
18	1.007845	0.257669	0.437686	
19	1.063786	0.279141	0.448921	
20	1.188173	0.254601	0.435971	
21	1.169484	0.248466	0.432455	
22	1.204047	0.095541	0.294196	
23	1.219291	0.230061	0.421195	
24	1.201077	0.202454	0.402137	
25	1.196954	0.187117	0.390304	
26	1.207964	0.180982	0.385299	
27	1.195443	0.173313	0.378808	
28	1.169858	0.154908	0.362095	
29	1.207447	0.150307	0.357646	
30	1.217825	0.142638	0.349972	
31	1.172671	0.128834	0.335274	
32	1.142687	0.112308	0.315988	

33	1.107580	0.105100	0.306920
34	1.138719	0.095975	0.294786
35	1.061895	0.093897	0.291913
36	1.023141	0.086342	0.281089
37	1.044542	0.087025	0.282095
38	1.042356	0.066667	0.249647
39	1.173507	0.070033	0.255410

	dep_cpp_bankquarter_mean/std
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN
6	NaN
7	NaN
8	NaN
9	NaN
10	NaN
11	NaN
12	NaN
13	NaN
14	NaN
15	NaN
16	NaN
17	NaN
18	0.588706
19	0.621804
20	0.583986
21	0.574548
22	0.324755
23	0.546211
24	0.503445
25	0.479412
26	0.469718
27	0.457522
28	0.427811
29	0.420267
30	0.407570
31	0.384266
32	0.355418
33	0.342436
34	0.325576
35	0.321659
36	0.307170
37	0.308496
38	0.267044
39	0.274197

[40 rows x 28 columns]
[0.6323413381283575, 2.6818312305841623, 10.542414501310363, 1.0746497011945464, 3.095305356410314, 5.533146855685416, 4.724686107635749, 1.1751940700494448, 0.4223643453567341]

In [158]:

```
fixed_deci = pd.DataFrame(dep_name,columns={'dep_name'})
fixed_deci["quarter_mean/std"] = quarter_mean_std
fixed_deci["company_mean/std"] = company_mean_std
print(fixed_deci)
```

```
                dep_name  quarter_mean/std  company_mean/std
0               dep_roa1          2.204779          0.632341
1           dep_leverage          9.879336          2.681831
2           dep_lnassets         89.318648         10.542415
3  dep_creditrisk_total3          1.940507          1.074650
4                dep_cir          5.000216          3.095305
5       dep_depositratio         16.395762          5.533147
6      dep_loans_REratio         29.454430          4.724686
7          dep_liquidity          2.657614          1.175194
8    dep_cpp_bankquarter          0.591919          0.422364
```

In [108]:

```
print(df.head())
```

```
   rssd9001    rssd9999  bhc_avgtradingratio  treat_3_b_avg  after_DFA_1  \
0   1020180  20040930.0                  0.0            0.0          0.0
1   1020180  20041231.0                  0.0            0.0          0.0
2   1020180  20050331.0                  0.0            0.0          0.0
3   1020180  20050630.0                  0.0            0.0          0.0
4   1020180  20050930.0                  0.0            0.0          0.0

   dep_roa1  dep_leverage  dep_lnassets  dep_creditrisk_total3   dep_cir  \
0  0.002772      0.081957     15.601202               0.013304  0.463811
1  0.003045      0.082480     15.630583               0.009732  0.456392
2  0.002616      0.082074     15.644925               0.011830  0.444011
3  0.002647      0.081712     15.679702               0.013654  0.433771
4  0.002867      0.082944     15.661868               0.012456  0.400985

   dep_depositratio  dep_loans_REratio  dep_liquidity  dep_cpp_bankquarter
0          0.561805           0.593738       0.024337                  0.0
1          0.557617           0.601763       0.025446                  0.0
2          0.556980           0.600700       0.025153                  0.0
3          0.571642           0.601042       0.023670                  0.0
4          0.577408           0.581438       0.029793                  0.0
```

In [174]:

```
# beforeDFA: Affect
affect = df.groupby(["rssd9001"])['bhc_avgtradingratio'].mean()
df["affect"] = df["rssd9001"].apply(lambda x: affect[x])

# before2007: Affect_pre2007
df_pre2007 = df[df["rssd9999"]<=20060931]
affect_pre2007 = df_pre2007.groupby(["rssd9001"])['bhc_avgtradingratio'].mean()
df["affect_pre2007"] = df["rssd9001"].apply(lambda x: affect_pre2007[x])
```

Model Regression

In [4]:

```
df_after_PSM = pd.read_csv("df_after_PSM.csv")
df_before_PSM = pd.read_csv("df_before_PSM.csv")
```

In [5]:

```
# df_before_PSM
# beforeDFA: Affect
affect = df_before_PSM.groupby(["rssd9001"])['bhc_avgtradingratio'].mean()
df_before_PSM["affect"] = df_before_PSM["rssd9001"].apply(lambda x: affect[x])

# before2007: Affect_pre2007
df_pre2007 = df_before_PSM[df_before_PSM["rssd9999"]<=20060931]
affect_pre2007 = df_pre2007.groupby(["rssd9001"])['bhc_avgtradingratio'].mean()
df_before_PSM["affect_pre2007"] = df_before_PSM["rssd9001"].apply(lambda x: affect_pre2007[x])


# df_after_PSM
# beforeDFA: Affect
affect = df_after_PSM.groupby(["rssd9001"])['bhc_avgtradingratio'].mean()
df_after_PSM["affect"] = df_after_PSM["rssd9001"].apply(lambda x: affect[x])

# before2007: Affect_pre2007
df_pre2007 = df_after_PSM[df_after_PSM["rssd9999"]<=20060931]
affect_pre2007 = df_pre2007.groupby(["rssd9001"])['bhc_avgtradingratio'].mean()
df_after_PSM["affect_pre2007"] = df_after_PSM["rssd9001"].apply(lambda x: affect_pre2007[x])
```

In [6]:

```
print(len(df_before_PSM))
print(len(df_after_PSM))
```

```
26080
1600
```

In [7]:

```
import statsmodels.formula.api as smf

regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1', df_before_PSM).fit()
regout.summary2()
```

Out[7]:

Model:	OLS	Adj. R-squared:	0.000
Dependent Variable:	bhc_avgtradingratio	AIC:	-145207.0567
Date:	2021-10-26 00:15	BIC:	-145190.7188
No. Observations:	26080	Log-Likelihood:	72606.
Df Model:	1	F-statistic:	3.808
Df Residuals:	26078	Prob (F-statistic):	0.0510
R-squared:	0.000	Scale:	0.00022359

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.0024	0.0001	18.3215	0.0000	0.0021	0.0027
after_DFA_1	-0.0004	0.0002	-1.9513	0.0510	-0.0007	0.0000

Omnibus:	43278.144	Durbin-Watson:	0.082
Prob(Omnibus):	0.000	Jarque-Bera (JB):	26141661.751
Skew:	11.442	Prob(JB):	0.000
Kurtosis:	156.405	Condition No.:	3

In [8]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + dep_roa1 + dep_leverage + dep_lnassets + dep_creditrisk_total3 + dep_cir + dep_depositratio + dep_loans_REratio + dep_liquidity + dep_cpp_bankquarter', df_before_PSM).fit()
regout.summary2()
```

Out[8]:

Model:	OLS	Adj. R-squared:	0.272
Dependent Variable:	bhc_avgtradingratio	AIC:	-153481.7735
Date:	2021-10-26 00:15	BIC:	-153391.9153
No. Observations:	26080	Log-Likelihood:	76752.
Df Model:	10	F-statistic:	976.4
Df Residuals:	26069	Prob (F-statistic):	0.00
R-squared:	0.272	Scale:	0.00016274

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.0413	0.0015	-28.2015	0.0000	-0.0441	-0.0384
after_DFA_1	-0.0020	0.0002	-11.0418	0.0000	-0.0023	-0.0016
dep_roa1	0.0431	0.0111	3.8889	0.0001	0.0214	0.0649
dep_leverage	-0.0347	0.0023	-15.3637	0.0000	-0.0392	-0.0303
dep_lnassets	0.0044	0.0001	63.2951	0.0000	0.0043	0.0046
dep_creditrisk_total3	0.0415	0.0028	14.6055	0.0000	0.0359	0.0470
dep_cir	0.0024	0.0002	11.9930	0.0000	0.0020	0.0028
dep_depositratio	-0.0206	0.0008	-26.8166	0.0000	-0.0221	-0.0191
dep_loans_REratio	-0.0056	0.0006	-9.8621	0.0000	-0.0068	-0.0045
dep_liquidity	0.0027	0.0017	1.5780	0.1146	-0.0007	0.0061
dep_cpp_bankquarter	-0.0011	0.0003	-3.6484	0.0003	-0.0017	-0.0005

Omnibus:	38260.668	Durbin-Watson:	0.110
Prob(Omnibus):	0.000	Jarque-Bera (JB):	16537568.194
Skew:	8.944	Prob(JB):	0.000
Kurtosis:	125.060	Condition No.:	2025

In [55]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + affect + after_DFA_1 * affect', df_before_
PSM).fit()
regout.summary2()
```

Out[55]:

Model:	OLS	Adj. R-squared:	0.910
Dependent Variable:	bhc_avgtradingratio	AIC:	-208107.1056
Date:	2021-10-25 15:29	BIC:	-208074.4299
No. Observations:	26080	Log-Likelihood:	1.0406e+05
Df Model:	3	F-statistic:	8.829e+04
Df Residuals:	26076	Prob (F-statistic):	0.00
R-squared:	0.910	Scale:	2.0044e-05

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.0000	0.0000	-0.5502	0.5822	-0.0001	0.0001
after_DFA_1	0.0000	0.0001	0.7781	0.4365	-0.0001	0.0002
affect	1.0913	0.0028	395.4581	0.0000	1.0859	1.0967
after_DFA_1:affect	-0.1826	0.0039	-46.7823	0.0000	-0.1902	-0.1749

Omnibus:	43017.736	Durbin-Watson:	0.561
Prob(Omnibus):	0.000	Jarque-Bera (JB):	99194706.806
Skew:	10.616	Prob(JB):	0.000
Kurtosis:	304.385	Condition No.:	184

In [56]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + affect + after_DFA_1 * affect + dep_roa1 + dep_leverage + dep_lnassets + dep_creditrisk_total3 + dep_cir + dep_depositratio + dep_loans_REratio + dep_liquidity + dep_cpp_bankquarter', df_before_PSM).fit()
regout.summary2()
```

Out[56]:

Model:	OLS	Adj. R-squared:	0.910
Dependent Variable:	bhc_avgtradingratio	AIC:	-208120.5854
Date:	2021-10-25 15:29	BIC:	-208014.3893
No. Observations:	26080	Log-Likelihood:	1.0407e+05
Df Model:	12	F-statistic:	2.209e+04
Df Residuals:	26067	Prob (F-statistic):	0.00
R-squared:	0.910	Scale:	2.0026e-05

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.0009	0.0005	-1.7950	0.0727	-0.0020	0.0001
after_DFA_1	-0.0000	0.0001	-0.3202	0.7488	-0.0001	0.0001
affect	1.0908	0.0030	358.4538	0.0000	1.0848	1.0967
after_DFA_1:affect	-0.1831	0.0039	-46.8715	0.0000	-0.1908	-0.1755
dep_roa1	0.0017	0.0039	0.4414	0.6589	-0.0059	0.0093
dep_leverage	0.0011	0.0008	1.3749	0.1692	-0.0005	0.0027
dep_lnassets	0.0000	0.0000	1.2801	0.2005	-0.0000	0.0001
dep_creditrisk_total3	0.0036	0.0010	3.5759	0.0003	0.0016	0.0055
dep_cir	0.0002	0.0001	2.8979	0.0038	0.0001	0.0003
dep_depositratio	0.0005	0.0003	1.8271	0.0677	-0.0000	0.0010
dep_loans_REratio	-0.0002	0.0002	-0.8443	0.3985	-0.0006	0.0002
dep_liquidity	-0.0010	0.0006	-1.6944	0.0902	-0.0022	0.0002
dep_cpp_bankquarter	-0.0001	0.0001	-1.1203	0.2626	-0.0003	0.0001

Omnibus:	43070.699	Durbin-Watson:	0.562
Prob(Omnibus):	0.000	Jarque-Bera (JB):	99808540.175
Skew:	10.643	Prob(JB):	0.000
Kurtosis:	305.316	Condition No.:	2353

In [237]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + affect + after_DFA_1 * affect', df_after_P
SM).fit()
regout.summary2()
```

Out[237]:

Model:	OLS	Adj. R-squared:	0.945
Dependent Variable:	bhc_avgtradingratio	AIC:	-9565.6802
Date:	2021-10-24 22:58	BIC:	-9544.1692
No. Observations:	1600	Log-Likelihood:	4786.8
Df Model:	3	F-statistic:	9177.
Df Residuals:	1596	Prob (F-statistic):	0.00
R-squared:	0.945	Scale:	0.00014791

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.0024	0.0005	-4.8585	0.0000	-0.0034	-0.0014
after_DFA_1	0.0048	0.0007	6.8709	0.0000	0.0034	0.0062
affect	1.0981	0.0086	128.2213	0.0000	1.0813	1.1149
after_DFA_1:affect	-0.1962	0.0121	-16.2030	0.0000	-0.2200	-0.1725

Omnibus:	497.181	Durbin-Watson:	0.363
Prob(Omnibus):	0.000	Jarque-Bera (JB):	6957.776
Skew:	1.053	Prob(JB):	0.000
Kurtosis:	12.997	Condition No.:	52

In [231]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + affect + after_DFA_1 * affect + dep_roa1 + dep_leverage + dep_lnassets + dep_creditrisk_total3 + dep_cir + dep_depositratio + dep_loans_REratio + dep_liquidity + dep_cpp_bankquarter', df_after_PSM).fit()
regout.summary2()
```

Out[231]:

Model:	OLS	Adj. R-squared:	0.947
Dependent Variable:	bhc_avgtradingratio	AIC:	-9620.9882
Date:	2021-10-24 22:42	BIC:	-9551.0773
No. Observations:	1600	Log-Likelihood:	4823.5
Df Model:	12	F-statistic:	2395.
Df Residuals:	1587	Prob (F-statistic):	0.00
R-squared:	0.948	Scale:	0.00014208

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.0065	0.0047	-1.3818	0.1672	-0.0157	0.0027
after_DFA_1	0.0025	0.0008	3.1452	0.0017	0.0009	0.0040
affect	1.1046	0.0099	111.3525	0.0000	1.0852	1.1241
after_DFA_1:affect	-0.2078	0.0120	-17.2592	0.0000	-0.2314	-0.1842
dep_roa1	-0.0078	0.0879	-0.0889	0.9291	-0.1802	0.1645
dep_leverage	0.0173	0.0113	1.5245	0.1276	-0.0050	0.0396
dep_lnassets	-0.0000	0.0002	-0.1398	0.8889	-0.0004	0.0004
dep_creditrisk_total3	0.0512	0.0114	4.5107	0.0000	0.0290	0.0735
dep_cir	0.0015	0.0005	2.9774	0.0030	0.0005	0.0025
dep_depositratio	0.0050	0.0026	1.9555	0.0507	-0.0000	0.0101
dep_loans_REratio	-0.0039	0.0023	-1.7025	0.0889	-0.0084	0.0006
dep_liquidity	0.0005	0.0077	0.0701	0.9441	-0.0145	0.0155
dep_cpp_bankquarter	0.0048	0.0009	5.0224	0.0000	0.0029	0.0066

Omnibus:	533.121	Durbin-Watson:	0.391
Prob(Omnibus):	0.000	Jarque-Bera (JB):	9004.187
Skew:	1.101	Prob(JB):	0.000
Kurtosis:	14.411	Condition No.:	5439

In [238]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + affect_pre2007 + after_DFA_1 * affect_pre2
007', df_before_PSM).fit()
regout.summary2()
```

Out[238]:

Model:	OLS	Adj. R-squared:	0.872
Dependent Variable:	bhc_avgtradingratio	AIC:	-198860.0393
Date:	2021-10-24 23:00	BIC:	-198827.3636
No. Observations:	26080	Log-Likelihood:	99434.
Df Model:	3	F-statistic:	5.934e+04
Df Residuals:	26076	Prob (F-statistic):	0.00
R-squared:	0.872	Scale:	2.8573e-05

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.0005	0.0000	9.6622	0.0000	0.0004	0.0005
after_DFA_1	0.0000	0.0001	0.5373	0.5911	-0.0001	0.0002
affect_pre2007	1.0505	0.0032	330.1421	0.0000	1.0443	1.0568
after_DFA_1:affect_pre2007	-0.2147	0.0045	-47.7169	0.0000	-0.2236	-0.2059

Omnibus:	39705.450	Durbin-Watson:	0.398
Prob(Omnibus):	0.000	Jarque-Bera (JB):	60430825.810
Skew:	9.065	Prob(JB):	0.000
Kurtosis:	238.122	Condition No.:	178

In [239]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + affect_pre2007 + after_DFA_1 * affect_pre2
007', df_after_PSM).fit()
regout.summary2()
```

Out[239]:

Model:	OLS	Adj. R-squared:	0.914
Dependent Variable:	bhc_avgtradingratio	AIC:	-8853.5202
Date:	2021-10-24 23:00	BIC:	-8832.0092
No. Observations:	1600	Log-Likelihood:	4430.8
Df Model:	3	F-statistic:	5689.
Df Residuals:	1596	Prob (F-statistic):	0.00
R-squared:	0.914	Scale:	0.00023083

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.0011	0.0006	1.8597	0.0631	-0.0001	0.0023
after_DFA_1	0.0055	0.0008	6.4931	0.0000	0.0038	0.0072
affect_pre2007	1.0430	0.0101	103.3581	0.0000	1.0233	1.0628
after_DFA_1:affect_pre2007	-0.2367	0.0143	-16.5873	0.0000	-0.2647	-0.2087

Omnibus:	234.946	Durbin-Watson:	0.260
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2973.881
Skew:	0.187	Prob(JB):	0.000
Kurtosis:	9.668	Condition No.:	49

In [234]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + affect_pre2007 + after_DFA_1 * affect_pre2007 + dep_roa1 + dep_leverage + dep_lnassets + dep_creditrisk_total3 + dep_cir + dep_depositratio + dep_loans_REratio + dep_liquidity + dep_cpp_bankquarter', df_before_PSM).fit()
regout.summary2()
```

Out[234]:

Model:	OLS	Adj. R-squared:	0.876
Dependent Variable:	bhc_avgtradingratio	AIC:	-199719.6862
Date:	2021-10-24 22:42	BIC:	-199613.4901
No. Observations:	26080	Log-Likelihood:	99873.
Df Model:	12	F-statistic:	1.541e+04
Df Residuals:	26067	Prob (F-statistic):	0.00
R-squared:	0.876	Scale:	2.7637e-05

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.0084	0.0006	-13.7798	0.0000	-0.0096	-0.0072
after_DFA_1	-0.0003	0.0001	-3.9526	0.0001	-0.0004	-0.0001
affect_pre2007	1.0148	0.0034	299.3911	0.0000	1.0081	1.0214
after_DFA_1:affect_pre2007	-0.2147	0.0044	-48.4642	0.0000	-0.2234	-0.2060
dep_roa1	0.0038	0.0046	0.8416	0.4000	-0.0051	0.0128
dep_leverage	-0.0010	0.0009	-1.0173	0.3090	-0.0028	0.0009
dep_lnassets	0.0007	0.0000	22.6217	0.0000	0.0006	0.0008
dep_creditrisk_total3	0.0109	0.0012	9.2734	0.0000	0.0086	0.0132
dep_cir	0.0006	0.0001	7.5808	0.0000	0.0005	0.0008
dep_depositratio	-0.0019	0.0003	-5.9613	0.0000	-0.0025	-0.0013
dep_loans_REratio	0.0001	0.0002	0.3014	0.7631	-0.0004	0.0005
dep_liquidity	-0.0017	0.0007	-2.3803	0.0173	-0.0031	-0.0003
dep_cpp_bankquarter	-0.0004	0.0001	-3.6260	0.0003	-0.0007	-0.0002

Omnibus:	40112.492	Durbin-Watson:	0.414
Prob(Omnibus):	0.000	Jarque-Bera (JB):	64878656.161
Skew:	9.242	Prob(JB):	0.000
Kurtosis:	246.645	Condition No.:	2265

In [235]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + affect_pre2007 + after_DFA_1 * affect_pre2007 + dep_roa1 + dep_leverage + dep_lnassets + dep_creditrisk_total3 + dep_cir + dep_depositratio + dep_loans_REratio + dep_liquidity + dep_cpp_bankquarter', df_after_PSM).fit()
regout.summary2()
```

Out[235]:

Model:	OLS	Adj. R-squared:	0.931
Dependent Variable:	bhc_avgtradingratio	AIC:	-9179.5455
Date:	2021-10-24 22:42	BIC:	-9109.6346
No. Observations:	1600	Log-Likelihood:	4602.8
Df Model:	12	F-statistic:	1785.
Df Residuals:	1587	Prob (F-statistic):	0.00
R-squared:	0.931	Scale:	0.00018723

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.0522	0.0054	-9.6825	0.0000	-0.0628	-0.0416
after_DFA_1	0.0014	0.0009	1.6321	0.1028	-0.0003	0.0032
affect_pre2007	1.0009	0.0104	96.3608	0.0000	0.9805	1.0213
after_DFA_1:affect_pre2007	-0.2472	0.0130	-19.0382	0.0000	-0.2727	-0.2218
dep_roa1	0.0750	0.1009	0.7437	0.4572	-0.1229	0.2730
dep_leverage	0.0313	0.0130	2.3983	0.0166	0.0057	0.0569
dep_lnassets	0.0026	0.0002	11.1654	0.0000	0.0021	0.0030
dep_creditrisk_total3	0.0967	0.0130	7.4247	0.0000	0.0711	0.1222
dep_cir	0.0035	0.0006	6.1106	0.0000	0.0024	0.0047
dep_depositratio	-0.0011	0.0030	-0.3864	0.6992	-0.0069	0.0047
dep_loans_REratio	0.0007	0.0026	0.2823	0.7777	-0.0044	0.0059
dep_liquidity	0.0108	0.0088	1.2267	0.2201	-0.0064	0.0280
dep_cpp_bankquarter	0.0021	0.0011	1.8918	0.0587	-0.0001	0.0042

Omnibus:	306.312	Durbin-Watson:	0.345
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3445.484
Skew:	0.547	Prob(JB):	0.000
Kurtosis:	10.105	Condition No.:	5441

In [240]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + treat_3_b_avg + after_DFA_1 * treat_3_b_av
g', df_before_PSM).fit()
regout.summary2()
```

Out[240]:

Model:	OLS	Adj. R-squared:	0.533
Dependent Variable:	bhc_avgtradingratio	AIC:	-165046.0333
Date:	2021-10-24 23:01	BIC:	-165013.3576
No. Observations:	26080	Log-Likelihood:	82527.
Df Model:	3	F-statistic:	9912.
Df Residuals:	26076	Prob (F-statistic):	0.00
R-squared:	0.533	Scale:	0.00010448

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.0009	0.0001	10.0998	0.0000	0.0007	0.0011
after_DFA_1	-0.0001	0.0001	-0.7567	0.4492	-0.0003	0.0002
treat_3_b_avg	0.0970	0.0007	133.1949	0.0000	0.0956	0.0984
after_DFA_1:treat_3_b_avg	-0.0173	0.0010	-16.7613	0.0000	-0.0193	-0.0152

Omnibus:	31194.623	Durbin-Watson:	0.136
Prob(Omnibus):	0.000	Jarque-Bera (JB):	13934289.866
Skew:	5.903	Prob(JB):	0.000
Kurtosis:	115.621	Condition No.:	21

In [241]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + treat_3_b_avg + after_DFA_1 * treat_3_b_avg', df_after_PSM).fit()
regout.summary2()
```

Out[241]:

Model:	OLS	Adj. R-squared:	0.464
Dependent Variable:	bhc_avgtradingratio	AIC:	-5921.1566
Date:	2021-10-24 23:02	BIC:	-5899.6456
No. Observations:	1600	Log-Likelihood:	2964.6
Df Model:	3	F-statistic:	463.3
Df Residuals:	1596	Prob (F-statistic):	1.81e-216
R-squared:	0.465	Scale:	0.0014429

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.0059	0.0016	3.7889	0.0002	0.0028	0.0089
after_DFA_1	0.0047	0.0022	2.1599	0.0309	0.0004	0.0090
treat_3_b_avg	0.0921	0.0031	29.6802	0.0000	0.0860	0.0981
after_DFA_1:treat_3_b_avg	-0.0221	0.0044	-5.0384	0.0000	-0.0307	-0.0135

Omnibus:	531.893	Durbin-Watson:	0.058
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2701.770
Skew:	1.479	Prob(JB):	0.000
Kurtosis:	8.637	Condition No.:	7

In [232]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + treat_3_b_avg + after_DFA_1 * treat_3_b_avg + dep_roa1 + dep_leverage + dep_lnassets + dep_creditrisk_total3 + dep_cir + dep_depositratio + dep_loans_REratio + dep_liquidity + dep_cpp_bankquarter', df_before_PSM).fit()
regout.summary2()
```

Out[232]:

Model:	OLS	Adj. R-squared:	0.588
Dependent Variable:	bhc_avgtradingratio	AIC:	-168322.6942
Date:	2021-10-24 22:42	BIC:	-168216.4982
No. Observations:	26080	Log-Likelihood:	84174.
Df Model:	12	F-statistic:	3103.
Df Residuals:	26067	Prob (F-statistic):	0.00
R-squared:	0.588	Scale:	9.2116e-05

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.0290	0.0011	-26.2233	0.0000	-0.0311	-0.0268
after_DFA_1	-0.0011	0.0001	-7.8839	0.0000	-0.0013	-0.0008
treat_3_b_avg	0.0844	0.0007	116.3033	0.0000	0.0829	0.0858
after_DFA_1:treat_3_b_avg	-0.0168	0.0010	-17.3601	0.0000	-0.0187	-0.0149
dep_roa1	0.0202	0.0083	2.4227	0.0154	0.0039	0.0366
dep_leverage	-0.0184	0.0017	-10.7729	0.0000	-0.0217	-0.0150
dep_lnassets	0.0025	0.0001	45.0705	0.0000	0.0023	0.0026
dep_creditrisk_total3	0.0236	0.0021	11.0242	0.0000	0.0194	0.0278
dep_cir	0.0012	0.0002	7.9432	0.0000	0.0009	0.0015
dep_depositratio	-0.0091	0.0006	-15.6338	0.0000	-0.0103	-0.0080
dep_loans_REratio	0.0030	0.0004	6.8114	0.0000	0.0021	0.0038
dep_liquidity	-0.0018	0.0013	-1.4028	0.1607	-0.0044	0.0007
dep_cpp_bankquarter	0.0000	0.0002	0.0222	0.9823	-0.0004	0.0004

Omnibus:	31476.423	Durbin-Watson:	0.155
Prob(Omnibus):	0.000	Jarque-Bera (JB):	14617483.707
Skew:	5.993	Prob(JB):	0.000
Kurtosis:	118.360	Condition No.:	2025

In [233]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + treat_3_b_avg + after_DFA_1 * treat_3_b_av
g + dep_roa1 + dep_leverage + dep_lnassets + dep_creditrisk_total3 + dep_cir + dep_depositratio
 + dep_loans_REratio + dep_liquidity + dep_cpp_bankquarter', df_after_PSM).fit()
regout.summary2()
```

Out[233]:

Model:	OLS	Adj. R-squared:	0.677
Dependent Variable:	bhc_avgtradingratio	AIC:	-6722.5205
Date:	2021-10-24 22:42	BIC:	-6652.6096
No. Observations:	1600	Log-Likelihood:	3374.3
Df Model:	12	F-statistic:	280.6
Df Residuals:	1587	Prob (F-statistic):	0.00
R-squared:	0.680	Scale:	0.00086954

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.1548	0.0124	-12.5103	0.0000	-0.1790	-0.1305
after_DFA_1	0.0009	0.0020	0.4604	0.6453	-0.0030	0.0048
treat_3_b_avg	0.0881	0.0027	32.2561	0.0000	0.0827	0.0934
after_DFA_1:treat_3_b_avg	-0.0192	0.0035	-5.5100	0.0000	-0.0261	-0.0124
dep_roa1	-0.2431	0.2174	-1.1183	0.2636	-0.6695	0.1833
dep_leverage	-0.1625	0.0276	-5.8761	0.0000	-0.2167	-0.1082
dep_lnassets	0.0095	0.0005	18.9898	0.0000	0.0085	0.0105
dep_creditrisk_total3	0.1028	0.0281	3.6640	0.0003	0.0478	0.1578
dep_cir	0.0028	0.0013	2.2267	0.0261	0.0003	0.0052
dep_depositratio	-0.0128	0.0065	-1.9659	0.0495	-0.0255	-0.0000
dep_loans_REratio	0.0195	0.0059	3.3254	0.0009	0.0080	0.0310
dep_liquidity	-0.0265	0.0192	-1.3806	0.1676	-0.0643	0.0112
dep_cpp_bankquarter	0.0068	0.0024	2.8566	0.0043	0.0021	0.0114

Omnibus:	533.965	Durbin-Watson:	0.095
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3443.690
Skew:	1.403	Prob(JB):	0.000
Kurtosis:	9.617	Condition No.:	5440

In [11]:

```
# Model Regression for different company groups
cmp11 = pd.read_csv("cmp11.csv")
cmp22 = pd.read_csv("cmp22.csv")
print(len(cmp11),len(cmp22))

# beforeDFA: Affect
affect = cmp11.groupby(["rssd9001"])['bhc_avgtradingratio'].mean()
cmp11["affect"] = cmp11["rssd9001"].apply(lambda x: affect[x])

affect = cmp22.groupby(["rssd9001"])['bhc_avgtradingratio'].mean()
cmp22["affect"] = cmp22["rssd9001"].apply(lambda x: affect[x])
```

960 640

In [14]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + treat_3_b_avg + after_DFA_1 * treat_3_b_avg + dep_roa1 + dep_leverage + dep_lnassets + dep_creditrisk_total3 + dep_cir + dep_depositratio + dep_loans_REratio + dep_liquidity + dep_cpp_bankquarter', cmp11).fit()
regout.summary2()
```

Out[14]:

Model:	OLS	Adj. R-squared:	0.695
Dependent Variable:	bhc_avgtradingratio	AIC:	-6024.0148
Date:	2021-10-26 00:22	BIC:	-5960.7447
No. Observations:	960	Log-Likelihood:	3025.0
Df Model:	12	F-statistic:	182.9
Df Residuals:	947	Prob (F-statistic):	7.76e-237
R-squared:	0.699	Scale:	0.00010876

	Coef.	**Std.Err.**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-0.0553	0.0068	-8.1374	0.0000	-0.0686	-0.0419
after_DFA_1	-0.0031	0.0010	-3.1396	0.0017	-0.0051	-0.0012
treat_3_b_avg	0.0425	0.0013	33.2813	0.0000	0.0400	0.0450
after_DFA_1:treat_3_b_avg	-0.0137	0.0016	-8.5564	0.0000	-0.0169	-0.0106
dep_roa1	0.9724	0.1520	6.3951	0.0000	0.6740	1.2708
dep_leverage	-0.0520	0.0117	-4.4488	0.0000	-0.0750	-0.0291
dep_lnassets	0.0025	0.0003	9.1171	0.0000	0.0020	0.0031
dep_creditrisk_total3	0.1836	0.0245	7.4950	0.0000	0.1355	0.2316
dep_cir	0.0158	0.0024	6.6462	0.0000	0.0111	0.0205
dep_depositratio	-0.0037	0.0031	-1.1690	0.2427	-0.0098	0.0025
dep_loans_REratio	0.0147	0.0023	6.4518	0.0000	0.0102	0.0192
dep_liquidity	0.0109	0.0075	1.4560	0.1457	-0.0038	0.0257
dep_cpp_bankquarter	0.0013	0.0011	1.2384	0.2159	-0.0008	0.0034

Omnibus:	400.647	Durbin-Watson:	0.553
Prob(Omnibus):	0.000	Jarque-Bera (JB):	8353.234
Skew:	1.391	Prob(JB):	0.000
Kurtosis:	17.181	Condition No.:	7870

In [15]:

```
regout = smf.ols('bhc_avgtradingratio ~ after_DFA_1 + treat_3_b_avg + after_DFA_1 * treat_3_b_av
g + dep_roa1 + dep_leverage + dep_lnassets + dep_creditrisk_total3 + dep_cir + dep_depositratio
 + dep_loans_REratio + dep_liquidity + dep_cpp_bankquarter', cmp22).fit()
regout.summary2()
```

Out[15]:

Model:	OLS	Adj. R-squared:	0.918
Dependent Variable:	bhc_avgtradingratio	AIC:	-3119.1051
Date:	2021-10-26 00:22	BIC:	-3061.1060
No. Observations:	640	Log-Likelihood:	1572.6
Df Model:	12	F-statistic:	596.7
Df Residuals:	627	Prob (F-statistic):	0.00
R-squared:	0.919	Scale:	0.00043875

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.3736	0.0369	-10.1141	0.0000	-0.4461	-0.3011
after_DFA_1	-0.0059	0.0026	-2.2658	0.0238	-0.0110	-0.0008
treat_3_b_avg	0.1242	0.0049	25.1887	0.0000	0.1145	0.1339
after_DFA_1:treat_3_b_avg	-0.0293	0.0043	-6.7824	0.0000	-0.0378	-0.0208
dep_roa1	0.4346	0.4258	1.0207	0.3078	-0.4015	1.2707
dep_leverage	-0.1160	0.0672	-1.7248	0.0851	-0.2480	0.0161
dep_lnassets	0.0231	0.0018	12.8123	0.0000	0.0196	0.0266
dep_creditrisk_total3	0.1145	0.0575	1.9897	0.0471	0.0015	0.2274
dep_cir	0.0018	0.0010	1.8967	0.0583	-0.0001	0.0038
dep_depositratio	-0.0581	0.0214	-2.7122	0.0069	-0.1002	-0.0160
dep_loans_REratio	-0.0854	0.0115	-7.3916	0.0000	-0.1081	-0.0627
dep_liquidity	0.0670	0.0402	1.6667	0.0961	-0.0119	0.1459
dep_cpp_bankquarter	0.0040	0.0035	1.1656	0.2442	-0.0028	0.0108

Omnibus:	225.656	Durbin-Watson:	0.170
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1341.438
Skew:	1.445	Prob(JB):	0.000
Kurtosis:	9.477	Condition No.:	10335

In [ ]: