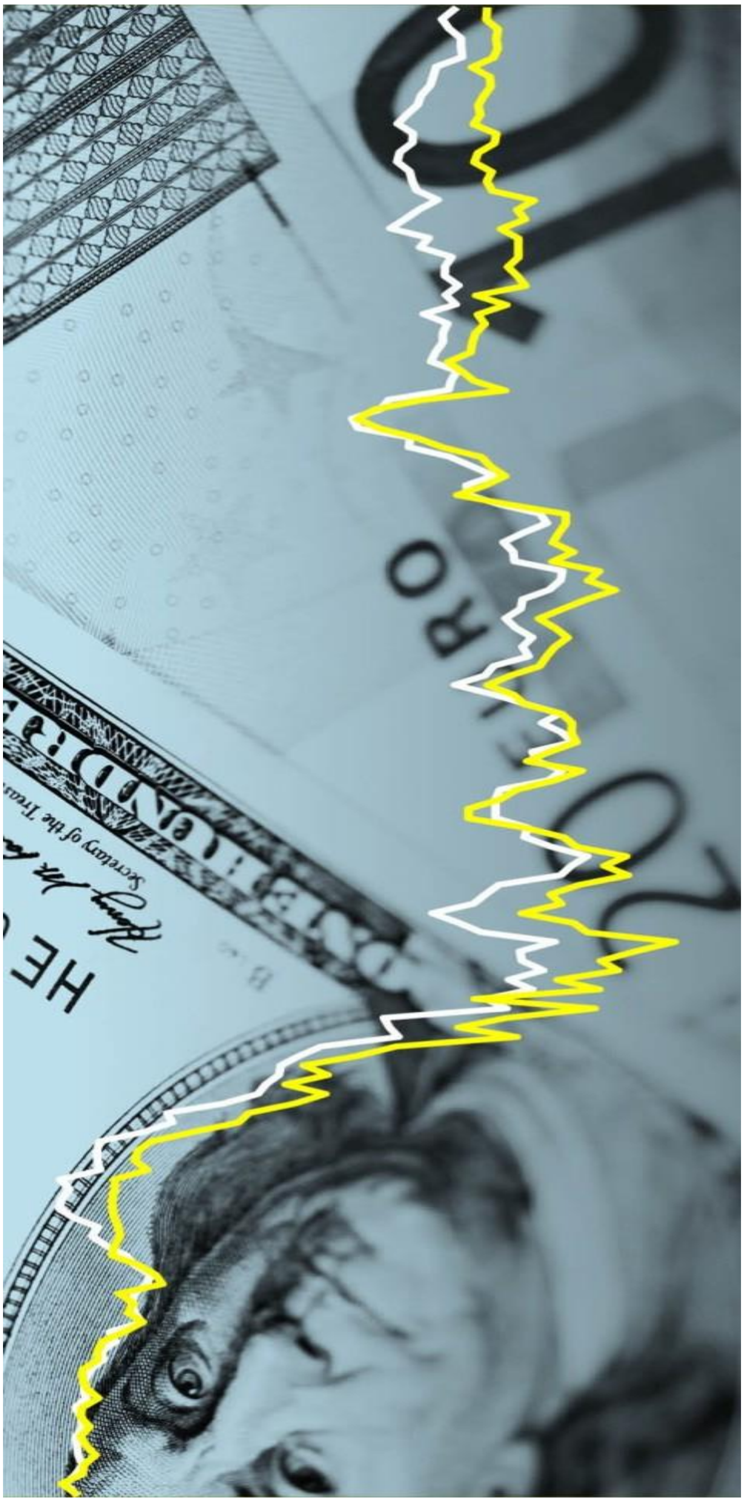


Analytics in Managerial Economics

Group Project 2

Estimating the effect of a banking regulation

TEAM MEMBERS

Chee Seng You Paul/ A0232013A

Kuan Ju Lin / A0231904M

Mukeshwaran Baskaran / A0165086Y

Yao Chuhan / A0231918A

Zhang Shichao / A0232016X

1. Introduction

In this project, we are trying to estimate the announcement effect of Volcker Rule (new banking regulation in US) on US banks. More specifically, answer the following questions:

1) Did the banks decrease their trading assets after the announcement of the new regulation?
2) If they responded to the regulation, which banks responded most and which banks least? Why?
3) How should banks or regulators use these results?

2. Dataset, variable definitions, and descriptive statistics

From the provided 'DiD-data.csv' dataset, we further calculate the *Affect* (average trading asset ratio from the third quarter of 2004 to the second quarter of 2009) and *Affect pre2007* (Average trading asset ratio from the second quarter of 2003 to the fourth quarter of 2006) to form the variable set showed in Table 1.

Dependent variables To analyze how the Volcker Rule affected the non-banking business, we first use *Trading asset ratio* as the dependent variable of both baseline and robustness tests. The *Trading asset ratio* is defined as the ratio of the trading account to total assets.

Explanatory variables and controls The main explanatory variables include *After DFA, Affect, Affect pre2007* and *Affect BHC*. The control variables include *Total assets, Leverage ratio, Profitability, Liquidity ratio, Deposit ratio, Cost income ratio, Non-performing loan ratio, Real estate loan ratio* and *CPP recipient indicator*.

[Table 1]

3. Baseline model and identification

To test the effect of the Volcker Rule, we start from a simple baseline model showed below.

$$TAR_{i,t} = \alpha + \beta_1 * After\ DFA_t + \beta_2 * Affect_i + \beta_3 * After\ DFA_t * Affect_i + \gamma_i + \delta_t + X_{i,t} + \varepsilon_{i,t} \tag{1}$$

With data being available on a BHC-quarter level, i indicates a BHC and t indicates a quarter. In the baseline model, $TAR_{i,t}$ means the trading asset ratio of bank i in quarter t. The core explanatory variables are $After\,DFA_t$ and $Affect_i$ that captures the varying degree of exposure to activities limited or banned by the Volcker Rule. γ_i and δ_t indicate the BHCs and time fixed effects used to control for influences constant either over time or across BHCs. The model is complemented by the set of control variables ($X_{i,t}$) to test for additional covariates that might vary over both time and bank and that might influence banks' business models. We test the model both including and excluding these control variables to test for potential endogeneity. We cluster the standard errors at the BHC level to account for possible autocorrelation.

4. Results and Robustness

(Question 1) Did the banks decrease their trading assets after the announcement of the new regulation?

The hypothesis of this part is that banks started to reduce their trading asset ratios after the announcement of the Volcker Rule. If this happened, we would have a significant and negative coefficient of the time indicator (*After DFA*) in the baseline model.

We test the hypothesis using a simple model that includes only time indicator and adding our vector of control variables. The results are reported in Panel A (columns (1) and (2)) of Table 2.

For this model, we can find a significant but small coefficient of *After DFA* when controlling other effects, indicating that banks started to decrease their trading assets after the announcement but not a strong shift.

The possible reason of this is that most of the bank holding companies (BHCs) had low or zero trading asset ratios when the Volcker Rule was introduced.

[Table 2]

(Question 2) If they responded to the regulation, which banks responded most and which banks least? Why?

BHCs that were particularly affected, i.e., had high trading asset ratios before, responded most to the regulation, also BHCs that had low trading asset ratios before responded least. Of these, the top 10 BHCs reduced the highest amount of trading asset percentage when compared with the rest of the BHCs, evident by the narrowing gap in trading asset ratio after the enactment of the Volcker rule (Figure 1).

We test this by running to the model including the interaction between *Affect* and *After DFA*, which is reported in column (3) and complemented by bank and quarter fixed effect in column (4) in Panel A of Table 2.

The level effects are not very surprising. First, consistent with Question 1, there is a slight but not significant decrease in the trading asset ratio after the Volcker Rule is passed. Second, the coefficient of *Affect* is significantly positive indicating that banks that had a relatively high trading asset ratio before the regulation tend to have a relatively high trading asset ratio thereafter. Finally, for the interaction term, the coefficient is negative and significant, which means that those BHCs that were particularly affected responded most to the regulation, also BHCs that had low trading asset ratios before responded least. This effect holds even when controlling for other potential explanations and for fixed effects.

This may relate to the specific content of Volcker Rule. The Volcker Rule explicitly prohibits two types of non-banking activities: (a) proprietary trading and (b) indirect trading by investing in hedge funds and private equity funds, subject to a list of permitted exceptions including de minimis investments like less than 3% of the total

ownership of a fund.
The penalty announcement of Volcker Rule can force BHCs that had 3% or more trading asset ratio to quickly reduce the activity.

For robustness tests of the above results, we try models in varying specifications. The results are reported in Panel B of Table 2. The outcomes of all tests show a significantly negative coefficient on the interaction term, which indicates that our results are robust.

[Table 2]

Besides using the propensity score matching approach to test the robustness of our DID result, we also applied Synthetic Differences in Differences (SDID) to get a better estimate. Because using SDID requires data to be a balanced panel in which banks must be observed on every quarter, we need to do further data cleaning. We find in total 170 banks that satisfy the requirement. The result as shown in Table 3 again confirms our previous conclusion by telling us the treated group has a significant negative coefficient compared to the synthetic control group. Among all the banks selected to merge the synthetic controls group, 1069778, 1020902 and 1053496 shows highest weights which are 0.348, 0.257 and 0.086. From Figure 2 we can also see the obvious decrease of trading asset ratio that treated group shows after Volcker Rule compared to the synthetic control group.

(Question 3) How should banks or regulators use these results?

As mentioned above, the result of banks decreasing trading assets is robust. If the aim of the Volcker rule is to decrease trading assets, then the rule would be effective in achieving its aim. This is especially true when noting that on the whole the big 10 BHCs are the ones who have enacted the highest trading asset percent reduction, since they pose the highest systemic risk to the banking system (Figure 1).

However, we need to note what the real aim of the Volcker Rule is, which is to decrease the riskiness of the banks to prevent meltdown in the financial system. We believe that this rule may not decrease the riskiness of the banks (or worse still increase it), and that given the resources poured into this, there may be better ways to reduce systemic risk in the financial system.

Firstly, we note that decreasing trading assets may not necessarily decrease bank riskiness. Harada et al shows that the HMV distance to default model is significant in predicting bank riskiness. Due to the limited data, we have decided to use a simplified version of the HMV model to approximate distance-to-default. This is done by using the sum of the capital asset ratio plus return on asset divided by the various standard deviations of return on asset. This is presented in Figure 3.

Figure 3 shows that in fact there can be evidence that bank riskiness has increased overall after the announcement of the Volcker rule. This can be attributed partly to the fact that risk taking incentives have not decreased while stock market expectations remain constant. In fact, the reverse is probably true.

Banks make money is by taking risk, whether in terms of loans or other indirect instruments available to them that makes use of their available capital. When limiting the risk banks can take while holding expectations constant, the banks will need to seek additional undiversified returns from other places. Thus, a greater amount of risk may be needed to achieve the same returns.

With the ultimate aim of preventing another financial crisis in mind, we return to examine what caused the previous financial crises. Table 4 shows a sample of these crises and most of them have to do with a revaluation of national currencies, credit rating agencies not rating risk properly, and most importantly, leverage. It may be that ultimately, it may be more effective to have regulations target these instead of having banks reduce riskiness by reducing trading assets.

References

Amadou N. R. Sy, and Chan-Lau J. A. (2006) "Distance-to-Default in Banking: A Bridge Too Far?" *Journal of Banking Regulation 06 (215)*

Arkhangelsky, D., Athey, S., Hirshberg D. A., Imbens, G. W. and Wager, S. (2019) "Synthetic Difference in Differences" *NBER Working Paper 25532 DOI 10.3386/w25532*

Elsas, R., Hackethal A., and Holzhauser M. (2009) "The anatomy of bank diversification." *Journal of Banking and Finance 34 (6): 1274–1287*

Harada, K., Ito, and T., Takahashi, S. (2012) "Is the Distance to Default a good measure in predicting bank failures? A case study of Japanese major banks." *Japan and the World Economy Volume 27, August 2013, Pages 70-82*

Keppo J., Korte J., "Evidence from the Announcement of the Volcker Rule" *Management Science,* 64 (2018), 215-234

Merton, R. C. (1973). "An intertemporal capital asset pricing model." *Econometrica: Journal of the Econometric Society, 41, 867-887*

Table 1: **Summary statistics**

This table reports variable names, units, means, standard deviations, minimum and maximum values, and the number of observations for the main variables of the dataset. The dataset covers the time period from Q3 2004 to Q2 2015.

Variable	**Unit**	**Mean**	**SD**	**Min**	**Max**	**N**
Dependent variables						
Trading asset ratio	Percent	0.27	2.00	0.00	42.97	41,442
Explanatory variables and controls						
After DFA	Dummy	0.45	0.50	0.00	1.00	81,560
Affect	Percent	0.19	1.70	0.00	42.94	81,560
Affect pre2007	Percent	0.14	1.30	0.00	38.95	79,701
Affect BHC	Dummy	0.01	0.11	0.00	1.00	81,560
Total assets	Ln(USD mn)	13.52	1.35	5.89	21.67	61,771
Leverage ratio	Percent	9.35	4.25	-76.23	115.80	57,017
Profitability	Percent	0.18	0.61	-38.71	93.43	56,938
Liquidity ratio	Percent	5.36	5.18	0.02	84.35	55,159
Deposit ratio	Percent	68.19	11.20	0.00	99.81	79,172
Cost income ratio	Percent	53.15	35.60	-1,247.83	4,593.33	42,382
Non-performing loan ratio	Percent	2.78	3.40	0.00	73.42	44,432
Real estate loan ratio	Percent	73.59	16.05	0.00	101.01	44,432
CPP recipient indicator	Dummy	0.04	0.20	0.00	1.00	81,560

Table 2: **Changes in the trading book – Initial compliance with the Volcker Rule?**

Panel A reports multivariate estimates of the enactment effect of the Volcker Rule (part of the Dodd-Frank Act) on bank holding companies' trading asset ratios. Panel B reports the robustness tests. *After DFA* is one for the quarters Q3 2010 – Q2 2015 and zero for the quarters Q3 2004 – Q2 2009. *Affect* is the average trading asset ratio during the pre-DFA period (Q3 2004 – Q2 2009). *Affected BHC* 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. *Affect pre2007* is the average trading asset ratio in the 15 quarters previous to 2007 (Q2 2003 – Q4 2006). Control variables comprise total assets, profitability, leverage ratio, liquidity ratio, deposit ratio, NPL ratio, RE loan ratio, cost-income ratio, and an indicator variable that takes the value of one if the bank was a recipient of the TARP CPP program in a respective quarter (and zero otherwise). Quarter and BHC fixed effects are included in the models as indicated. Standard errors are clustered at the BHC level and reported in parentheses; significance levels are indicated by *** $p < 0.01$, ** $p < 0.05$, * $p < 0.1$.

Panel A: Baseline tests

	(1)	(2)	(3)	(4)
Dependent variable	Trading asset ratio			
After DFA	0.0005**	-0.0010***	-0.00002	
	(2.53)	(-4.87)	(-0.27)	
Affect			0.9925***	
			(402.17)	
After DFA x Affect			-0.1611***	-0.2024***
			(-52.82)	(-4.30)
Controls	NO	YES	YES	YES
FE	NO	NO	NO	YES
Observations	41,442	40,026	40,026	40,026
R-squared	0.000	0.234	0.902	0.925
F	6.419	1222	30608	4.129

Panel B: Robustness tests

	(1)	(2)	(3)	(4)
Robustness test	Treatment dummy	Propensity score matching	Pre-2007 affectedness	Excluding non-trading BHCs
Dependent variable	Trading asset ratio			
After DFA x Affected BHC	-0.0234***	-0.0275*		
	(-2.66)	(-1.97)		
After DFA x Affect pre2007			-0.2052***	-0.1858***
			(-3.55)	(-3.09)
Controls & FE	YES	YES	YES	YES
Observations	40,026	518	38,783	4,493
R-squared	0.923	0.975	0.894	0.911
F	2.042	.	4.614	3.227

Table 3: **Synthetic Differences in Differences**

Robustness tests	
	(5)
Robustness test	Synthetic DID
Dependent variable	
After DFA x Affected BHC	-0.0080***
	(-2.66)
After DFA x Affect pre2007	
Controls & FE	YES
Observations	6,460
Squared-error	0.0147

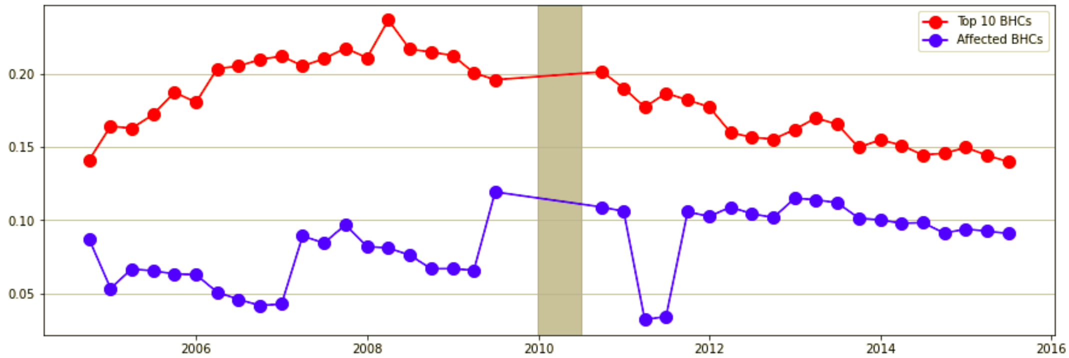
Table 4: **Sample of past crises**

Year	**Crisis**	**Cause**
1637	Tulip Mania	Coincided with outbreak of the bubonic plague
1772	Credit Crisis of 1772	Chain of Bills; credit chaining
1929	Stock Crash	Market Exuberance and Oversupply of Commodities; Leveraging
1973	OPEC Oil Crisis	Oil embargo
1994	Tequila Crisis	Devaluation of the Mexican Peso
1997	Asian Financial Crisis	Loss of the Thai currency peg amidst export dependent strategies; leverage
2007	Global Financial Crisis	Risky subprime assets and overleveraging of Lehman

Source: https://www.investopedia.com/terms/f/financial-crisis.asp

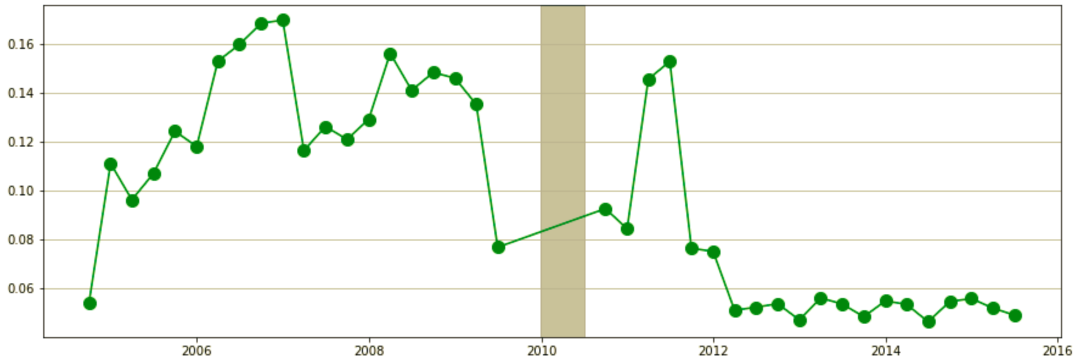
Figure 1:

Top 10 BHCs and other Affected BHCs Trading Asset Ratio over time



The grey bar represents the period where the Volcker rule was announced

Trading Asset Ratio gap between top 10 BHCs vs other Affected BHCs



Sample Count of other BHCs, explaining why there is a kink in 2011

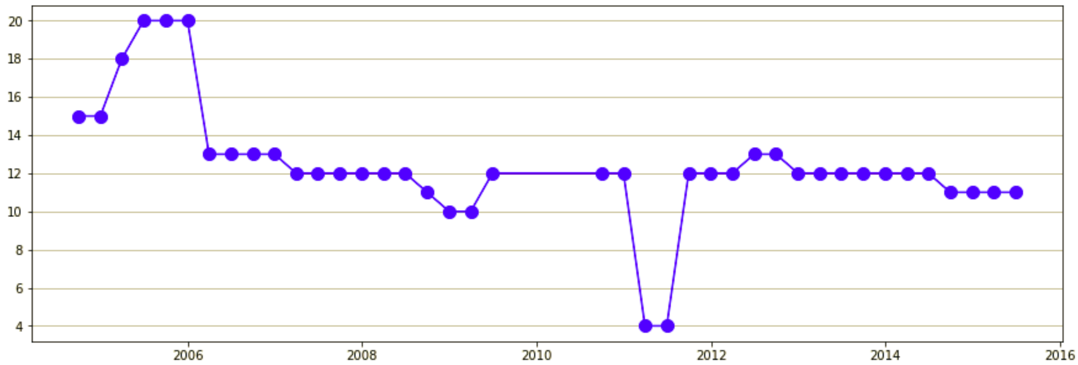


Figure 2: **Synthetic DID**

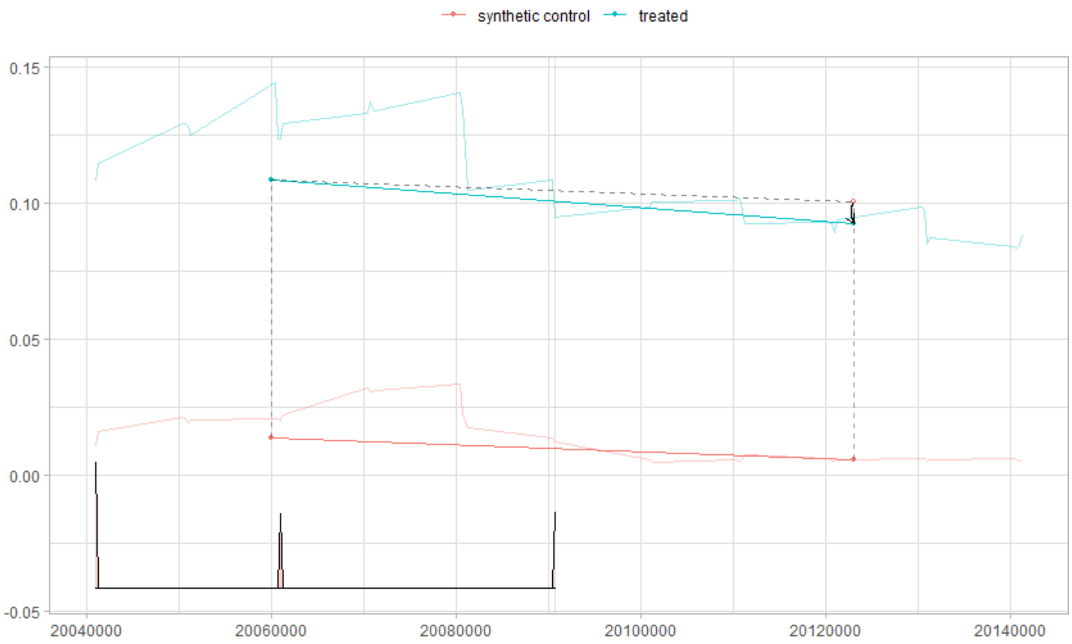
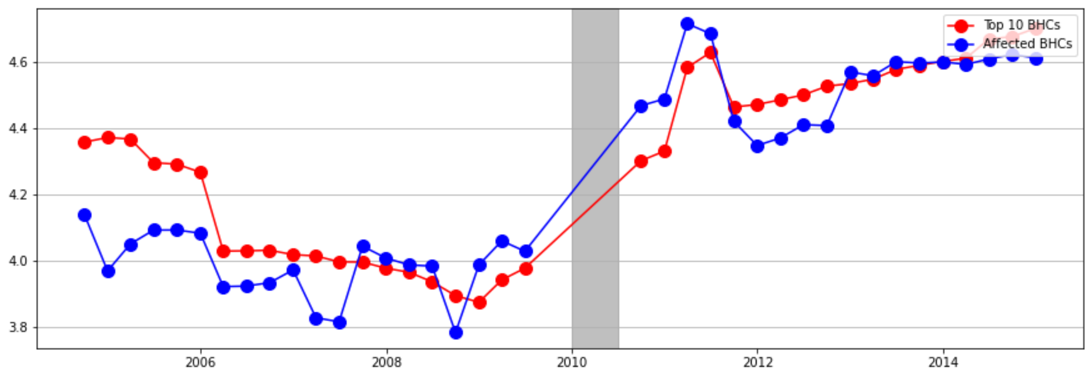


Figure 3: **Z-scores**

z-scores of top 10 BHCs and other Affected BHCs



The spike in 2011 is again caused by insufficient data points

Appendix: Stata Code

```
* Table 1: Summary statistics
/*import data*/
import delimited "D:\Python\jupyterCode\Economics\GP-2\***.csv", case(preserve) clear
/* Output descriptive statistics to Word, named sumamry */
logout, save(C:\Users\Thinkpad\Desktop\NUS\Analytics in Managerial Economics\group projects
\Group Project 2\summary) word dec(2) replace: tabstat  Trading_asset_ratio After_DFA
Affect Affect_pre_2007 Affect_BHC Total_assets Leverage_ratio Profitability Liquidity_ratio
 Deposit_ratio Cost_income_ratio Non_performing_loan_ratio Real_estate_loan_ratio CPP,
stats(mean sd min max n) c(s) f(%10.4f)

* Table 2: Panel A
/*import data*/
import delimited "D:\Python\jupyterCode\Economics\GP-2\***.csv", case(preserve) clear
/* Run regressions and save results */
reg Trading_asset_ratio After_DFA
est store m1
reg Trading_asset_ratio After_DFA Total_assets Leverage_ratio Profitability Liquidity_ratio
 Deposit_ratio Cost_income_ratio Non_performing_loan_ratio Real_estate_loan_ratio CPP
est store m2
reg Trading_asset_ratio After_DFA Affect After_DFAAffect Total_assets Leverage_ratio
Profitability Liquidity_ratio Deposit_ratio Cost_income_ratio Non_performing_loan_ratio
Real_estate_loan_ratio CPP
est store m3
areg Trading_asset_ratio After_DFAAffect Total_assets Leverage_ratio Profitability
Liquidity_ratio Deposit_ratio Cost_income_ratio Non_performing_loan_ratio
Real_estate_loan_ratio CPP i.Quater, absorb(BHC) vce(cluster BHC)
est store m4
/* Output to Word, named Baseline_tests */
outreg2 [m1 m2 m3 m4] using Baseline_tests.doc,replace tstat e(r2_a,F) bdec(4) tdec(2)
adjr2 dec(4)

* Table 2: Panel B
/*import data*/
import delimited "D:\Python\jupyterCode\Economics\GP-2\***.csv", case(preserve) clear
/* Run regressions and save results */
areg TAR After_DFAxAffectedBHC Total_assets Leverage_ratio Profitability Liquidity_ratio
Deposit_ratio Cost_income_ratio Non_performing_loan_ratio Real_estate_loan_ratio
CPP_recipient i.Date, absorb(Bank) vce(cluster Bank)
est store m
/* Output to Word, named Robustness */
outreg2 m using Robustness.doc,replace tstat e(r2,F) bdec(4) tdec(2) adjr2 dec(4)

* try xtreg
xtset Bank Date
xtreg TAR After_DFAxAffectedBHC Total_assets Leverage_ratio Profitability Liquidity_ratio
Deposit_ratio Cost_income_ratio Non_performing_loan_ratio Real_estate_loan_ratio
CPP_recipient i.Date, fe vce(cluster Bank)
```

Appendix: R code for SDID

```
Control variables:
"Profitability","Leverage_ratio","Total_assets","Non_performing_loan_rat
io","Cost_income_ratio",
"Deposit_ratio","Real_estate_loan_ratio","Liquidity_ratio",
"CPP_recipient"

Code:
did <- read.csv("C:/Users/Nick Zhang/Desktop/NUS 2021-
2022/Econ/GP2/Data/data.csv")

setup = synthdid::panel.matrices(did,"Bank","Date","TAR","Treated")

tem.X =
did[,c("Bank","Date","Profitability","Leverage_ratio","Total_assets","No
n_performing_loan_ratio",
             "Cost_income_ratio",
"Deposit_ratio","Real_estate_loan_ratio","Liquidity_ratio",
"CPP_recipient")]
temp.X = tem.X[order("Bank","Date")]

X =
array(matrix(unlist(tem.X[,c("Profitability","Leverage_ratio","Total_ass
ets","Non_performing_loan_ratio",
                            "Cost_income_ratio",
"Deposit_ratio","Real_estate_loan_ratio","Liquidity_ratio",
                            "CPP_recipient")]), nrow = nrow(setup$Y),
byrow =TRUE),dim=c(dim(setup$Y),1))

tau.hat.X = synthdid::synthdid_estimate(setup$Y, setup$N0, setup$T0, X)
print(summary(tau.hat.X))
se.X = sqrt(vcov(tau.hat.X, method='placebo'))
sprintf('95%% CI (%1.2f, %1.2f)', tau.hat.X-1.96*se.X,
tau.hat.X+1.96+se.X)
plot(tau.hat.X, se.method='placebo')

$estimate
[1] -0.008024389

$se
          [,1]
[1,] 0.01470447

$controls
```

```
       estimate 1
1069778     0.348
1020902     0.257
1053496     0.086
1048773     0.085
1049828     0.059
1074156     0.046
1096505     0.028

$periods
        estimate 1
20040930     0.457
20090630     0.272
20060930     0.271

$dimensions
        N1          N0 N0.effective          T1          T0 T0.effective
     4.000     166.000        4.754      18.000      20.000        2.809

> se.X = sqrt(vcov(tau.hat.X, method='placebo'))
> sprintf('95%% CI (%1.2f, %1.2f)', tau.hat.X-1.96*se.X,
tau.hat.X+1.96+se.X)
[1] "95% CI (-0.01, 1.95)"
> plot(tau.hat.X, se.method='placebo')
>
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv('DiD_data.csv')
df = pd.DataFrame(data)
```

In [3]:
```
df.head()
```

Out[3]:

	rssd9001	rssd9999	bhc_avgtradingratio	treat_3_b_avg	after_DFA_1	dep_roa1	dep_leverage	de
0	1020180	20040930	0.0	0	0	0.002772	0.081957	
1	1020180	20041231	0.0	0	0	0.003045	0.082480	
2	1020180	20050331	0.0	0	0	0.002616	0.082074	
3	1020180	20050630	0.0	0	0	0.002647	0.081712	
4	1020180	20050930	0.0	0	0	0.002867	0.082944	

In [4]:
```
df.shape
```

Out[4]: (81560, 14)

In [5]:
```
# rename columns
df.columns = ['Bank', 'Date', 'TAR', 'Affected_BHC', 'After_DFA', 'Profitability', '
```

In [6]:
```
df.describe().round(2)
```

Out[6]:

	Bank	Date	TAR	Affected_BHC	After_DFA	Profitability	Leverage_ratio	To
count	81560.00	81560.00	41442.00	81560.00	81560.00	56938.00	57017.00	
mean	1803535.29	20092215.02	0.00	0.01	0.45	0.00	0.09	
std	803001.25	33273.16	0.02	0.11	0.50	0.01	0.04	
min	1020180.00	20040930.00	0.00	0.00	0.00	-0.39	-0.76	
25%	1118434.00	20060930.00	0.00	0.00	0.00	0.00	0.07	
50%	1248304.00	20090331.00	0.00	0.00	0.00	0.00	0.09	
75%	2537957.00	20120930.00	0.00	0.00	1.00	0.00	0.11	
max	3836442.00	20150630.00	0.43	1.00	1.00	0.93	1.16	

Baseline all

$$TAR_{i,t} = \alpha + \beta_1 * AfterDFA + \beta_2 * TAR_{affectedness}$$
$$+\beta_3 * AfterDFA * TAR_{affectedness} + \gamma_i + ControlVariables_{i,t} + \delta_t + \epsilon_{i,t}$$

Basic DiD model

$$TAR_{i,t} = \beta_1 * AfterDFA + \beta_2 * TAR_{affectedness} + \beta_3 * AfterDFA * TAR_{affectedness} + \gamma_i + FixedEffects$$

AfterDFA = 1 for Q3'10 to Q2'15 and 0 for Q3'04 to Q2'09

Affectedness = average TAR where After DFA = 0

In [7]:

```
# reconstruct panel A
import statsmodels.api as sm
from statsmodels.formula.api import ols

panel_A_1 = ols('TAR ~ After_DFA', df).fit()
panel_A_1.summary()
```

Out[7]:

OLS Regression Results

Dep. Variable:	TAR	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	6.419
Date:	Mon, 25 Oct 2021	**Prob (F-statistic):**	0.0113
Time:	23:15:42	**Log-Likelihood:**	1.0329e+05
No. Observations:	41442	**AIC:**	-2.066e+05
Df Residuals:	41440	**BIC:**	-2.066e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.0024	0.000	19.207	0.000	0.002	0.003
After_DFA	0.0005	0.000	2.534	0.011	0.000	0.001

Omnibus:	71532.559	**Durbin-Watson:**	0.198
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	57471366.578
Skew:	12.333	**Prob(JB):**	0.00
Kurtosis:	183.761	**Cond. No.**	2.45

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [8]:

```
# reconstruct panel A2
panel_A_2 = ols('TAR ~ After_DFA + Profitability + Leverage_ratio + Total_assets + N
panel_A_2.summary()
```

Out[8]:

OLS Regression Results

Dep. Variable:	TAR	**R-squared:**	0.234

Model:	OLS	**Adj. R-squared:**	0.234
Method:	Least Squares	**F-statistic:**	1222.
Date:	Mon, 25 Oct 2021	**Prob (F-statistic):**	0.00
Time:	23:15:42	**Log-Likelihood:**	1.0595e+05
No. Observations:	40026	**AIC:**	-2.119e+05
Df Residuals:	40015	**BIC:**	-2.118e+05
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0207	0.002	-13.677	0.000	-0.024	-0.018
After_DFA	-0.0010	0.000	-4.868	0.000	-0.001	-0.001
Profitability	0.0321	0.022	1.484	0.138	-0.010	0.075
Leverage_ratio	-0.0495	0.003	-19.054	0.000	-0.055	-0.044
Total_assets	0.0043	7.27e-05	59.434	0.000	0.004	0.004
Non_performing_loan_ratio	0.0203	0.003	7.224	0.000	0.015	0.026
Cost_income_ratio	0.0010	0.000	2.921	0.003	0.000	0.002
Deposit_ratio	-0.0337	0.001	-41.532	0.000	-0.035	-0.032
Real_estate_loan_ratio	-0.0138	0.001	-23.362	0.000	-0.015	-0.013
Liquidity_ratio	-0.0006	0.002	-0.297	0.766	-0.005	0.003
CPP_recipient	-0.0016	0.000	-4.407	0.000	-0.002	-0.001

Omnibus:	63938.993	**Durbin-Watson:**	0.174
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	43593657.697
Skew:	10.478	**Prob(JB):**	0.00
Kurtosis:	163.312	**Cond. No.**	3.55e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.55e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [9]:

```
# add variable Affect
df['Affect'] = (df['TAR'].mask(~df['After_DFA'].eq(0)).groupby(df['Bank']).transform
```

In [10]:

```
# reconstruct panel A3
panel_A_3 = ols('TAR ~ After_DFA + Affect + Affect*After_DFA + Profitability + Lever
panel_A_3.summary()
```

Out[10]:

OLS Regression Results

Dep. Variable:	TAR	**R-squared:**	0.902

Model:	OLS	**Adj. R-squared:**	0.902
Method:	Least Squares	**F-statistic:**	3.061e+04
Date:	Mon, 25 Oct 2021	**Prob (F-statistic):**	0.00
Time:	23:15:42	**Log-Likelihood:**	1.4705e+05
No. Observations:	40026	**AIC:**	-2.941e+05
Df Residuals:	40013	**BIC:**	-2.940e+05
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0030	0.001	-5.469	0.000	-0.004	-0.002
After_DFA	-2.039e-05	7.68e-05	-0.266	0.791	-0.000	0.000
Affect	0.9925	0.002	402.166	0.000	0.988	0.997
Affect:After_DFA	-0.1611	0.003	-52.818	0.000	-0.167	-0.155
Profitability	-0.0051	0.008	-0.656	0.512	-0.020	0.010
Leverage_ratio	0.0013	0.001	1.338	0.181	-0.001	0.003
Total_assets	0.0002	2.72e-05	8.258	0.000	0.000	0.000
Non_performing_loan_ratio	0.0017	0.001	1.724	0.085	-0.000	0.004
Cost_income_ratio	0.0002	0.000	1.465	0.143	-6.33e-05	0.000
Deposit_ratio	0.0004	0.000	1.298	0.194	-0.000	0.001
Real_estate_loan_ratio	-0.0007	0.000	-3.500	0.000	-0.001	-0.000
Liquidity_ratio	-0.0005	0.001	-0.709	0.479	-0.002	0.001
CPP_recipient	-0.0002	0.000	-1.936	0.053	-0.000	3.08e-06

Omnibus:	51911.637	**Durbin-Watson:**	0.524
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	103525182.575
Skew:	6.371	**Prob(JB):**	0.00
Kurtosis:	251.822	**Cond. No.**	3.55e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.55e+03. This might indicate that there are strong multicollinearity or other numerical problems.

In [11]:

```
# reconstructing panel A4
from linearmodels import PanelOLS

df_FE = df.copy()
df_FE = df_FE.set_index(['Bank', 'Date'])
df_FE['After_DFAxAffect'] =  df_FE['After_DFA'] * df_FE['Affect']
df_FE = df_FE.dropna()
exog = df_FE.drop(columns = ['TAR', 'After_DFA', 'Affected_BHC','Affect'])
```

```python
# Regression
FE = PanelOLS(df_FE.TAR, exog,
              entity_effects = True,
              time_effects=True,
              drop_absorbed=True
              )

# Result
result = FE.fit(cov_type = 'clustered',
             cluster_entity=True,
             )

result
```

Out[11]:

PanelOLS Estimation Summary

Dep. Variable:	TAR	**R-squared:**	0.0824
Estimator:	PanelOLS	**R-squared (Between):**	-0.5099
No. Observations:	40026	**R-squared (Within):**	0.0805
Date:	Mon, Oct 25 2021	**R-squared (Overall):**	-0.5159
Time:	23:15:43	**Log-likelihood**	1.525e+05
Cov. Estimator:	Clustered		
		F-statistic:	337.16
Entities:	2473	**P-value**	0.0000
Avg Obs:	16.185	**Distribution:**	F(10,37551)
Min Obs:	0.0000		
Max Obs:	38.000	**F-statistic (robust):**	4.6735
		P-value	0.0000
Time periods:	38	**Distribution:**	F(10,37551)
Avg Obs:	1053.3		
Min Obs:	334.00		
Max Obs:	2281.0		

Parameter Estimates

	Parameter	**Std. Err.**	**T-stat**	**P-value**	**Lower CI**	**Upper CI**
Profitability	0.0064	0.0080	0.8047	0.4210	-0.0093	0.0221
Leverage_ratio	0.0048	0.0038	1.2636	0.2064	-0.0027	0.0123
Total_assets	-0.0002	0.0005	-0.3189	0.7498	-0.0011	0.0008
Non_performing_loan_ratio	0.0010	0.0019	0.5054	0.6133	-0.0028	0.0047
Cost_income_ratio	0.0003	0.0002	1.9046	0.0568	-1.013e-05	0.0007
Deposit_ratio	0.0007	0.0011	0.6115	0.5409	-0.0015	0.0029
Real_estate_loan_ratio	-0.0103	0.0046	-2.2543	0.0242	-0.0192	-0.0013
Liquidity_ratio	-0.0012	0.0028	-0.4406	0.6595	-0.0068	0.0043
CPP_recipient	-5.492e-06	0.0004	-0.0152	0.9878	-0.0007	0.0007

After_DFAxAffect	-0.2024	0.0470	-4.3056	0.0000	-0.2945	-0.1102

F-test for Poolability: 86.393
P-value: 0.0000
Distribution: F(2464,37551)

Included effects: Entity, Time
id: 0x7fdb618a2730

Robustness Tests

Panel B1

In [12]:

```
# reconstructing panel B1

df_B1 = df.copy()
df_B1 = df_B1.set_index(['Bank', 'Date'])
df_B1['After_DFAxAffectedBHC'] =  df_B1['After_DFA'] * df_B1['Affected_BHC']
df_B1 = df_B1.dropna()
exog = df_B1.drop(columns = ['TAR', 'After_DFA', 'Affected_BHC','Affect'])


# Regression
FE = PanelOLS(df_B1.TAR, exog,
              entity_effects = True,
              time_effects=True,
              drop_absorbed=True
              )

# Result
result = FE.fit(cov_type = 'clustered',
             cluster_entity=True,
             cluster_time=True
             )

result
```

Out[12]:

PanelOLS Estimation Summary

Dep. Variable:	TAR	**R-squared:**	0.0532
Estimator:	PanelOLS	**R-squared (Between):**	-0.3306
No. Observations:	40026	**R-squared (Within):**	0.0519
Date:	Mon, Oct 25 2021	**R-squared (Overall):**	-0.3389
Time:	23:15:44	**Log-likelihood**	1.519e+05
Cov. Estimator:	Clustered		
		F-statistic:	211.07
Entities:	2473	**P-value**	0.0000
Avg Obs:	16.185	**Distribution:**	F(10,37551)
Min Obs:	0.0000		

Max Obs:	38.000	**F-statistic (robust):**	2.0684
		P-value	0.0234
Time periods:	38	**Distribution:**	F(10,37551)
Avg Obs:	1053.3		
Min Obs:	334.00		
Max Obs:	2281.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
Profitability	0.0062	0.0070	0.8821	0.3777	-0.0076	0.0200
Leverage_ratio	0.0022	0.0046	0.4661	0.6412	-0.0069	0.0113
Total_assets	-0.0001	0.0005	-0.2183	0.8272	-0.0011	0.0009
Non_performing_loan_ratio	-1.796e-05	0.0020	-0.0091	0.9927	-0.0039	0.0038
Cost_income_ratio	0.0003	0.0002	1.8434	0.0653	-2.138e-05	0.0007
Deposit_ratio	0.0008	0.0013	0.5909	0.5546	-0.0018	0.0033
Real_estate_loan_ratio	-0.0096	0.0046	-2.0601	0.0394	-0.0187	-0.0005
Liquidity_ratio	-0.0004	0.0031	-0.1326	0.8945	-0.0065	0.0057
CPP_recipient	-0.0002	0.0005	-0.4162	0.6773	-0.0011	0.0007
After_DFAxAffectedBHC	-0.0234	0.0087	-2.6758	0.0075	-0.0405	-0.0063

F-test for Poolability: 113.32
P-value: 0.0000
Distribution: F(2464,37551)

Included effects: Entity, Time
id: 0x7fdb6142b550

Panel B2

In [13]:

```
# reconstruction panel B2
# pg 15 "based on a simple logit regression on our vector of control variables"
from pymatch.Matcher import Matcher

df_B2 = df.copy()
df_B2['After_DFAxAffectedBHC'] =  df_B2['After_DFA'] * df_B2['Affected_BHC']
df_B2 = df_B2.drop(columns = ['After_DFA', 'Affected_BHC','Affect'])
test = df_B2[df_B2.After_DFAxAffectedBHC == 1]
control = df_B2[df_B2.After_DFAxAffectedBHC == 0]

m = Matcher(test, control, yvar = 'After_DFAxAffectedBHC', exclude=['TAR', 'Bank', '
```

```
Formula:
After_DFAxAffectedBHC ~ Profitability+Leverage_ratio+Total_assets+Non_performing_loa
n_ratio+Cost_income_ratio+Deposit_ratio+Real_estate_loan_ratio+Liquidity_ratio+CPP_r
ecipient
```

```
n majority: 39767
n minority: 259
```

In [14]:

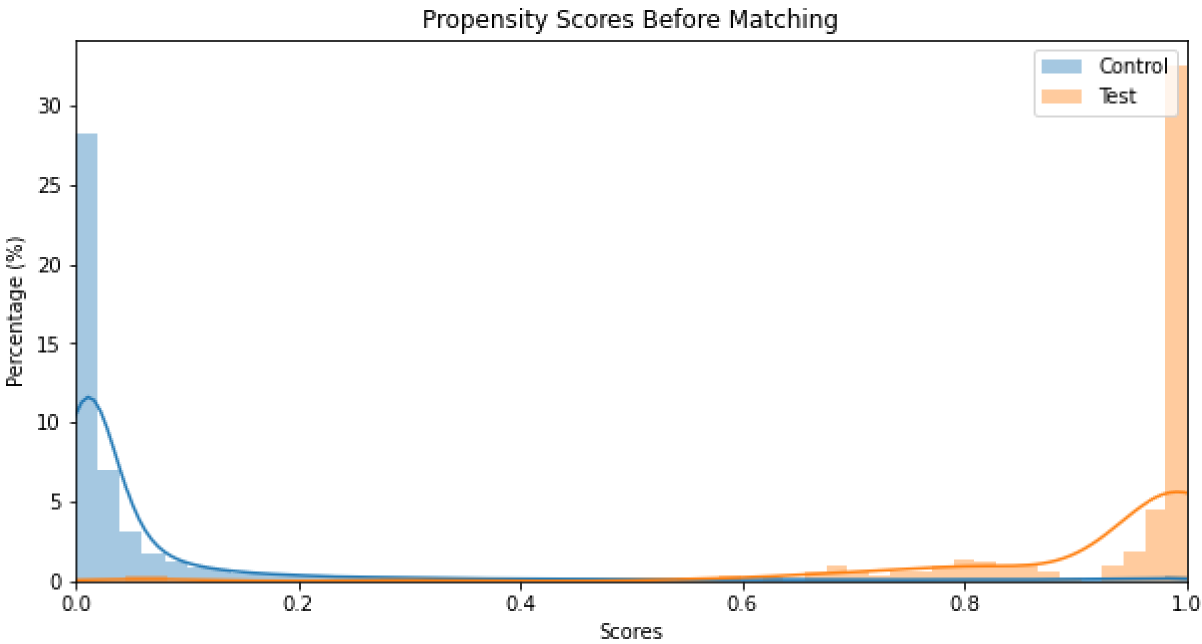
```
m.fit_scores(balance=True, nmodels=100)
```

```
Fitting Models on Balanced Samples: 31\100
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/genmod/families/links.py:188:
RuntimeWarning: overflow encountered in exp
  t = np.exp(-z)
Fitting Models on Balanced Samples: 100\100
Average Accuracy: 96.16%
```

In [15]:

```
import matplotlib.pyplot as plt
m.predict_scores()
m.plot_scores()
```

```
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [16]:

```
m.match(method="min", nmatches=1, threshold=0.0001)
m.record_frequency()
```

Out[16]:

	freq	n_records
0	1	381
1	2	29
2	3	8
3	4	1
4	5	4

	freq	n_records
5	6	2
6	9	1
7	10	1

In [17]:

```
m.matched_data.head(10)
```

Out[17]:

	Bank	Date	TAR	Profitability	Leverage_ratio	Total_assets	Non_performing_loan
0	1032473	20120630	0.069622	-0.001221	0.102024	17.989805	0.0
20801	1245415	20141231	0.033597	0.000946	0.121822	18.568062	0.0
1	1032473	20120930	0.058777	0.001633	0.091507	18.108091	0.0
11010	1111435	20051231	0.029548	0.002466	0.061725	18.400434	0.0
2	1032473	20121231	0.063189	0.002133	0.087786	18.121574	0.0
11011	1111435	20060331	0.034140	0.002887	0.063212	18.461699	0.0
3	1032473	20130331	0.052756	0.001280	0.088133	18.139948	0.0
33908	2816906	20090630	0.299876	0.001631	0.004784	19.719099	0.0
4	1032473	20130630	0.045880	0.001478	0.090702	18.092066	0.0
11013	1111435	20060930	0.032520	0.002587	0.062736	18.537058	0.0

In [18]:

```
df_B2m = m.matched_data.copy()
df_B2m = df_B2m.set_index(['Bank', 'Date'])
df_B2m = df_B2m.dropna()
exog = df_B2m.drop(columns = ['TAR', 'scores', 'match_id', 'record_id'])

# Regression
FE = PanelOLS(df_B2m.TAR, exog,
              entity_effects = True,
              time_effects=True,
              drop_absorbed=True
              )

# Result
result = FE.fit(cov_type = 'clustered',
             cluster_entity=True,
             )

result
```

Out[18]:

PanelOLS Estimation Summary

Dep. Variable:	TAR	**R-squared:**	0.0612
Estimator:	PanelOLS	**R-squared (Between):**	-5.3254
No. Observations:	518	**R-squared (Within):**	0.1064
Date:	Mon, Oct 25 2021	**R-squared (Overall):**	-3.0880
Time:	23:15:51	**Log-likelihood**	1408.0
Cov. Estimator:	Clustered		

		F-statistic:	2.5604
Entities:	78	**P-value**	0.0052
Avg Obs:	6.6410	**Distribution:**	F(10,393)
Min Obs:	1.0000		
Max Obs:	69.000	**F-statistic (robust):**	0.7941
		P-value	0.6346
Time periods:	38	**Distribution:**	F(10,393)
Avg Obs:	13.632		
Min Obs:	3.0000		
Max Obs:	28.000		

Parameter Estimates

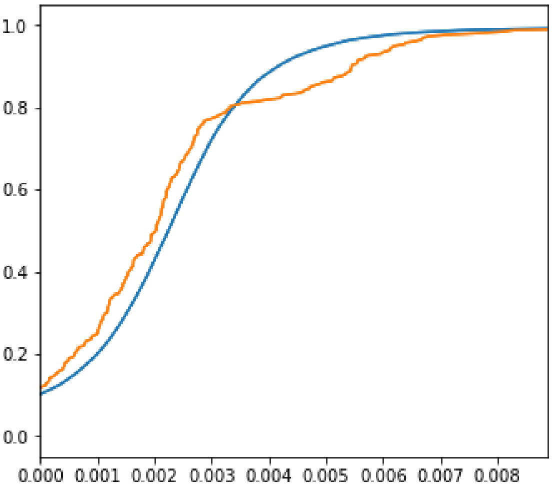
	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
Profitability	0.4812	0.5539	0.8688	0.3855	-0.6078	1.5703
Leverage_ratio	0.1340	0.2584	0.5186	0.6043	-0.3740	0.6421
Total_assets	-0.0075	0.0174	-0.4329	0.6654	-0.0418	0.0267
Non_performing_loan_ratio	0.0638	0.1452	0.4390	0.6609	-0.2218	0.3493
Cost_income_ratio	-0.0003	0.0007	-0.4459	0.6559	-0.0016	0.0010
Deposit_ratio	-0.0449	0.0569	-0.7891	0.4305	-0.1567	0.0669
Real_estate_loan_ratio	-0.0680	0.0871	-0.7814	0.4350	-0.2392	0.1031
Liquidity_ratio	-0.0068	0.0442	-0.1531	0.8784	-0.0936	0.0800
CPP_recipient	-0.0014	0.0212	-0.0658	0.9476	-0.0431	0.0403
After_DFAxAffectedBHC	-0.0194	0.0114	-1.7026	0.0894	-0.0417	0.0030

F-test for Poolability: 77.019
P-value: 0.0000
Distribution: F(114,393)

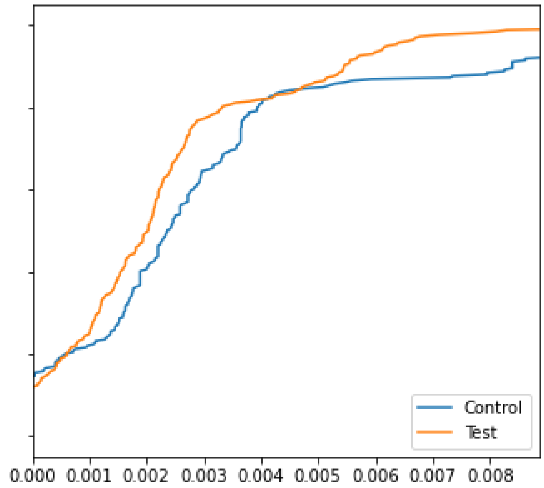
Included effects: Entity, Time
id: 0x7fdb49e13280

In [19]:
```
cc = m.compare_continuous(return_table=True)
```

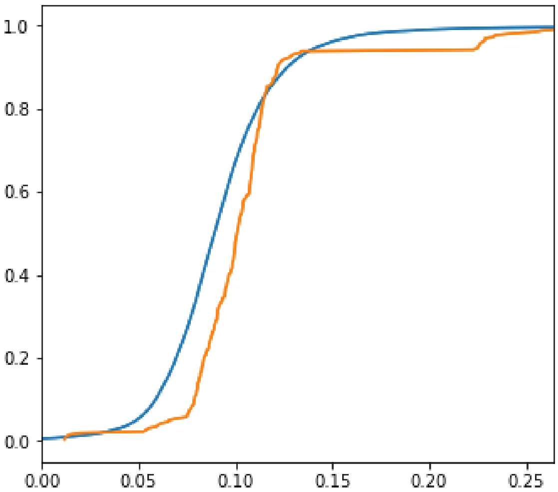
ECDF for Profitability before Matching
KS p-value: 0.001
Grouped Perm p-value: 0.942
Std. Median Difference: -0.04376404088711261
Std. Mean Difference: 0.04388609077701806



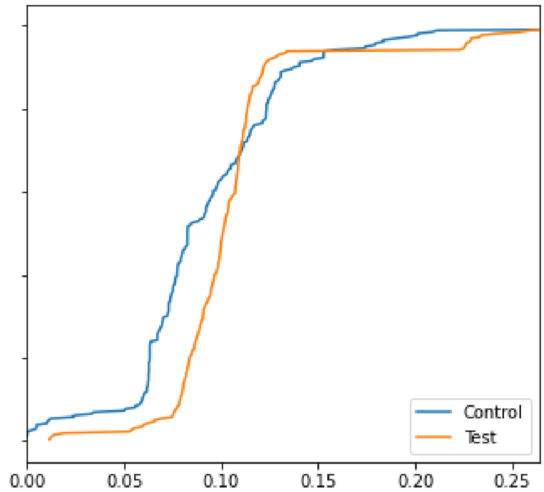
ECDF for Profitability after Matching
KS p-value: 0.003
Grouped Perm p-value: 0.189
Std. Median Difference: -0.052001985813449156
Std. Mean Difference: -0.10543907478129161



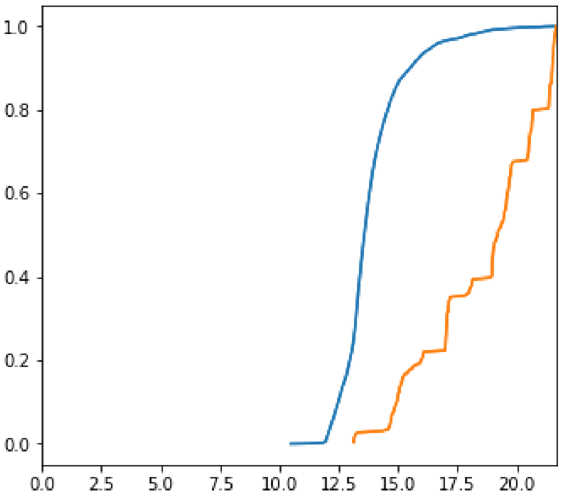
ECDF for Leverage_ratio before Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: 0.3335159705631456
Std. Mean Difference: 0.4045583451886257



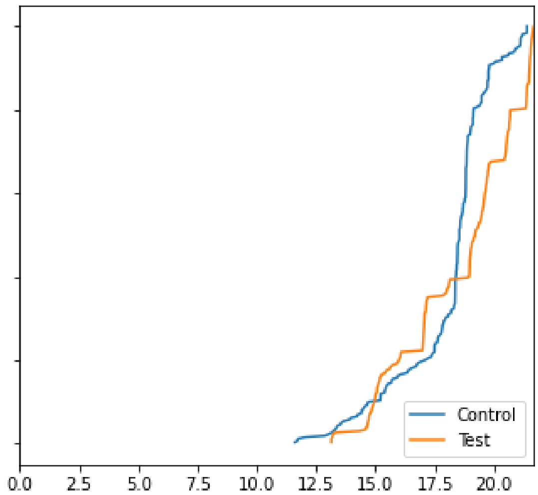
ECDF for Leverage_ratio after Matching
KS p-value: 0.0
Grouped Perm p-value: 0.135
Std. Median Difference: 0.40102415755835275
Std. Mean Difference: 0.27208953484573273



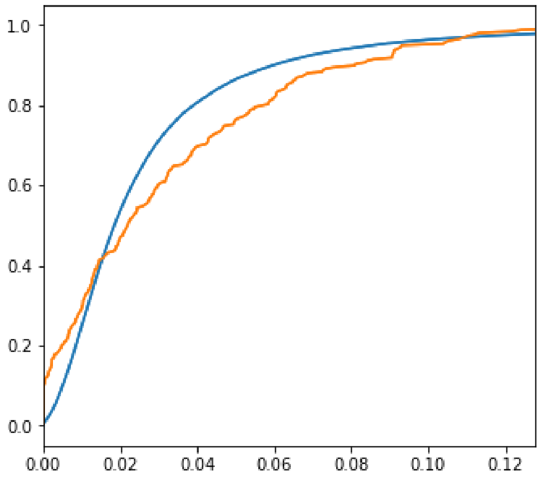
ECDF for Total_assets before Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: 4.030860326033872
Std. Mean Difference: 3.4025931723588805



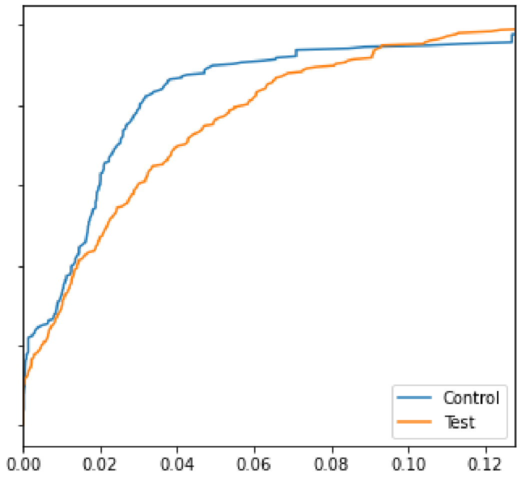
ECDF for Total_assets after Matching
KS p-value: 0.0
Grouped Perm p-value: 0.0
Std. Median Difference: 0.2968003789577051
Std. Mean Difference: 0.23734272924165167



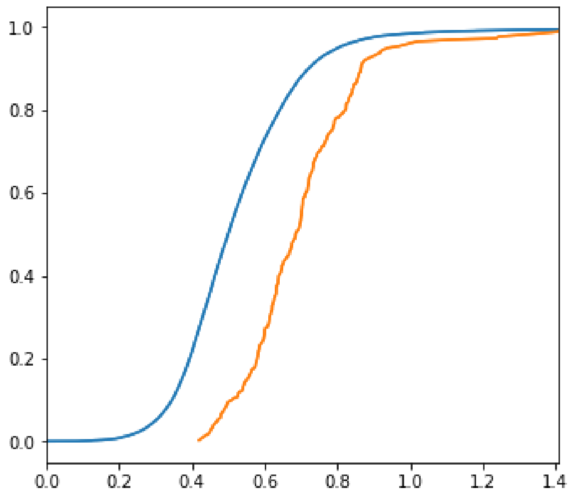
ECDF for Non_performing_loan_ratio before Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: 0.10386101811652804
Std. Mean Difference: 0.1078166768049816



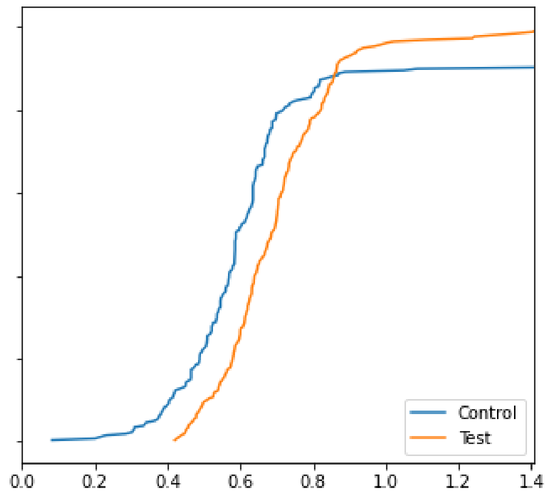
ECDF for Non_performing_loan_ratio after Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: 0.10284612500774193
Std. Mean Difference: 0.07801214313595926



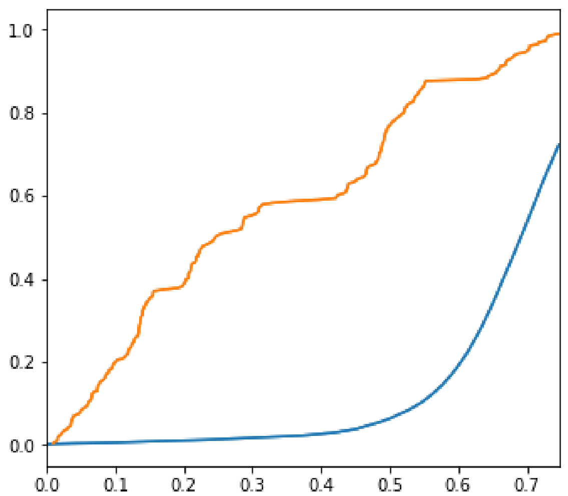
ECDF for Cost_income_ratio before Matching
KS p-value: 0.0
Grouped Perm p-value: 0.873
Std. Median Difference: 0.663080287681667
Std. Mean Difference: 0.8495232025625585



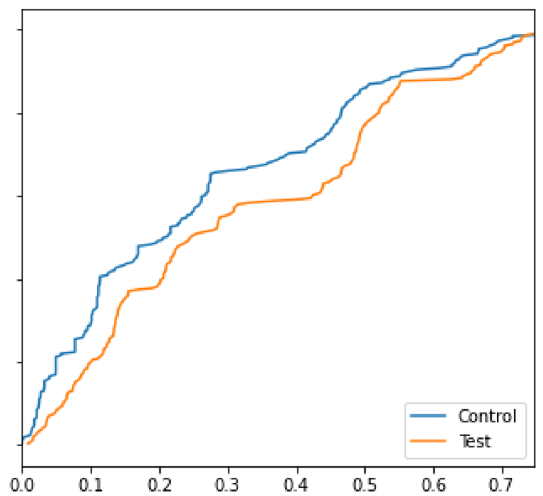
ECDF for Cost_income_ratio after Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: 0.09703044838040874
Std. Mean Difference: -0.05150183776678503



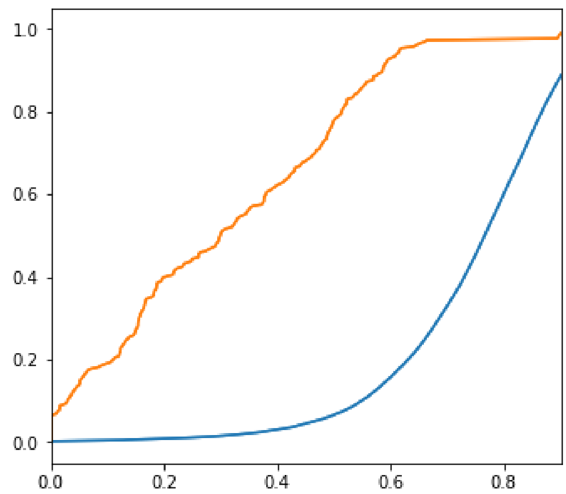
ECDF for Deposit_ratio before Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: -3.5400335565969665
Std. Mean Difference: -2.868288388587842



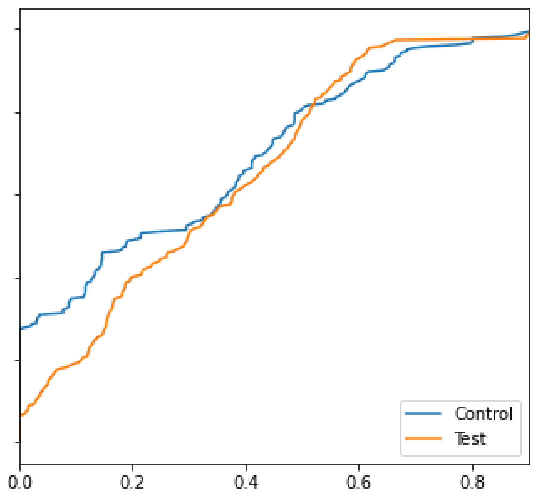
ECDF for Deposit_ratio after Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: 0.15587618670367223
Std. Mean Difference: 0.2599032559821958



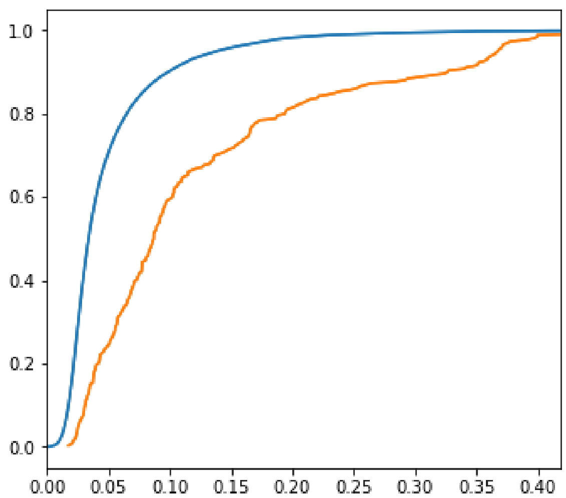
ECDF for Real_estate_loan_ratio before Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: -2.9636666333441726
Std. Mean Difference: -2.709518256669427



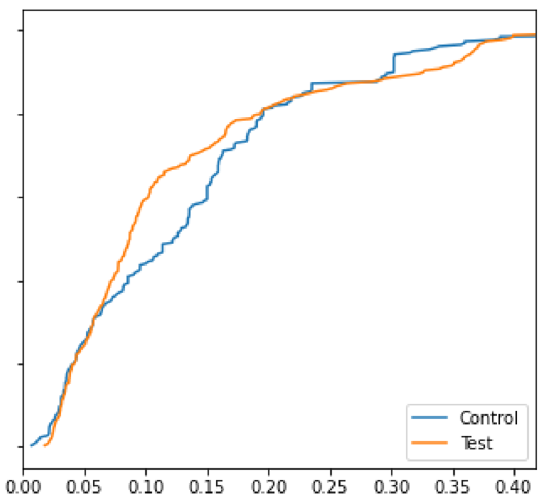
ECDF for Real_estate_loan_ratio after Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: 0.3558178485281236
Std. Mean Difference: 0.14852031326982315



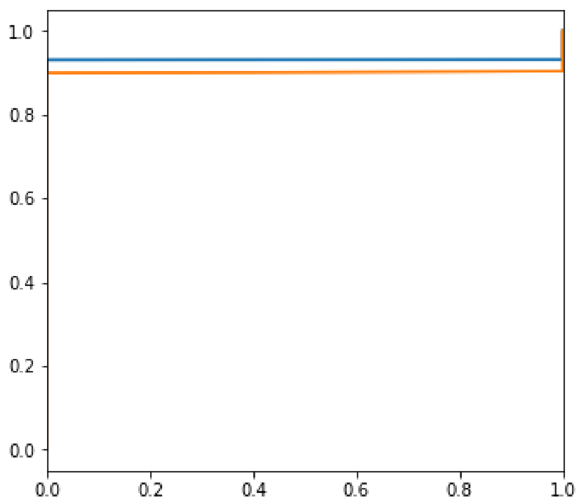
ECDF for Liquidity_ratio before Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: 1.1063826481497958
Std. Mean Difference: 1.5945472466458304



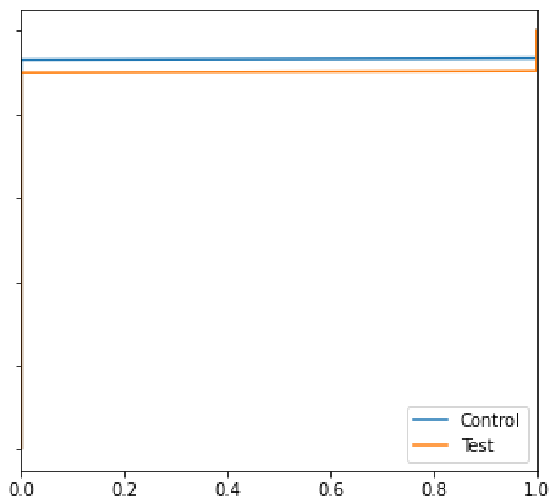
ECDF for Liquidity_ratio after Matching
KS p-value: 0.0
Grouped Perm p-value: 1.0
Std. Median Difference: -0.3488789328798527
Std. Mean Difference: -0.09203123449121661



ECDF for CPP_recipient before Matching
KS p-value: 0.055
Grouped Perm p-value: 1.0
Std. Median Difference: 0.0
Std. Mean Difference: 0.1234164858426771



ECDF for CPP_recipient after Matching
KS p-value: 0.236
Grouped Perm p-value: 1.0
Std. Median Difference: 0.0
Std. Mean Difference: 0.11079099776623341



Panel B3

In [20]:

```
# reconstructing panel B3

# add variable Affect_pre2007
df_B3 = df.copy()
df_B3['pre2007'] = np.where(df_B3['Date']<20070000, 1, 0)
df_B3['Affect_pre2007'] = (df_B3['TAR'].mask(~df_B3['pre2007'].eq(1)).groupby(df_B3[

df_B3 = df_B3.set_index(['Bank', 'Date'])
df_B3['After_DFAxAffect_pre2007'] =  df_B3['After_DFA'] * df_B3['Affect_pre2007']
df_B3 = df_B3.dropna()
exog = df_B3.drop(columns = ['TAR', 'After_DFA', 'Affected_BHC','Affect', 'Affect_pr

print(df_B3.TAR.shape)
print(exog.shape)

# Regression
FE = PanelOLS(df_B3.TAR, exog,
              entity_effects = True,
              time_effects=True,
              drop_absorbed=True
              )

# Result
result = FE.fit(cov_type = 'clustered',
             cluster_entity=True,
             )

result
```

```
(38783,)
(38783, 10)
```

Out[20]:

PanelOLS Estimation Summary

Dep. Variable:	TAR	**R-squared:**	0.0646
Estimator:	PanelOLS	**R-squared (Between):**	-0.3925
No. Observations:	38783	**R-squared (Within):**	0.0639
Date:	Mon, Oct 25 2021	**R-squared (Overall):**	-0.4237
Time:	23:19:04	**Log-likelihood**	1.486e+05
Cov. Estimator:	Clustered		
		F-statistic:	251.15
Entities:	2433	**P-value**	0.0000
Avg Obs:	15.940	**Distribution:**	F(10,36378)
Min Obs:	0.0000		
Max Obs:	38.000	**F-statistic (robust):**	3.9808
		P-value	0.0000
Time periods:	38	**Distribution:**	F(10,36378)
Avg Obs:	1020.6		
Min Obs:	332.00		
Max Obs:	2281.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
Profitability	0.0123	0.0079	1.5569	0.1195	-0.0032	0.0277
Leverage_ratio	0.0056	0.0040	1.4024	0.1608	-0.0022	0.0133
Total_assets	-0.0002	0.0005	-0.4540	0.6498	-0.0012	0.0007
Non_performing_loan_ratio	0.0021	0.0020	1.0511	0.2932	-0.0018	0.0059
Cost_income_ratio	0.0004	0.0002	2.1020	0.0356	2.779e-05	0.0008
Deposit_ratio	0.0004	0.0011	0.3283	0.7427	-0.0018	0.0025
Real_estate_loan_ratio	-0.0068	0.0036	-1.8681	0.0618	-0.0140	0.0003
Liquidity_ratio	-0.0024	0.0029	-0.8323	0.4053	-0.0080	0.0032
CPP_recipient	0.0001	0.0004	0.2956	0.7675	-0.0006	0.0008
After_DFAxAffect_pre2007	-0.2052	0.0578	-3.5537	0.0004	-0.3184	-0.0920

F-test for Poolability: 80.928
P-value: 0.0000
Distribution: F(2394,36378)

Included effects: Entity, Time
id: 0x7fdb50900520

Panel B4

In [21]:

```
# reconstruct panel B4

df_B4 = df.copy()
df_B4['pre2007'] = np.where(df_B4['Date']<20070000, 1, 0)
df_B4['Affect_pre2007'] = (df_B4['TAR'].mask(~df_B4['pre2007'].eq(1)).groupby(df_B4[

df_B4 = df_B4[df_B4['TAR']>0]
df_B4 = df_B4.set_index(['Bank', 'Date'])
df_B4['After_DFAxAffect_pre2007'] =  df_B4['After_DFA'] * df_B4['Affect_pre2007']
df_B4 = df_B4.dropna()
exog = df_B4.drop(columns = ['TAR', 'After_DFA', 'Affected_BHC','Affect', 'Affect_pr

print(df_B4.TAR.shape)
print(exog.shape)

# Regression
FE = PanelOLS(df_B4.TAR, exog,
              entity_effects = True,
              time_effects=True,
              drop_absorbed=True
              )

# Result
result = FE.fit(cov_type = 'clustered',
             cluster_entity=True
             )
```

```
result
```

```
(4493,)
(4493, 10)
```

Out[21]:

PanelOLS Estimation Summary

Dep. Variable:	TAR	**R-squared:**	0.0744
Estimator:	PanelOLS	**R-squared (Between):**	0.0851
No. Observations:	4493	**R-squared (Within):**	0.0785
Date:	Mon, Oct 25 2021	**R-squared (Overall):**	0.0854
Time:	23:19:04	**Log-likelihood**	1.307e+04
Cov. Estimator:	Clustered		
		F-statistic:	33.238
Entities:	338	**P-value**	0.0000
Avg Obs:	13.293	**Distribution:**	F(10,4136)
Min Obs:	0.0000		
Max Obs:	38.000	**F-statistic (robust):**	3.1648
		P-value	0.0005
Time periods:	38	**Distribution:**	F(10,4136)
Avg Obs:	118.24		
Min Obs:	55.000		
Max Obs:	136.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
Profitability	0.0431	0.0519	0.8305	0.4063	-0.0587	0.1449
Leverage_ratio	0.0361	0.0497	0.7253	0.4683	-0.0614	0.1335
Total_assets	0.0014	0.0044	0.3257	0.7447	-0.0072	0.0101
Non_performing_loan_ratio	0.0328	0.0226	1.4560	0.1455	-0.0114	0.0771
Cost_income_ratio	0.0012	0.0005	2.4827	0.0131	0.0003	0.0022
Deposit_ratio	0.0012	0.0078	0.1585	0.8741	-0.0140	0.0165
Real_estate_loan_ratio	-0.0315	0.0213	-1.4781	0.1394	-0.0733	0.0103
Liquidity_ratio	-0.0107	0.0162	-0.6612	0.5085	-0.0425	0.0211
CPP_recipient	-0.0006	0.0024	-0.2671	0.7894	-0.0053	0.0040
After_DFAxAffect_pre2007	-0.1858	0.0600	-3.0988	0.0020	-0.3034	-0.0683

F-test for Poolability: 74.340
P-value: 0.0000
Distribution: F(346,4136)

Included effects: Entity, Time
id: 0x7fdb4ecaa880

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv('DiD_data.csv')
df = pd.DataFrame(data)
```

In [3]:
```
# rename columns
df.columns = ['Bank', 'Date', 'TAR', 'Affected_BHC', 'After_DFA', 'Profitability', '
```

In [4]:
```
# establish top 10 Affected BHCs by trading assets

# create a column that shows total trading assets of banks
df_top10 = df.copy()
df_top10['trading_assets'] = df_top10['TAR'] * np.exp(df_top10['Total_assets'])
df_top10
```

Out[4]:

	Bank	Date	TAR	Affected_BHC	After_DFA	Profitability	Leverage_ratio	Total_assets
0	1020180	20040930	0.0	0	0	0.002772	0.081957	15.601202
1	1020180	20041231	0.0	0	0	0.003045	0.082480	15.630583
2	1020180	20050331	0.0	0	0	0.002616	0.082074	15.644925
3	1020180	20050630	0.0	0	0	0.002647	0.081712	15.679702
4	1020180	20050930	0.0	0	0	0.002867	0.082944	15.661868
...	...	...	...	...	...	...	...	..
81555	3832583	20110331	NaN	0	1	NaN	NaN	NaN
81556	3832583	20110630	NaN	0	1	NaN	NaN	13.061935
81557	3832583	20150331	0.0	0	1	0.005248	0.225874	13.562950
81558	3832583	20150630	0.0	0	1	0.005353	0.226806	13.558450
81559	3836442	20090331	NaN	0	0	NaN	NaN	13.456915

81560 rows × 15 columns

In [5]:
```
# We use the mean trading assets before 2007 to define the top 10
top10 = df_top10[(df_top10['Affected_BHC'] == 1) & (df_top10['Date'] < 20080000)].gr
```

In [6]:
```
# top 10 banks by trading assets by bank number
top10 = top10.sort_values(ascending=False).head(10)
df_top10 = df_top10[df_top10['Bank'].isin(top10.index)]
df_top10
```

Out[6]:

	Bank	Date	TAR	Affected_BHC	After_DFA	Profitability	Leverage_ratio	Total_a
1361	1039502	20040930	0.235039	1	0	0.001450	0.077595	20.85
1362	1039502	20041231	0.251247	1	0	0.001451	0.092131	20.86

	Bank	Date	TAR	Affected_BHC	After_DFA	Profitability	Leverage_ratio	Total_a:
1363	1039502	20050331	0.254006	1	0	0.001939	0.090340	20.88
1364	1039502	20050630	0.251873	1	0	0.000846	0.089686	20.88
1365	1039502	20050930	0.249962	1	0	0.002129	0.089087	20.90
...	...	...	...	...	...	...	...	
81133	3375370	20140630	NaN	1	1	NaN	NaN	
81134	3375370	20140930	NaN	1	1	NaN	NaN	
81135	3375370	20141231	NaN	1	1	NaN	NaN	
81136	3375370	20150331	NaN	1	1	NaN	NaN	
81137	3375370	20150630	NaN	1	1	NaN	NaN	

282 rows × 15 columns

In [7]:

```
# mean of the TAR of the top 10 group
top10plot = df_top10.groupby('Date')['TAR'].mean()
```

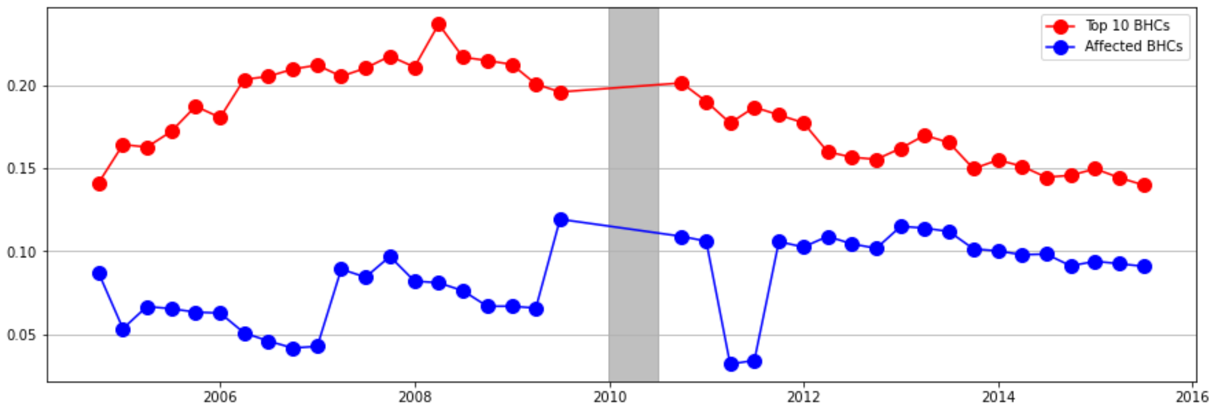
In [8]:

```
df_nottop10 = df[~df['Bank'].isin(top10.index)]
df_nottop10 = df_nottop10[(df_nottop10['Affected_BHC']==1)]

abhc_plot = df_nottop10.groupby('Date')['TAR'].mean()
```

In [9]:

```
# plot graph
# DFA announced 20090930 to 20100630

x_top10 = pd.to_datetime(top10plot.index, format='%Y%m%d', errors='ignore')
x_nottop10 = pd.to_datetime(abhc_plot.index, format='%Y%m%d', errors='ignore')

plt.figure(figsize=(15,5))
plt.grid(axis = 'y')
plt.axvspan(pd.to_datetime('20091231', format='%Y%m%d'), pd.to_datetime('20100630',
plt.plot(x_top10,top10plot.values,'o-', color='red', markersize=10, label='Top 10 BH
plt.plot(x_nottop10,abhc_plot.values,'o-', color='blue', markersize=10, label='Affec
plt.legend(loc="upper right")
```

Out[9]: <matplotlib.legend.Legend at 0x7fe2b3120d60>



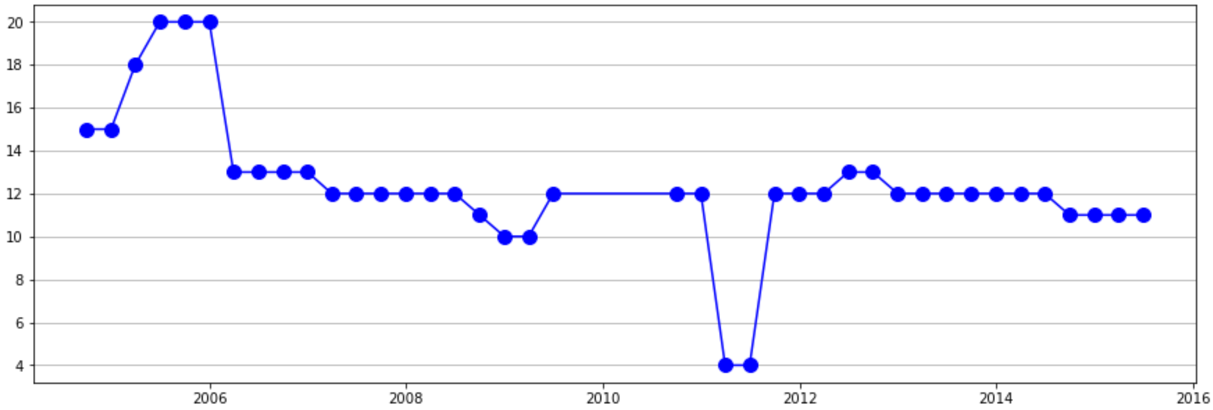
In [10]:

```
# we see that the anomaly in the two datapoints after DFA is due to insufficent data
```

```
df_nottop10_count = df_nottop10.groupby('Date')['TAR'].count()

x_count = pd.to_datetime(df_nottop10_count.index, format='%Y%m%d', errors='ignore')
plt.figure(figsize=(15,5))
plt.grid(axis = 'y')
plt.plot(x_count,df_nottop10_count.values,'o-', color='blue', markersize=10)
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x7fe2b6d44e50>]
```

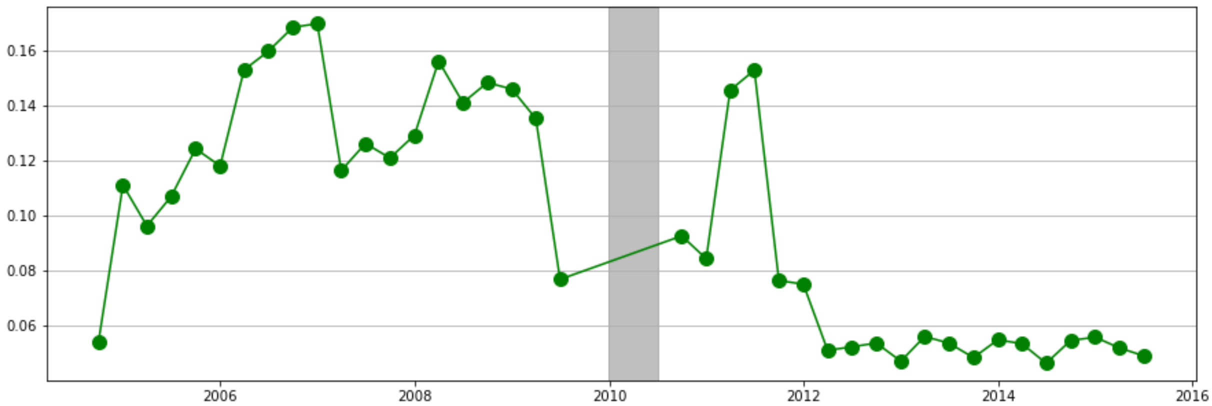


In [11]:

```
# Plotting the difference in TAr between the top 10 banks and the rest of the affect
diff = top10plot - abhc_plot

x_diff = pd.to_datetime(diff.index, format='%Y%m%d', errors='ignore')
plt.figure(figsize=(15,5))
plt.grid(axis = 'y')
plt.axvspan(pd.to_datetime('20091231', format='%Y%m%d'), pd.to_datetime('20100630',
plt.plot(x_diff,diff.values,'o-', color='green', markersize=10)
```

Out[11]:

```
[<matplotlib.lines.Line2D at 0x7fe2b6dfc820>]
```



In [2]:

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

In [2]:

```
data = pd.read_csv('DiD_data.csv')
df = pd.DataFrame(data)

# rename columns
df.columns = ['Bank', 'Date', 'TAR', 'Affected_BHC', 'After_DFA', 'Profitability', '
```

In [3]:

```
# Paul
# calculating z-score: (ROA+CAR)/std(ROA) [CAR=Leverage ratio, ROA=Profitability]

# calculating stdROA
df['stdROA1'] = df[df['After_DFA'].eq(0)].groupby('Bank')['Profitability'].transform
df['stdROA2'] = df[df['After_DFA'].eq(1)].groupby('Bank')['Profitability'].transform
df['stdROA1'] = df['stdROA1'].fillna(0)
df['stdROA2'] = df['stdROA2'].fillna(0)
df['stdROA'] = (df['stdROA1'] - df['stdROA2']).abs()
df = df.drop(['stdROA1', 'stdROA2'], axis=1)

# drop banks with stdROA == 0
df = df[df['stdROA']!=0]
df=df.dropna()
df['zscore'] = np.log((df['Profitability']+df['Leverage_ratio'])/df['stdROA'])
df
```

```
/Users/jimmylin/opt/anaconda3/lib/python3.7/site-packages/pandas/core/arraylike.py:3
64: RuntimeWarning: invalid value encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[3]:

	Bank	Date	TAR	Affected_BHC	After_DFA	Profitability	Leverage_ratio	Total_assets
0	1020180	20040930	0.0	0	0	0.002772	0.081957	15.601202
1	1020180	20041231	0.0	0	0	0.003045	0.082480	15.630583
2	1020180	20050331	0.0	0	0	0.002616	0.082074	15.644925
3	1020180	20050630	0.0	0	0	0.002647	0.081712	15.679702
4	1020180	20050930	0.0	0	0	0.002867	0.082944	15.661868
...	...	...	...	...	...	...	...	...
40020	3832583	20131231	0.0	0	1	0.004921	0.231972	13.475152
40021	3832583	20140331	0.0	0	1	0.006362	0.225532	13.525286
40022	3832583	20140630	0.0	0	1	0.006616	0.224154	13.519756
40023	3832583	20140930	0.0	0	1	0.006579	0.226952	13.523643
40024	3832583	20141231	0.0	0	1	0.006423	0.227009	13.552240

39961 rows × 16 columns

In [5]:

```
# establish top 10 BHCs
df_top10 = df.copy()
```

```
df_top10['trading_assets'] = df_top10['TAR'] * np.exp(df_top10['Total_assets'])
df_top10
```

Out[5]:

	Bank	Date	TAR	Affected_BHC	After_DFA	Profitability	Leverage_ratio	Total_assets
0	1020180	20040930	0.0	0	0	0.002772	0.081957	15.601202
1	1020180	20041231	0.0	0	0	0.003045	0.082480	15.630583
2	1020180	20050331	0.0	0	0	0.002616	0.082074	15.644925
3	1020180	20050630	0.0	0	0	0.002647	0.081712	15.679702
4	1020180	20050930	0.0	0	0	0.002867	0.082944	15.661868
...	...	...	...	...	...	...	...	...
40020	3832583	20131231	0.0	0	1	0.004921	0.231972	13.475152
40021	3832583	20140331	0.0	0	1	0.006362	0.225532	13.525286
40022	3832583	20140630	0.0	0	1	0.006616	0.224154	13.519756
40023	3832583	20140930	0.0	0	1	0.006579	0.226952	13.523643
40024	3832583	20141231	0.0	0	1	0.006423	0.227009	13.552240

39961 rows × 17 columns

In [6]:

```
top10 = df_top10[df_top10['Affected_BHC'] == 1].groupby('Bank')['trading_assets'].me
```

In [7]:

```
top10 = top10.sort_values(ascending=False).head(10)
top10.index
```

Out[7]:

```
Int64Index([1039502, 2380443, 1951350, 2162966, 1073757, 2816906, 1042351,
            2914521, 3232316, 3232325],
           dtype='int64', name='Bank')
```

In [8]:

```
dftop10 = df_top10[df_top10['Bank'].isin(top10.index)]
dftop10
```

Out[8]:

	Bank	Date	TAR	Affected_BHC	After_DFA	Profitability	Leverage_ratio	Total_a
1361	1039502	20040930	0.235039	1	0	0.001450	0.077595	20.85
1362	1039502	20041231	0.251247	1	0	0.001451	0.092131	20.86
1363	1039502	20050331	0.254006	1	0	0.001939	0.090340	20.88
1364	1039502	20050630	0.251873	1	0	0.000846	0.089686	20.88
1365	1039502	20050930	0.249962	1	0	0.002129	0.089087	20.90
...	...	...	...	...	...	...	...	
38032	3232325	20040930	0.087236	1	0	0.001762	0.083743	19.56
38033	3232325	20050331	0.102051	1	0	0.003155	0.080864	19.67
38034	3232325	20050630	0.103564	1	0	0.002137	0.079888	19.73
38035	3232325	20050930	0.116463	1	0	0.001675	0.078360	19.79
38036	3232325	20051231	0.118645	1	0	0.001966	0.076220	19.81

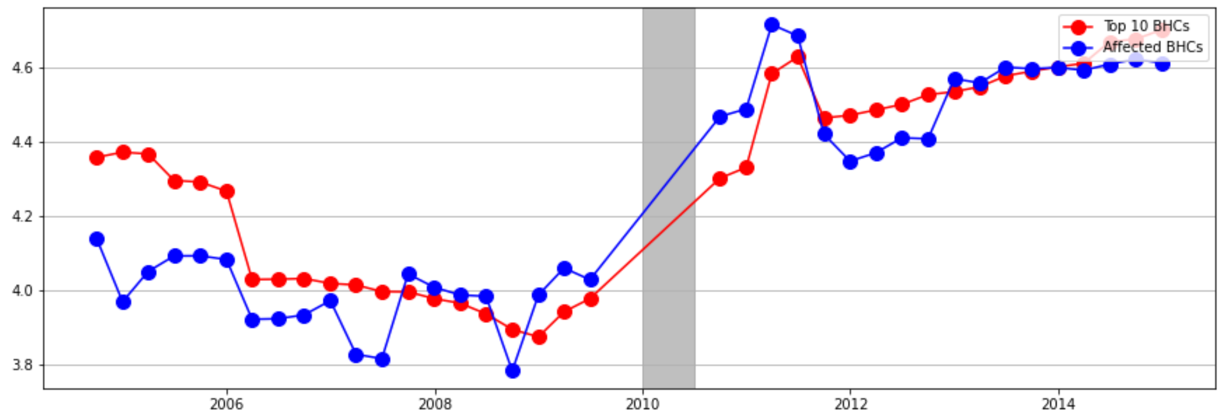
227 rows × 17 columns

In [9]:
```
# mean of the zscore of the top 10 group
top10plot = df_top10.groupby('Date')['zscore'].mean()
```

In [10]:
```
# mean of the zscore of the non top 10 group
df_nottop10 = df[~df['Bank'].isin(top10.index)]
df_nottop10 = df_nottop10[(df_nottop10['Affected_BHC']==1)]

abhc_plot = df_nottop10.groupby('Date')['zscore'].mean()
```

In [11]:
```
# plot graph
# DFA announced 20090930 to 20100630

x_top10 = pd.to_datetime(top10plot.index, format='%Y%m%d', errors='ignore')
x_nottop10 = pd.to_datetime(abhc_plot.index, format='%Y%m%d', errors='ignore')

plt.figure(figsize=(15,5))
plt.grid(axis = 'y')
plt.axvspan(pd.to_datetime('20091231', format='%Y%m%d'), pd.to_datetime('20100630',
plt.plot(x_top10,top10plot.values,'o-', color='red', markersize=10, label='Top 10 BH
plt.plot(x_nottop10,abhc_plot.values,'o-', color='blue', markersize=10, label='Affec
plt.legend(loc="upper right")
```

Out[11]:
```
<matplotlib.legend.Legend at 0x7fd22fe7ba90>
```



In [ ]:

In [ ]:

In [ ]:

In [9]:
```
data = pd.read_csv('DiD_data.csv')
df_1 = pd.DataFrame(data)

# rename columns
df_1.columns = ['Bank', 'Date', 'TAR', 'Affected_BHC', 'After_DFA', 'Profitability',
# add variable Affect
df_1['Affect'] = (df_1['TAR'].mask(~df_1['After_DFA'].eq(0)).groupby(df_1['Bank']).t
```

In [10]:
```
# for each bank, first calculate standard deviation of profitability based on 10 qua
# first split the original dataframe by different period

first_10 = [20040930, 20041231, 20050331, 20050630, 20050930, 20051231, 20060331, 20
second_10 = [20070331, 20070630, 20070930, 20071231, 20080331, 20080630, 20080930, 2
thrid_10 = [20100930, 20101231, 20110331, 20110630, 20110930, 20111231, 20120331, 20
forth_10 = [20130331, 20130630, 20130930, 20131231, 20140331, 20140630, 20140930, 20

df_zscore_1 = df_1.copy()
df_zscore_2 = df_1.copy()
df_zscore_3 = df_1.copy()
df_zscore_4 = df_1.copy()

df_zscore_1 = df_zscore_1[df_zscore_1['Date'].isin(first_10)]
df_zscore_2 = df_zscore_2[df_zscore_2['Date'].isin(second_10)]
df_zscore_3 = df_zscore_3[df_zscore_3['Date'].isin(thrid_10)]
df_zscore_4 = df_zscore_4[df_zscore_4['Date'].isin(forth_10)]
```

In [11]:
```
# group by bank_id to calculate the standard deviation of profitability
# and append the standard deviation back to the dataframe

def calculate_Profitability_std(dataframe):
    agg = dataframe.groupby(['Bank']).agg(np.std)['Profitability']
    dataframe['Profitability_std'] = 0

    for i, j in enumerate(dataframe['Bank']):
        for k, l in enumerate(agg.index):
            if j == l:
                dataframe.iloc[i,15] = agg.values[k]

    return dataframe
```

In [12]:
```
df_zscore_1 = calculate_Profitability_std(df_zscore_1)
df_zscore_2 = calculate_Profitability_std(df_zscore_2)
df_zscore_3 = calculate_Profitability_std(df_zscore_3)
df_zscore_4 = calculate_Profitability_std(df_zscore_4)
```

In [13]:
```
# merge dataframes into original one and calculate the z_score

df_zscore = pd.concat([df_zscore_1, df_zscore_2, df_zscore_3, df_zscore_4])
df_zscore['Profitability_std'] = df_zscore['Profitability_std'].fillna(0)

# drop banks with stdROA == 0
df_zscore = df_zscore[df_zscore['Profitability_std']!=0]
df_zscore = df_zscore.dropna()
df_zscore['z_score'] = (df_zscore['Profitability'] + df_zscore['Leverage_ratio']) /

# transform z_score using ln

df_zscore['ln_z_score'] = np.log(df_zscore['z_score'])
df_zscore
```

```
/Users/jimmylin/opt/anaconda3/lib/python3.7/site-packages/pandas/core/arraylike.py:3
64: RuntimeWarning: invalid value encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[13]:

	Bank	Date	TAR	Affected_BHC	After_DFA	Profitability	Leverage_ratio	Total_assets
0	1020180	20040930	0.0	0	0	0.002772	0.081957	15.601202
1	1020180	20041231	0.0	0	0	0.003045	0.082480	15.630583

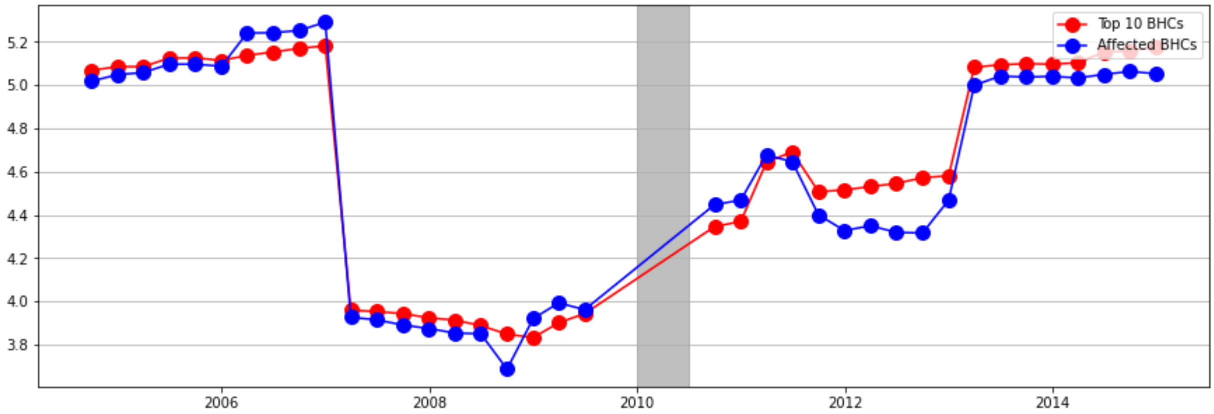
	Bank	Date	TAR	Affected_BHC	After_DFA	Profitability	Leverage_ratio	Total_assets
2	1020180	20050331	0.0	0	0	0.002616	0.082074	15.644925
3	1020180	20050630	0.0	0	0	0.002647	0.081712	15.679702
4	1020180	20050930	0.0	0	0	0.002867	0.082944	15.661868
...	...	...	...	...	...	...	...	...
40020	3832583	20131231	0.0	0	1	0.004921	0.231972	13.475152
40021	3832583	20140331	0.0	0	1	0.006362	0.225532	13.525286
40022	3832583	20140630	0.0	0	1	0.006616	0.224154	13.519756
40023	3832583	20140930	0.0	0	1	0.006579	0.226952	13.523643
40024	3832583	20141231	0.0	0	1	0.006423	0.227009	13.552240

39927 rows × 18 columns

In [17]:

```
# establish top 10 BHCs
df_top10 = df_zscore.copy()
df_top10['trading_assets'] = df_top10['TAR'] * np.exp(df_top10['Total_assets'])
top10 = df_top10[df_top10['Affected_BHC'] == 1].groupby('Bank')['trading_assets'].me
top10 = top10.sort_values(ascending=False).head(10)
dftop10 = df_top10[df_top10['Bank'].isin(top10.index)]

# mean of the zscore of the top 10 group
top10plot = df_top10.groupby('Date')['ln_z_score'].mean()

# mean of the zscore of the non top 10 group
df_nottop10 = df_zscore[~df_zscore['Bank'].isin(top10.index)]
df_nottop10 = df_nottop10[(df_nottop10['Affected_BHC']==1)]

abhc_plot = df_nottop10.groupby('Date')['ln_z_score'].mean()

# plot graph
# DFA announced 20090930 to 20100630

x_top10 = pd.to_datetime(top10plot.index, format='%Y%m%d', errors='ignore')
x_nottop10 = pd.to_datetime(abhc_plot.index, format='%Y%m%d', errors='ignore')

plt.figure(figsize=(15,5))
plt.grid(axis = 'y')
plt.axvspan(pd.to_datetime('20091231', format='%Y%m%d'), pd.to_datetime('20100630',
plt.plot(x_top10,top10plot.values,'o-', color='red', markersize=10, label='Top 10 BH
plt.plot(x_nottop10,abhc_plot.values,'o-', color='blue', markersize=10, label='Affec
plt.legend(loc="upper right")
```

Out[17]: <matplotlib.legend.Legend at 0x7fc1e904e050>


Top 10 BHCs
Affected BHCs
5.2
5.0
4.8
4.6
4.4
4.2
4.0
3.8
2006
2008
2010
2012
2014

In [ ]: