Handelshøyskolen BI

GRA 19703 Master Thesis

Thesis Master of Science 100% - W

Predefinert informasjon

Startdato:	09-01-2023 09:00 CET	**Termin:**	202310
Sluttdato:	03-07-2023 12:00 CEST	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	T		
Flowkode:	202310||11184||IN00||W||T		
Intern sensor:	(Anonymisert)		

Deltaker

Navn:	Issam Hafy og Mnohar Sheikh Ali

Informasjon fra deltaker

Tittel *:	Analyzing Returns and Predicting IPO Probability
Navn på veileder *:	Alfonso Irarrazabal

Inneholder besvarelsen konfidensielt materiale?:	Nei	**Kan besvarelsen offentliggjøres?:**	Ja

Gruppe

Gruppenavn:	(Anonymisert)
Gruppenummer:	308
Andre medlemmer i gruppen:	

BI Norwegian Business School
Oslo, Spring 2023

Analyzing Returns and Predicting IPO Probability

A Comparative Analysis of Private and Public Firms in Norway

Issam Hafy, Mnohar Sheikh Ali

Supervisor: Alfonso A. Irarrazabal

Master thesis, MSc in Business, major: Economics

BI Norwegian Business School

Acknowledgements

First, we want to show gratitude towards our supervisor, Alfonso A. Irarrazabal, who has guided, motivated, and helped us immensely when working on this thesis. We sincerely appreciate him for always being available - his flexibility, support, and words of encouragement have allowed us to complete this thesis.

We also extend our gratitude towards our families and friends, who continued to have supported, challenge, and motivated us when working on this thesis.

Lastly, we also want to thank the economics faculty at BI Norwegian Business School, who helped us find an exciting subject to write our dissertation through their engaging, active, and interesting lectures.

BI Norwegian Business School

Oslo, July 2023

[illegible] [illegible]

Issam Hafy Mnohar Sheikh Ali

Abstract

Analyzing returns is a practical approach to gaining insights into the growing wealth disparities. In our study, we utilize data from Norwegian firms spanning the period between 2007 and 2019. Our objective is to investigate the distinctive returns attributes for private and public firms, explore the influence of various financial and productivity measures on these returns, and examine whether returns can serve as a predictive indicator for the likelihood of a firm becoming publically traded. We show that private and public firms' returns are dispersed, and persistent, and the presence of an imperfect pass-through of input choices to changes in output. Furthermore, we present evidence that firm size and productivity positively affect returns. Lastly, we found that returns, the firm's growth, and the firms' output share of assets can help us predict the probability of a firm becoming publicly traded.

***Keywords* –** Returns, Private and Public Firms, Financial and productivity measures, Likelihood of IPO, Dispersion, Imperfect Pass-through

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction

Wealth inequalities are becoming more prominent in developed economies. In addition, we are witnessing a higher concentration of wealth among private business owners Boar et al. (2023). According to the International Monetary Fund Stanley (2022) 2020's world inequality report, 1 % of the world's population owns 38 % of the total wealth, and 10 % of the world's population owns 76%. Recent theoretical research has highlighted the importance of the returns on savings (Benhabib et al. (2011); Benhabib et al. (2017)). Furthermore, many empirical studies provided evidence of significant and persistent differences in the return on households' net worth increases in net worth Bach et al. (2020). The same paper also revealed that private equity makes up a more significant portion of net worth as it increases, highlighting the importance of entrepreneurship in understanding wealth disparities. Bach et al. (2020), alongside Fagereng et al. (2018), Smith et al. (2019), and Boar et al. (2023), all argue that this dispersion and persistence are essential drivers for wealth inequality. It is essential to understand these dynamics in the economy in order to tackle issues related to social mobility, firm- and household dynamics, and asset pricing concerning the risk premia. Additionally, this sparks discussion regarding wealth and earnings taxation policies. What is the sustainable and efficient way to reallocate the resources in the society?

Furthermore, it is vital to realize that wealth and returns are dispersed and persistent, and it is equally important to understand what drives the returns of the companies. Are the returns to private firms (equity) equal to the public, as Moskowitz and Vissing-Jørgensen (2002) suggests? If so, what is the implication of the equally profitable companies but with higher risk tolerance? If they are higher - is that solely because the entrepreneurs have a higher tolerance for risk? Could they possibly be more efficient with their input choices, and do they possibly have managerial talents contributing to higher returns? If they are lower, why would anyone want to be an entrepreneur? Traditionally, these aspects have been challenging to capture due to the lack of suitable and well-measured datasets, which also contain the same information over a more extended period in addition to detailed financial and accounting measures

Bach et al. (2020).

Our goal of this thesis is threefold: First, we want to investigate the returns of Norwegian firms, both private and public. Specifically, we want to look at whether the dispersion reported from other countries is similar to Norway and how persistent the returns are. Furthermore, we want to examine how the profit shares correlate with changes in the growth rate of the company's input variables. Secondly, we want to understand what drives the returns and inspect how the productivity levels vary and affect the returns across company types. Lastly, we want to examine how the public and private firms differ and what our findings say about the probability of a company becoming public. To do so, we utilize comprehensive, detailed, and rich data from Statistics Norway. We construct a panel dataset, with the firm entities as the individual groups, and look at their dynamics over the years. Our analysis considers Norwegian businesses from 2007 – 2019. Constructing panel data has many advantages: We can trust the information, as the data is collected through firms' own reportings, which decreases the measurement error significantly. Additionally, the data from SSB is rich in that it has financial and accounting information about numerous companies. Lastly, since the data range over a long period, we can study the persistence of the returns and examine the different factors affecting it. Finally, as there are limitations to what this thesis can cover, we aim to highlight attractive areas for further research.

The rest of the paper proceeds as follows. In section 2, we provide an overview of related works, highlighting some of the contributions that we found highly inspiring. In section 3, we present the data used and how we cleaned and formatted it for analysis. Section 4 provides an empirical analysis. Specifically, we show how each company type's return is dispersed and persistent. Moreover, we consider the imperfect pass-through of input choices (labor and capital) to changes in the firms' output. In section 5, we examine the determinants of profitability and their relationship with the firms' returns. We run several regressions to estimate the impact of these measures and productivity measures on the returns. In section 6, we shed light on our findings about the probability of a company becoming publicly listed. Section 7 concludes our findings and suggests further areas for research. Additionally, due to space limitations, we include supplementary material in the appendix, to which the interested reader is referred. Moreover, we include the codes (for Python and Stata) used to conduct our analysis.

2 Related Work

Our thesis uses micro-data on Norwegian businesses and studies how the returns are and what determines the firms' profitability. It also examines which factors can help predict whether a company gets listed on the Norwegian Stock Exchange.

For the first part, our thesis is, therefore, strongly related to Boar et al. (2023), who use balance sheets and income statements of private firms to study the dispersion of returns to business wealth. They focus their analysis on Spanish firms but have documented similar results for other European countries. Their findings include showing dispersion and persistence in returns and imperfect pass-through of labor and capital choices to changes in output, which results in significant changes in the firms' profit shares. Bach et al. (2020) investigates returns to wealth on an administrative panel with wealth tax returns of Swedish residents. Their work showed that the expected return on household wealth is strongly persistent and increases with net worth. They argue that systematic risk determines persistence and show how idiosyncratic risk is transitory but generates extended returns dispersion for the wealthier brackets. Furthermore, they show that wealthier population brackets allocate a larger share of their investments in risky portfolios and load more aggressively on risk factors, including market, size, and value factors (Bach et al., 2020, p.3). Their result highlights a significant role of the risk-takers in the top percentile as an explanation for the higher average returns in the households. Our thesis is also related to Fagereng et al. (2018), who tries to explain the right skewness of wealth distribution. Their contribution highlights the dispersion of wealth between different brackets of net worth. Similar to Bach et al. (2020), they find that exposure to risk plays a significant role in explaining the heterogeneity in returns. Moreover, they show the persistence of wealth returns and attribute that to three main explanations: the continuation of risk exposure through risk tolerance and compensation through a return premium. Secondly, they refer to Piketty (2014)'s findings about persistent wealth differences and the scale of wealth returns. Thirdly, they consider the fixed effects that capture the financially sophisticated individual traits, which affect their investment choices.

Our work has relied on several key contributions in the literature for the profitability analysis and hereunder productivity. At the forefront is Capon et al. (1990)'s meta-analysis of 320 studies that look into factors that affect financial performance. The authors summarize the factors' signed relationships and their significance. Goddard et al. (2005) investigate determinants of profitability in selected sectors in Europe and find, among other things, a negative relationship between debt-to-equity to profits and a positive relationship between the liquidity ratio and profits. Bartelsman and Doms (2000) set out to understand the factors that fuel productivity using panel data. They argue that research papers mainly establish the correlations between variables and productivity growth; they also go a step further by trying to identify the causality.

The last part of our thesis relies on the work of Pagano et al. (2022), who performed an empirical analysis of firms' decisions to go public. They develop an estimation model that uses firm size, CAPEX, growth, returns, leverage, and the cost of bank credit to predict the probability of an IPO. Jones and Hensher (2004) propose an extensive framework for using probabilistic modeling to predict corporate financial distress and techniques to test the model's stability.

Other supportive literature includes; Gopniath et al. (2017), who set out to analyze the capital allocation and productivity in South Europe, Sikveland et al. (2021), who analyze the firms (private and public) in the Norwegian salmon farming industry, Benhabib et al. (2019) who considers the wealth distribution in the United States, and Donangelo et al. (2019) who explores the role of labor share to the firms' variation in expected returns.

3 Data

3.1 Data Collection and Sources

Our primary data source is microdata retrieved from SSB (Statistics Norway). Access to this data is through a more extensive project conducted by the Department of Social Economics at BI Norwegian Business School. We focus our analysis on Norwegian businesses, and the data covers a wide range of Norwegian businesses concerning sector, size, and profitability, among many other things. For this thesis, we use three datasets described in more detail below.

3.2 The Accounting Data

First, we use balance sheets and income statements for non-financial Norwegian businesses, including limited liability and public limited companies. Before we clean the data, the dataset contains accounting statements for 471 113 companies between 2005 - 2019, and almost 3 400 000 firm-year observations. The dataset uses reported and recorded annual accounts SSB (2020). We follow the approach of Gopniath et al. (2017) when cleaning the data.

3.2.1 Basic Cleaning

We begin by dropping any rows that needed more information about the organization number and year of the reporting and any duplicates. Then, we check whether any firm-year observations had missing or negative information about total assets (including tangible, intangible fixed, and current assets), current and long-term liabilities, including loans and accounts payable, employment compensations, cost of goods sold, revenue, and wealth. Where that was the case, we dropped the observations.

3.2.2 Checking for Internal Inconsistencies

Next, we examine the internal consistency of the reported information. We sum up the variables and investigate whether they add to their respective aggregates. The following sums were investigated:

- Fixed Assets = Tangible Fixed Assets + Capital Assets + Intangible Assets
- Current Assets = Inventories + Claims + Securities and other Financial Current Assets + Cash and Bank Deposits
- Total Assets = Total Fixed Assets + Total Current Assets
- Long-Term Liabilities = Convertible Loans and Bond Loans + Long-Term Debt to Credit Institutions + Debt to Businesses in the Same Enterprise + Other Long-Term Liabilities

- Current Liabilities = Convertible Loans and Bond Loans (Current) + Overdraft + Trade Creditors + Payable Tax + Proposed Dividends + Advance From Customers + Debt to Companies in the Same Group + Salary Due + Other Short-Term Liabilities
- Equity = Deposit Capital + Retained Earnings
- Total Equity and Liabilities = Equity + Liabilities

We dropped the observation where we encountered some inconsistencies. These steps reduced our sample by 47%. The resulting dataset corresponds to more than 1 480 000 firm-year observations constituting more than 253 000 unique entities.

3.3 The Shareholder Registry

We use the shareholder registry provided, which contains information regarding share capital, organization form, establishment date, and whether the company is listed. The dataset ranges from 2005 to 2019 and contains information about just over 500 000 companies. Similarly to the accounting data, we drop duplicates and missing firm identifiers. We are interested in the variables indicating which company is listed on the stock exchange, organizational form, and establishment date, in addition to the organization number and year. We classify the companies using the stock exchange indicator and organizational form. We have three classifications of the firms in the sample:

- **Public Listed Companies**: the companies we have records of being listed on the Oslo Stock Exchange, or on Euronext Growth.
- **Public Non-listed Companies**: are the companies that are public limited companies but not necessarily listed on the stock exchange.
- **Private Companies**: the rest of the companies that do not fit the aforementioned categories.

Additionally, we use the establishment date reported to calculate the company's age.

The variable indicating whether it is a publicly listed company had its first recorded observation in 2007 - we, therefore, drop the observations before that year. The resulting dataset contains more than 3 300 000 firm-year observations, constituting almost 500 000 companies. This dataset has information about more than 482 000 private firms, 315 public non-listed companies, and 406 publicly listed firms.

3.4 The Central Register of Establishments and Enterprises

The registry contains information about various characteristics of entities, including organization number, organization address, activity, sector codes, and dates of bankruptcies, mergers, and acquisitions SSB (nd). The data ranges from 1996 up til 2021 and has information about just under 2.4 million different organizations. Again, similarly to the previous datasets, we drop observations with missing information about the organization number. Furthermore, we extract information about the firm-industry and sector. This is the only information of interest related to our thesis, so we dropped all other observations.

3.5 Merging the Data

What is common for these datasets is that they contain the unique organization number for each entity and year, which we use as a firm identifier to combine these datasets before performing a comprehensive analysis of the characteristics of Norwegian businesses and developing appropriate models for the analyses.

Since the variable indicating whether it is a publicly listed company had its first recorded observation in 2007, we cut the accounting data so it starts from the same year. The merged dataset contains almost 1.4 million firm-year observations for more than 241 000 unique firms. However, we constrict the sample for the profitability analysis such that we only include firms we have data for at least ten years. This is particularly important when assessing the persistence of the returns. The restricted sample has more than 730 000 firm-year observations for more than 60 000 unique companies. For the sake of simplicity, we will refer to the merged sample as the "**full sample**" and the restricted sample as the "**balanced sample**".

The full sample consists of almost 225 000 private firms and 171 (239) public non-listed (listed) companies. The balanced sample consists of almost 62 000 private firms, 88 public non-listed firms, and 154 publicly listed firms.

4 Empirical Analysis

4.1 Constructing the Variables

Our measure of net profit, π_{it}, is the net profit reported in each firm's income statement, which is net of all costs, expenses, and depreciation. Unlike Boar et al. (2023), we define the output, y_{it}, as the net profit added to the labor costs, interest expense, and depreciation. Our measure of labor, l_{it}, is the total of the firm's total employment compensation, including benefits and employers' taxes. We define equity, e_{it}, as total assets less total liabilities. Our measure of capital, k_{it}, also differs from that of Boar et al. (2023); we define it as the sum of tangible and intangible fixed assets. Additionally, we construct the firm's labor share as its total labor cost over its output, and the profit share as the firm's net profit over its output. Our return measure differs from that of Boar et al. (2023). We calculate the return on assets by dividing its earnings before interest, taxes, depreciation and amortization (EBITDA) by its total assets. This is to stay consistent with the analyses in the latter sections. Moreover, we construct the growth rate of output, labor, and capital as the difference of logged value at time t and t_{-1}. When examining the data, we noticed a high level of kurtosis in the distribution of the variables used in the analysis. Therefore, we winsorize the variables at different levels to eradicate potential errors in the analyses due to outliers. The rationale behind choosing winsorizing, rather than slicing, was to keep as much data as possible in our sample. Winsorizing is the act of reassigning values above or below a quantile x to the xth percentile Dash et al. (2023). For the variables explained above, we reassign the values of the outliers to the 1st and 99th percentile.

4.1.1 Summary Statistics

Table 4.1 shows the baseline descriptive statistics for the Balanced sample, with the variables winsorized. Public companies entail both listed and non-listed. In the analysis further, we consider these as public companies. However, we will expand our analysis when there are significant differences between these two categories. Additionally, the dispersion of the **balanced sample** regarding the variables shown is coincidental with the **full sample**.

Table 4.1: Summary Statistics

	mean	st.d	p10	p25	p50	p75	p90
Private							
Output	115,90	312,01	5,37	12,95	31,05	79,10	219,26
Labor	76,91	191,55	2,70	8,43	22,29	57,39	155,04
Capital	72,47	266,96	0,37	1,50	6,24	28,84	114,28
Equity	129,16	458,74	2,44	6,21	17,91	56,52	197,93
Net Profit	19,79	72,60	- 2,24	0,25	3,25	12,02	38,54
Public							
Output	7 327,32	30 140,33	- 469,46	27,32	658,37	3 529,64	12 230,73
Labor	1 022,14	2 486,33	36,34	94,74	258,70	726,29	2 006,08
Capital	2 646,95	8 818,31	11,20	68,91	274,29	1 032,45	3 813,55
Equity	36 210,73	125 743,57	568,17	1 541,39	4 863,84	16 510,74	60 133,28
Net Profit	4 122,04	20 714,69	- 1 611,36	- 276,66	79,98	1 325,50	6 741,51

Note: These statistics is using the **Balanced sample**
Numbers are expressed in 100 000NOK.

The average (median) private Norwegian firm produces 11M (3.1M) NOK in output, as opposed to the public firm that produces 732M and 65M NOK worth of output, in average and median, respectively. In labor costs, the average (median) firm spends almost 7.6M (2.2M) NOK, and the average (median) public firm spends 102M (25M) NOK. The private firm owns, on average (median) 7.2M (0.6M) NOK in the capital, whereas the average (median) public firm owns 265M (27M) NOK. The average (median) private firm holds 12.9M (1.7M) NOK in total equity, as opposed to the public firm that holds 3.6n (486M) NOK. The private firm earns 1.9M (0.32M) NOK on average (median) in net profit, whereas the public firm earns 412M (7.9M) NOK on average (median). Evidently, from the high level of variation in the sample, there are some large companies in both categories. The volatility drives the dispersion of these variables, signaled by the high level of standard deviations for both type of firms on all variables.

4.2 Rate of Returns

We start our analysis by investigating the relationship between the return on assets and the firm's total assets. We plot the returns against the firms' total assets. We use bins to sort the firms by size and plot the mean of returns for each bin. Figure 4.1 shows the plot. A clearly negative relationship exists between the assets and returns for private firms. The bigger the firm gets, the lower the returns are, on average.

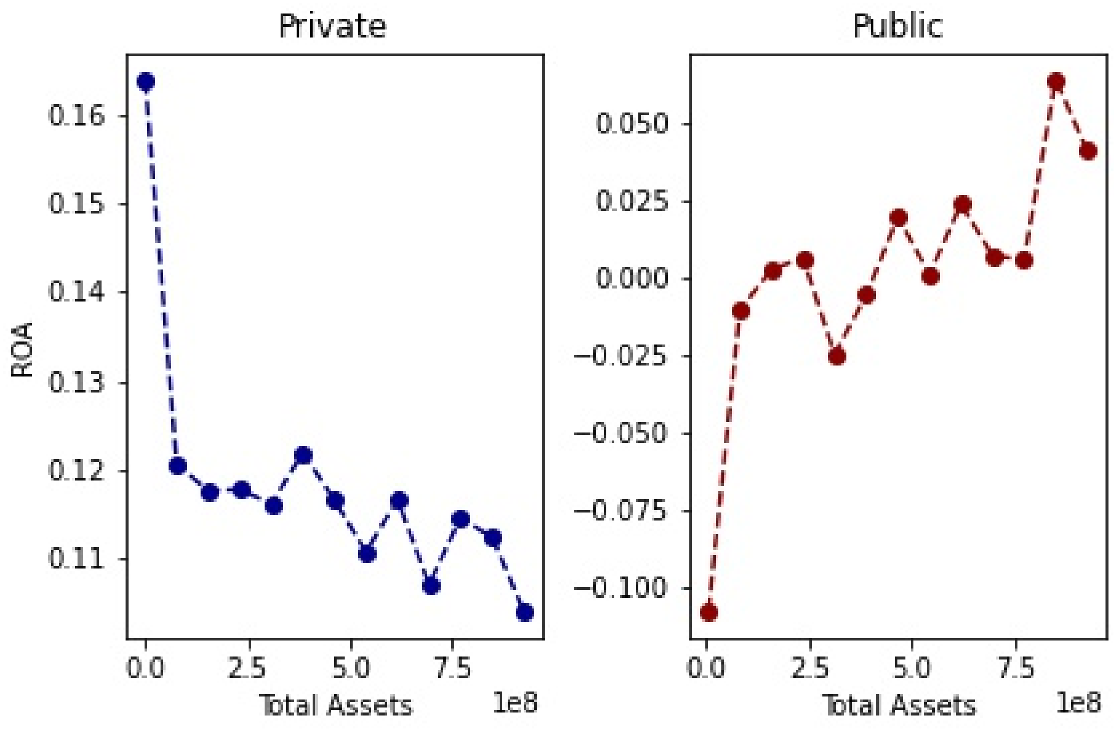

Figure 4.1: Relationship between Returns and Assets
Note: The assets and ROA are winsorized at the 1st and 99th percentile.

Interestingly, the relationship is the opposite for public firms, where they experience an increasing trend of returns with the assets. Notably, public firms have far more negative returns. As noted by Boar et al. (2023)[1], this might seem contradictory to Fagereng et al. (2018)'s and Bach et al. (2020)'s findings about how wealthier households enjoy higher returns. However, our findings (and Boar et al. (2023)) consider business wealth, which differs from the assets these authors study.

Next, we consider the dispersion of the returns and report them in table 4.2.

Table 4.2: Dispersion in Average Returns

	mean	std	p10	p25	p50	p75	p90	p95
Private								
ROA	0.16	0.18	-0.02	0.06	0.14	0.26	0.39	0.49
$\overline{ROA}$	0.16	0.13	0.03	0.08	0.14	0.22	0.31	0.39
Public								
ROA	-0.001	0.26	-0.24	-0.03	0.04	0.11	0.22	0.29
$\overline{ROA}$	-0.01	0.20	-0.25	-0.06	0.04	0.09	0.15	0.23

These results corroborate with the results of Boar et al. (2023), Fagereng et al.

[1]They discuss this in the first edition of their research, published in December 2021.

(2018) and Bach et al. (2020), who also showed the dispersion and persistence of returns to business wealth. Differently from their findings, we also consider the public companies in our sample. As highlighted by (Boar et al., 2023, p.1), the accounting returns include the effects of a fixed factor that can be attributed to executive or entrepreneurial talent or market power. This realization contributes to our understanding of what drives the returns, as well as its persistence.

The first row of Table 4.2 shows the transverse distribution of returns. The mean return for private and public firms is 0.16 and -0.001, respectively. The table highlights the returns' dispersion in addition to the volatility of returns, through the variation levels reflected by the standard deviation of .18 (.26) for private (public) firms. The returns for public firms range from -24% at the 10th percentile to 22% at the 95th. On the other hand, private firms' returns range from -0.02% at the 10th percentile to 49% at the 95th percentile. These results are slightly less dispersed, compared to what Boar et al. (2023) report for Norway. This can be attributed to the fact that we use different datasets and the differences in our definitions of the net profit, in addition to calculating the returns to equity and we assets. Additionally, they exclude several sectors in their analysis, whereas we have included them. Regardless of the level of returns, our results tell the same story of dispersion in the returns. The second row reports the distribution for each firm and their time-series average. Given that we only include firms we have data for at least ten years, these results indicate that some firms unfailingly earn high returns over time. Private firms' returns range from 3% to 39% at the 10th and 95th. Public firms' returns range from -25% to 23%.

To further highlight the persistence of returns to business wealth, we perform the following regression: $ROA_{it} = ROA_{it-1}$. The resulting coefficients are presented in Table 4.3. The results highlight the persistence of the returns. The returns are significantly more persistent for private firms, compared to public. When we split the public firms into listed and non-listed, the coefficients are 0.485 and 0.256 respectively.

Table 4.3: Persistency of Returns by Company Type

	Private	Public
ROA_{it}	0.5199	0.3969

Note: Both estimates are significant at 0.01%.

Another question is how much of the dispersion is due to financial frictions and how much is attributable to risk premia. Moll (2014) raises the concern of dislocating capital, shown as low total factor productivity, and argues that financial frictions can explain differences in per capita income (Moll, 2014, p.3187). Boar et al. (2023) construct a model showing that returns are still considerably dispersed despite financial friction. Moskowitz and Vissing-Jørgensen (2002) found that, on average, 70% invest in a single private company in which they have an active management interest in the United States. They propose the "Private Equity Premium Puzzle," which is that returns to private equity are equal to that of the public, despite private equity being a riskier investment. Unfortunately, our analysis does not include who invests in these private firms and whether or not the investors have an active managerial interest in these firms. However, these results show that returns to private companies are much higher, on average than the public's. Moreover, we found that publicly listed companies have less average returns than non-listed companies (-0.08 vs. -0.06), and the listed companies are far more volatile. The distribution table of returns for publicly listed and non-listed companies is in the appendix. Moskowitz and Vissing-Jørgensen (2002) and Boar et al. (2023) highlight that private businesses are poorly diversified in their income sources. Hence, these differences reflect a risk premium. Additionally, examining the marginal returns, as opposed to the average, one could further examine the role of risk profile and financial constraints. However, that is outside the scope of this thesis. (Bach et al., 2020, p.29) discuss the vital implication this has for the policies regarding private wealth - what should be the tradeoff between the compensation for taking risks and wealth inequality?

4.3 Output Growth Rates and Their Effect on Labor and Capital Choices

Table 4.4: Output Growth Rates

	St.d	P0.1	P0.5	P25	P75	P90	P99.5	P99.9
Private Companies	0.52	-1.70	-0.59	-0.10	0.17	0.39	2.22	3.65
Public Companies	0.81	-2.97	-1.15	-0.08	0.23	0.81	2.72	3.53
Gaussian	0.52	-1.60	-1.33	-0.35	0.35	0.66	1.33	1.59

Note: Private companies based on 566 298 firm-year observations. Public Companies based on 870 firm-year observations.

Following Boar et al. (2023)'s framework, we next document the dispersion of output growth rates. The growth rates are highly dispersed - regardless of whether the company is private or public. We see how dispersed the output growth is when we compare the distribution to the Gaussian distribution with the same standard deviation as the Balanced sample. Similar to Boar et al. (2023) results, we also document the extensive and fat-tailed changes in output. A slight increase from one percentile to the other (99.5th to 99.9th) increases the growth rate from 2.22 to 3.65 for private firms. Interestingly, public firms experience a larger dispersion in growth rates considering the standard deviation. The fat-tailed changes become even more visible when comparing the growth rates to the Gaussian distribution.

Furthermore, we investigate how labor and capital choices are affected by the changes in the output growth rate. We normalize the logarithm of output such that it centers around zero. Assuming these factors are chosen before one can measure the output (meaning that the labor- and capital choices chosen beforehand are risky), their response to output changes is slow. Figure 4.2 shows how the change in output for a particular firm from the private companies' sample affects the labor and capital shares. The aim is to illustrate the imperfect pass-through of these input variables.

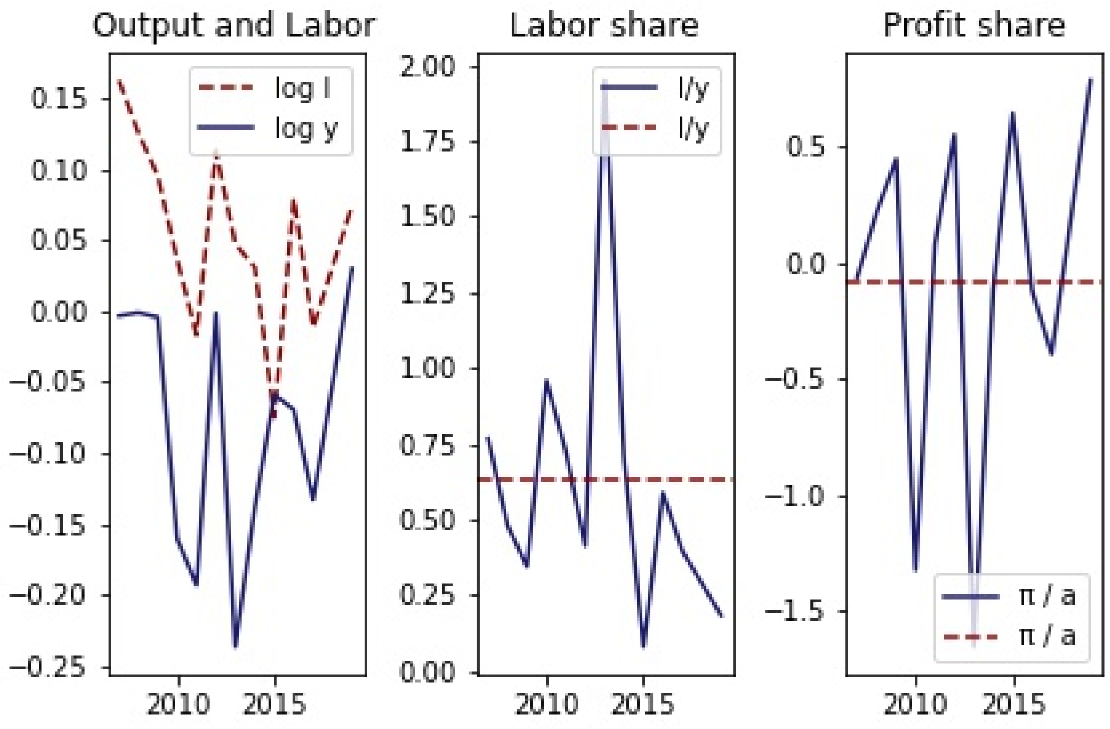

Figure 4.2: Example of a Firm

This particular firm experiences high volatility in its output and labor employment,

Table 4.5: Comovement Between Capital, Labor, Profits and Output

		$\Delta\, log\ l$	$\Delta\, log\ k$	$\Delta\, \pi/y$	$\Delta\, \widehat{\pi}/y$
			Panel I		
Private Companies	$\Delta\, log\ y$	.312***	.0277***	5.638*	.444***
Public Companies	$\Delta\, log\ y$	.0298	.0206	.847***	.388***
All Companies	$\Delta\, log\ y$	.311***	.0281***	5.596*	.444***
			Panel II: | log y| <0.5		
Private Companies	$\Delta\, log\ y$	.469***	.0908***	.481***	.474***
Public Companies	$\Delta\, log\ y$	.0775	-.302*	.475***	.475***
All Companies	$\Delta\, log\ y$	.469***	.0906***	.481***	.474***

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Note: Panel I (II) is using 494 993 (437 145) observations. Standard errors are clustered at the firm-level, and no year fixed effects are included. Including these had a neglible effect on the overall results.

as shown in the left panel of the figure. The middle and the right panel show the labor and profit share of the company alongside its time-series mean. We observe how the shares have opposite effects - when the labor share increases, the net profit falls. Since labor and capital react imperfectly to changes in output, a sharp decline in output leads to a sharp increase in labor share and, thereby, a decrease in profit share. This dynamic applies to all private, public non-listed, and listed companies.

We study the comovements between labor, capital, and profit shares to output by regressing the growth rate output to the growth rates of labor and capital, in addition to the profit shares. Table 4.5 shows the resulting coefficients and their significance level. The first column shows how the growth rate of labor reacts to a change in the output growth rate — the second column report likewise for the growth rate of capital. For private companies, a 10% decline in output is connected to a 3.1% drop in the private firms' labor costs and 027% in its capital stock. These results differ from what Boar et al. (2023) report. They report a much higher decline in capital stock, equal to 1.5%. The difference is due to the way we have defined capital. However, interestingly, the same coefficients for public (listed and non-listed) companies are not significant. (Cooper and Haltiwanger, 2006, p.1) explains that the investments made by a company are subject to significant adjustment costs, which explain the low response rate of capital, both in our results and that of Boar et al. (2023)'s.

We also investigate how the firms' profit share co-moves with the output growth rates reported in the third column. The high estimate highlights the correlation between changes in output and profit shares. For public companies, the resulting drop in the profit share is 8.47% when facing an output decline of 10%. When we split the public companies into listed and non-listed, the declines are 8.15% and 4.7%, respectively. The reaction of the profit share for private firms is extensive, considering that the average non-winsorized profit share for that category is 0.48. However, the estimate is only significant at 5%. We, therefore, consider this as highlighting the correlation between the profit share and output changes, more than the size itself. On the contrary, public companies experience a fall equal to 8.4% (4.7% for nonlisted and 8.1% for listed), all significant at the 0.1% level. Furthermore, to highlight how much the labor share affects the comovement between the profit share and output growth, we construct a counterfactual profit share proposed by Boar et al. (2023):

1. We calculate the time-series average labor share for each firm in our sample
2. We assume the firm's labor cost equals the mean calculated times the output produced
3. We calculate the net profit with the new labor cost

Next, we then conduct the same regression with the counterfactual profit share, which we report in the last column. Unsurprisingly, this constructed profit share comoves considerably less with the output growth rates. The results are 4.4% and 3.8% for private and public firms, respectively. Both results are significant at the 0.1% significance level. Further, we can split the sample into the listed and nonlisted companies. For listed companies, the counterfactual profit share decrease 5.7% following a decline in output of 10%. The same change in output leads to a decline equal to 1.9% for non-listed companies. Both estimates are significant at 0.1% level. The fact remains that the volatility of labor share plays a significant role between profit share and output growth rate.

Further, to address measurement errors in the output, we analyze the same data but limit the sample to include only firms with an absolute value of 0.5 in output growth rate. Despite the increases in the coefficients, the storyline remains the same. There is still evidence of imperfect pass-through of the input choices, with 0.469 and 0.0908 for labor and capital for private firms. What is different for public companies is that the capital growth rates are significant at a 5% level, which was insignificant before the restriction. Additionally, the coefficients for the profit shares are considerably lesser than before restricting the sample. Once again, most of the change in profit can be attributed to the change in the firms' labor share.

Our sample's average labor share for private (public) firms is 0.23 (0.27). Therefore, as anticipated, the variation in profit share is attributed to changes in labor shares. (Boar et al., 2023, p.11) highlight the importance of including these frictions (imperfect comovements) in models that analyze firm dynamics to reproduce the same volatility in profits observed in the data.

4.4 Labor and Profit Share Deviations

Table 4.6: Deviation of Labor and Profit Shares from Time Series Mean

	P01	P10	P25	P50	P75	P90	P99
				Labor Deviations			
Balanced Sample	-0.79	-0.17	-0.07	-0.00	0.07	0.19	1.67
Private Companies	-0.78	-0.17	-0.07	-0.00	0.07	0.19	1.65
Public Companies	-7.78	-0.29	-0.09	-0.00	0.13	0.50	6.70
				Profit Deviations			
Balanced Sample	-2.53	-0.22	-0.07	0.00	0.08	0.20	1.14
Private Companies	-2.51	-0.22	-0.07	0.00	0.08	0.20	1.13
Public Companies	-9.20	-1.12	-0.31	0.00	0.23	0.55	12.86

Note: Mean of labor deviations for private (public) firms: 0.35 (0.27).
Mean of profit deviations for private (public) firms: -0.40 (-0,50)

We can further analyze the fluctuations in labor and profit shares of the firms, which are reported in 4.6. We calculate the deviation as the difference between each firm's share from their time-series average. The results reveal how dispersed the deviations are. For private firms, labor (profit) share exceeds its time series mean by 1.65 (1.13) at the 99th percentile. Furthermore, it falls as much as .78 (2.51) at the first percentile. For public firms, the labor (profit) share is a staggering 6.70 (12.86) at the top percentile and falls vastly by 7.78 (9.20) at first. To further highlight how much the variance in labor shares affects the profit shares, we can calculate the correlations between the shares for each type of company. For private firms, it is nearly perfectly negatively correlated with a correlation coefficient equal to -.99. For public firms; however, the correlation coefficient is -.95.

Arellano et al. (2016) highlight the risks of a firm's labor and capital choices. Specifically, they highlight that labor inputs are risky because the firms hire employees and increase their financial obligations before producing any output and generating income. Therefore, any transitory shock the firm experiences make the firm's input choices risky. Additionally, Jensen (1986) studies the principal agent agency problem if the firm has a large buffer to combat these shocks. The principal-agent problem arises because the managers exploit these funds for their benefit. In contrast, the shareholders would expect these funds to be paid out knowing the managers' interest. This emphasizes the risk companies entail when deciding their input choices.

In summary, we have shown that firms have dispersed and persistent returns to their wealth and face significant and fat-tailed changes in output. Additionally, we have documented the imperfect pass-through of input choices (labor and capital) and how they comove with changes in the output growth rate.

5 Determinants of Returns

5.1 Determinants of Profitability

The accounting literature determines profitability in many ways; however, the most used measures are return on assets (ROA), return on equity (ROE), and returns on sales (ROS). Usually, these measures are determined using earnings before financial costs and taxes since the last ones are irrelevant to operational activity. Capon et al. (1990) report that they found a significant positive effect of market shares and growth proxied by sales or assets growth on financial profitability. In addition, they report no significant effect of the size and a negative effect of debt. These results are consistent between industry and firm levels. When analyzing the profitability in Norwegian salmon farming, Sikveland et al. (2021) used firm size and price to explain profits and controlled for variables such as operating leverage, financial leverage, and working capital. The operating leverage and working capital have negative and positive effects, respectively, on profitability (Sikveland et al., 2021, p.428) These findings excite us since they used a sample of Norwegian firms. However, we must note that they estimate these results for the salmon farming industry in Norway, so we must consider them cautiously for our analysis.

5.1.1 Constructing the Variables

We calculate the firm's market share in its corresponding primary activity sector. We built this variable by dividing the revenue of a firm *i* at time *t* by the total revenues made by all the firms in the same industry at time *t*. Furthermore, we construct the firms' operating leverage, defined as total fixed assets over total assets, as a proxy to capture the cost structure of the firm, mainly the mixture of fixed and variable costs Selling and Stickney (1989). We calculate the firm's working capital by adding the sum of current assets and liabilities to the firm's total assets, a measure found to impact profitability significantly Deloof (2003). We also use the firm's financial leverage as its long-term liabilities over its total assets. We introduce the firms' liquidity ratio as the total current assets net of inventory over their current liabilities. A higher liquidity ratio means that the firm can better meet its short-term financial obligations. Goddard et al. (2005) found that the liquidity ratio positively impacts profitability.

We also use the firm's output and ROA, defined in the previous section. Additionally, we use the firms' operating income, as reported in their income statements.

Lastly, we introduce measures of productivity. We calculate the firm's output share of labor as the output divided by its labor costs. The output share of capital is output divided by the firm's capital. We calculate the output share of assets by dividing the output by its total assets. By including these measures, we can formally inspect the firm's return's and productivity relationship.

We deal with potential measurement errors in the same way as in the previous section. Specifically, we winsorize all the variables mentioned above at the 1st and 99th percentile, except the market share.

5.1.2 Summary Statistics

We report the summary statistics of these variables by private firms, public firms, and the full sample[2] on table 5.1 The mean of returns on assets is around 15% in the entire sample and for private firms. However, it is around 9% for public firms. We can also see that the mean of market share is very low at less than 0.1% for private and public firms indicating that, on average, Norwegian industries are competitive.

[2]The full sample is described in chapter 3.

Additionally, the 90th percentile of market shares is surprisingly low; however, when we expand the percentiles, the highest reaches 100%, indicative of monopolies and firms with high monopolistic power in Norway. Regarding productivity measures, we notice that public firms have more productive labor, with an average output share of labor equal to 6.24 compared to private, which equals 3.92. On the other hand, private firms are much more productive concerning capital and assets. Their output share of capital (assets) is 27 (0.76), whereas the public has 8.5 (0.08).

The high productivity can be attributed to many factors; for one, it can imply better managerial skills when making strategic decisions on deciding excess capacity. Bartelsman and Doms (2000) points out that firms producing the same good do not necessarily deploy the same amount of capital. Thereby, differences in the output share of capital are bound to occur.

Private firms have a lower liquidity ratio, financial leverage, and operating leverage. These differences communicate that public firms are in a better financial position than private firms. We report the summary statistics for publicly listed and non-listed companies in the appendix.

Table 5.1: Summary Statistics

	mean	p10	p25	p50	p75	p90
Full Sample						
ROA	0.15	-0.06	0.04	0.13	0.26	0.41
Total Assets	266,68	6,10	14,35	39,44	118,71	407,36
Operating Income	247,21	6,39	17,85	51,12	156,02	486,50
Market Share	0.0009	6.8e-6	2.04e-5	7.53e-5	0.0002	0.00103
Output Share of Labor	8.71	0.91	1.07	1.25	1.70	4.95
Output Share of Capital	26.51	0.17	1.12	4.83	16.91	54.74
Output Share of Assets	0.78	0.08	0.28	0.63	1.10	1.71
Operating Leverage	0.33	0.02	0.08	0.23	0.54	0.83
Working Capital	0.22	-0.09	0.04	0.20	0.40	0.59
Liquidity Ratio	2.37	0.44	0.82	1.23	1.96	3.81
Financial Leverage	0.16	0.00	0.00	0.00	0.27	0.55
Private						
ROA	0.16	-0.02	0.06	0.14	0.26	0.39
Total Assets	335,75	9,78	21,30	53,64	154,48	534,43
Operating Income	361,79	11,01	27,35	74,50	221,80	684,94
Market Share	0.01623	1.39e-5	3.8e-5	0.00013	0.000498	0.00185
Output Share of Labor	3.92	0.97	1.09	1.25	1.64	3.39
Output Share of Capital	27.82	0.30	1.38	5.41	18.04	57.04
Output Share of Assets	0.77	0.11	0.32	0.63	1.06	1.60
Operating Leverage	0.31	0.02	0.07	0.21	0.50	0.79
Working Capital	0.25	-0.05	0.07	0.23	0.42	0.60
Liquidity Ratio	2.19	0.49	0.87	1.27	1.99	3.71
Financial Leverage	0.14	0.00	0.00	0.00	0.23	0.49
Public						
ROA	-0.00	-0.24	-0.03	0.04	0.11	0.22
Total Assets	72 720,26	1 092,47	2 707,01	8 266,98	39 707,03	141 273,51
Operating Income	6 129,67	0,57	63,66	393,23	1 900,84	8 567,85
Market Share	0.0188	1.54e-6	0.0001	0.0009	0.004	0.0229
Output Share of Labor	6.24	-4.34	0.32	2.04	7.39	22.38
Output Share of Capital	8.52	-6.77	0.14	1.65	8.52	35.55
Output Share of Assets	0.08	-0.14	0.01	0.07	0.16	0.34
Operating Leverage	0.68	0.29	0.54	0.75	0.88	0.95
Working Capital	0.10	-0.22	-0.07	0.06	0.22	0.48
Liquidity Ratio	5.49	0.28	0.65	1.37	3.21	10.19
Financial Leverage	0.20	0.00	0.00	0.15	0.32	0.47

Note: These statistics is using the **Full sample**
Total Assets and Operating Income are expressed in 100 000 NOK.

5.2 The Model

To estimate our model, we run a panel regression on three samples:

1. Estimates the ROA for all the firms
2. Estimates the returns for private firms
3. Estimates the returns for public firms.

Additionally, we run the same regression on samples 4 and 5, which are for public non-listed and listed companies, respectively. These are reported in the appendix. As noted by Dkhili and Dhiab (2018), panel data leads to less multicollinearity due to its two dimensions, in our case: individual firms and years. The models are estimated using the following specification:

$$\begin{gathered} ROA_{it} = \alpha \;+\; \alpha_1 Market\_Share_{it} \;+\; \alpha_2 Total\_Assets_{it} \;+\; \alpha_3 Output_{it} \;+ \\ \alpha_4 Operating\_Income_{it} \;+\; \alpha_4 Capital\_Share\_of\_Assets_{it} \;+ \\ \alpha_5 Capital\_Share\_of\_Capital_{it} \;+\; \alpha_6 Capital\_Share\_of\_Labor_{it} \;+ \\ \alpha_7 Operating\_Leverage_{it} \;+\; \alpha_8 Working\_Capital_{it} \;+ \\ \alpha_8 Financial\_Leverage_{it} \;+\; \alpha_9 Liquidity\_Ratio \;+ \\ \alpha_{10} Age_{it} \;+\; \Sigma_{Year}\alpha_{Year}D_{Year} \;+\; \Sigma_{Industry}\alpha_{Industry}I_{Industry} \;+\; \mu_i \\ \hfill , \mu \overset{\text{iid}}{\sim} N(0, \sigma^2) \end{gathered} \tag{5.1}$$

The variables: $Total\ Assets$, $Output$ and $Operating\ Income$ are logged.

D_{year} is a year-specific dummy, and $I_{Industry}$ is an industry-specific dummy. We include these to control for potential year and industry fixed effects. Additionally, we include a dummy variable in Subsample 1, indicating whether the company is public or private (1 if public, 0 otherwise). In addition, we computed the variance-covariance matrix to control for any potential presence of multicollinearity, which is reported in the appendix. Evidently, there is a high correlation between *Total Assets*, *Output*, and *Operating Income*. We also want to examine the effect on the return of firm size; therefore, we run four separate regressions, one without a variable for firm size and the other three including only one of the abovementioned variables.

5.3 Results and Discussion

Table 5.2, 5.3, and 5.4 report the results for samples 1, 2, and 3, respectively. In sample 1, the company size significantly and positively affects returns regardless of which measure of firm-size we are using. These results were coherent with those found in private and public firms (separately).

Before introducing our size measures, the firms' market shares had a positive, large, and significant effect on returns. However, once we introduce any of our size measures, the market shares' effect becomes insignificant in regression 3 (where output is the measure of size). In regression 4 (where operating income is the measure of size), the market shares' effect on returns turned out to be negative for private and all firms. Both effects were significant, but the estimate of market share in the second sample is more significant, being a bigger negative number. A similar result is observed in the second regression (where the assets are the measure of size), but this time similarly significant for all the firms and private firms. However, market shares' effect on returns for public firms stayed positive with all the size measures, but it was not significant.

Another interesting finding of our regressions is a significant positive relationship between the three productivity measures we presented earlier and returns. Even though significant and positive, we observe that the output share of capital is the least impactful of our productivity measures, with the lowest coefficient in all the models' estimations. Reflecting on the reason behind this finding, we suspect that the reason is that the output share of assets already explains everything the output share of capital explains. After all, the capital is nothing more than the sum of the tangible and intangible assets already included in the total assets used to compute the asset share.

Another interesting finding is that the output share of labor is the most important one of our productivity measures that affect returns being the profitability measure with the highest estimated coefficient in all the regressions for all the samples. The only exception is the second regression, where we used the total assets as our size measure for the private and all firms' panels. The potential reason behind this exception is the higher correlation between the output share of assets and total assets compared to its correlation with the other size measures we used in the other regressions (see the correlation matrix in the appendix). That exception is not valid in the public firms' panel. The reason behind it is the higher importance of the output share of capital in the public firm's panel in particular. This importance can be observed in the difference between the output share of labor and the other productivity measures' coefficients. The difference is significant enough to absorb the increase in the importance of the share of assets.

Our models provide evidence of a negative relationship between age and returns. Interestingly, the coefficients from all the regressions on the public firms' panel are around two times larger than the ones from the coefficients of the private panel. The first sample, with all the firms, shows relatively close coefficients to the private firms. Doğan (2013) showed a negative relationship between firm-age and the returns in Turkish firms. Even though, his study suggested that younger firms experience a decline in their return but could increase their return in the latter stages of their life-cycle.

Operating leverage and working capital significantly positively affect private and public firms' returns. This is the case in all the regressions except the one where our size measure is output. In the last regression, the operating leverage and working capital significantly negatively affect returns.

Finally, we found that financial leverage and liquidity negatively affect profitability, with two exceptions. The first one is the financial leverage positively affecting the returns in the three panels when we run the third regression. Moreover, the second one is the positive effect of the liquidity ratio in the private firms' panel and the one with all the firms.

Taking a step back, we notice that the first, second, and fourth regressions tell the same story of how our independent variables of choice affect returns. However, the third regression (the one with output as a size measure) tweaks the story.

The sample with only public firms have the highest explanatory power when we consider the r-squares. One potential reason is the higher dispersion of returns of the private firms compared to the public ones. Finally, we also notice that sample 1 and 2's estimates are comparable, even after introducing the dummy variable controlling for the firms' profiles when estimating the first sample. This can be explained by the difference in the number of observations in the private and public firms (of private

Table 5.2: Model A: All Companies

	Regression 1 Est.	Regression 2 Est.	Regression 3 Est.	Regression 4 Est.
Market Share	0.364***	-0.0825**	-0.232	-0.0985***
Total Assets		0.0266***		
Output			0.128***	
Operating Income				0.0222***
Output Share of Assets	0.0698***	0.0792***	0.196***	0.0648***
Output Share of Capital	0.00671***	0.00394***	0.0393***	0.00672***
Output Share of Labor	0.0732***	0.0702***	0.310***	0.0776***
Operating Leverage	0.00511***	0.000104	-1.472***	0.0244***
Working Capital	0.0619***	0.0668***	-1.280***	0.0700***
Financial Leverage	-0.0832***	-0.104***	1.586***	-0.0989***
Liquidity Ratio	-0.00112***	-0.00101***	0.00163***	-0.000437***
Age	-0.000868***	-0.00177***	-0.00815***	-0.00152***
Public	-0.0101	-0.0581***	-0.197**	-0.0288***
Constant	0.205***	-0.173***	-0.369***	-0.121***
Within R^2	0.267	0.284	0.0961	0.282
Between R^2	0.241	0.238	0.112	0.242
Overall R^2	0.248	0.251	0.138	0.253

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Note: Regression based on 1074455 observations

firms obs VS. of private firms obs) in addition to our dummy variable being equal to zero when the firm is private.

Table 5.3: Model B: Private Companies

	Regression 1 Est.	Regression 2 Est.	Regression 3 Est.	Regression 4 Est.
Market Share	0.369***	-0.105***	-0.195	-0.119***
Total Assets		0.0268***		
Output			0.125***	
Operating Income				0.0225***
Output Share of Assets	0.0705***	0.0799***	0.193***	0.0654***
Output Share of Capital	0.00666***	0.00391***	0.0385***	0.00669***
Output Share of Labor	0.0720***	0.0690***	0.299***	0.0764***
Operating Leverage	-0.000679	-0.00582***	-1.511***	0.0186***
Working Capital	0.0558***	0.0607***	-1.319***	0.0639***
Financial Leverage	-0.0812***	-0.102***	1.597***	-0.0971***
Liquidity Ratio	-0.00105***	-0.000939***	0.00187***	-0.000364***
Age	-0.000854***	-0.00175***	-0.00804***	-0.00150***
Constant	0.210***	-0.172***	-0.301***	-0.120***
Within R^2	0.267	0.283	0.0980	0.282
Between R^2	0.242	0.239	0.112	0.242
Overall R^2	0.249	0.251	0.141	0.254

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Note: Regression based on 1035430 observations

Table 5.4: Model C: Public Companies

	Regression 1 Est.	Regression 2 Est.	Regression 3 Est.	Regression 4 Est.
Market Share	0.336***	0.190*	0.0451	0.151
Total Assets		0.0137***		
Output			0.132***	
Operating Income				0.0131***
Output Share of Assets	0.0567***	0.0638***	0.326***	0.0545***
Output Share of Capital	0.00490***	0.00215*	0.0365***	0.00457***
Output Share of Labor	0.111***	0.109***	0.692***	0.114***
Operating Leverage	0.140***	0.135***	-0.549***	0.156***
Working Capital	0.197***	0.205***	-0.348***	0.205***
Financial Leverage	-0.140***	-0.157***	1.345***	-0.154***
Liquidity Ratio	-0.00279***	-0.00274***	-0.0152**	-0.00217***
Age	-0.00160***	-0.00230***	-0.0126***	-0.00219***
Constant	0.116**	-0.0759	-1.060*	-0.0858
Within R^2	0.335	0.345	0.111	0.346
Between R^2	0.279	0.277	0.146	0.282
Overall R^2	0.281	0.280	0.126	0.288

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Note: Regression based on 39025 observations

6 Determinants of the Decision to Go Public

6.1 The Probability Model

Pagano et al. (2022) ask why companies decide to get listed on the stock exchange and others do not. Although eligible, we have already revealed that many companies in our sample do not get listed. After studying the determinants of profitability, we ask: To what degree can this understanding allow us to predict which companies get listed and which variables (accounting/financial) are relevant for this probability estimation?

We begin our analysis by first comparing large public firms to large private firms. We use firms with assets worth more than 3Bn NOK. This restriction gives us two subsamples: big private - and public firms. Both of which have total assets on average worth 9.6Bn NOK. The subsample for private firms consists of 479 unique entities and the public firms consist of 344 unique entities. This sub-analysis allows us to consider how the companies differ in different productivity measures and their returns.

Table 6.1: Comparable Private Firms vs. Public Firms

	st.d	p10	p25	p50	p75	p90
			Private Firms			
Assets	11 174,72	3 299,04	3 868,27	5 502,91	9 084,55	16 826,75
ROA	0.14	-0.00	0.04	0.08	0.15	0.27
Output Share of Labor	682.89	1.05	1.69	4.84	15.20	57.27
Output Share of Capital	233.56	0.08	0.20	1.14	6.25	38.24
Output Share of Assets	0.16	0.02	0.05	0.11	0.21	0.38
Financial Leverage	0.26	0.00	0.01	0.24	0.51	0.65
Liquidity Ratio	9.93	0.28	0.56	1.00	1.85	5.88
			Public Firms			
Assets	17 546,21	83,28	243,96	819,85	3 612,64	11 446,66
ROA	0.27	-0.29	-0.05	0.03	0.11	0.21
Output Share of Labor	33.07	-6.75	-0.40	1.82	7.24	25.39
Output Share of Capital	385.49	-10.68	-0.16	1.37	8.95	45.08
Output Share of Assets	0.27	-0.18	-0.01	0.06	0.15	0.31
Financial Leverage	0.20	0.00	0.00	0.14	0.32	0.48
Liquidity Ratio	16.38	0.27	0.64	1.41	3.47	11.22

Note: The variables are winsorized at the 1st and 99.9th percentile.

The big median private firm has higher returns, more assets, and is more productive concerning labor and assets than the median public firm. However, the productivity measure related to labor is far more volatile than that of public firms. Moreover, the public firms' returns are almost twice as volatile as the private (private firms have a standard deviation = 0.14, and public firms' = 0.27). Additionally, the returns at the 1st and the 90th percentile are higher for private than public.

The financial leverage is higher for private firms, indicating that the big private companies are heavily debt-ridden. To understand how much of the returns are attributable to their financial leverage - meaning that companies could borrow at a lower cost and generate a greater return, we regressed the financial leverage on the returns for each sub-sample but found no significant results. The fact remains that private firms have a higher debt-to-assets ratio, which implies a higher tolerance to risk.

The public firms are more productive with their capital, with the 50th and the 90th being higher than the private firms. However, it is far more volatile, and the bottom ten and 25 percentile are negative compared to private companies, which indicates that some private companies are very productive. In contrast, others could be more productive concerning their capital.

6.1.1 Who Can Get Listed?

In order to be eligible for listing a company in Norway on the stock exchange, some conditions must be met. Firstly, the gender quotas require a representation of both genders based on the number of board members (LOV-1997-06-13-45). Furthermore, the company must have a minimum share capital of 2M NOK in cash or non-cash assets. Additionally, registration costs, auditing, certifications, and adherence to the Public Limited Liability Companies Act are some facets the company must consider. By the measure of total equity held, we can estimate the number of companies eligible to become publicly listed. Assuming all other conditions are met, 48 073 private firms and all non-listed companies are eligible to become listed.

A capital of 1M NOK is enough to become a public company (ASA) in Norway, and technically 2M NOK market value is enough to get listed in the Oslo Stock Exchange Børs (2022)[3]. We made our choice of restriction after observing that the minimum

[3]Oslo Børs requires a minimum of 200 shareholders, each holding at least 10 000 equity.

total assets of both listed companies and the newly listed ones are slightly over 10M NOK.

We use the **full sample** to conduct this analysis, including only companies with total assets equal to or more significant than 10M NOK, even though In our sample, the number of companies that became publicly listed is 56. In the period between 2007 and 2019, the average of new listings is four each year.

To perform this analysis, we estimate a probit model of the probability of a firm going to IPO. The dependent variable $List_{it}$ is equal to 1 if the firm went public, and 0 otherwise. In our estimated model we tried to include as many variables argued to affect the probability to go listed and introduced some productivity measures mentioned in the previous section. We estimate the following model of the probability of becoming listed:

$$\begin{aligned} Pr(List_{it} = 1) = F(&\alpha_1\, ROA_{it} + \alpha_2\, Revenue_{it} + \alpha_3\, CAPEX_{it} + \alpha_4\, Growth_{it} \\ &+\alpha_5\, Financial\_Leverage_{it} + \alpha_7\, Output\_to\_Labor_{it} \\ &+\alpha_8\, Output\_to\_Capital_{it} + \alpha_9\, age + \gamma_1\, Year_t + \gamma_2\, Industry_t) \end{aligned} \tag{6.1}$$

The ROA, $Financial\ Leverage$, and the productivity measures $Output\ to\ Capital$ and $Output\ to\ Labor$ are lagged by one period. $Revenue$ is the lagged value of the natural logarithm of firm i's sales at time t. We define $CAPEX$ as the lagged natural logarithm of the firms' tangible fixed assets' expenditures summed with depreciation. $Growth$ is defined as the growth rate of the firm's revenue. Age is the company's age. Additionally, we control for any year and industry fixed effects.

Pagano et al. (p. 36, 2022) describe many theories about the costs and benefits of going public and nest the essential predictors in a model. They cite Chemmanur and Fulghieri (1999) when discussing the challenge of adverse selection costs, particularly for younger firms. They argue that because of this, the age/size of the company should positively correlate with the probability of going public Pagano et al. (p. 36, 2022). Additionally, they highlight the administrative expenses related to going public. As such, more prominent firms would be better suited to handle these costs.

Unlike Pagano et al. (2022), we have information regarding the firms' age, which we include in our model. Furthermore, we include total assets as a proxy for firm size. Similarly to Pagano et al. (2022), we expect a positive relationship between a firm's growth rate, leverage, and our measure of CAPEX.

Unlike Pagano et al. (2022), we do not have information about the firm's market-to-book value. We considered including other measures, such as return on equity, as a proxy. However, this would produce multicollinearity because of the high correlation between the return on assets and the return on equity. Furthermore, they include the concentration of borrowing using the Herfindahl index, something we also disclude from our model. In addition, we need to include data on the interest rate the firms are paying on their long-term debt, a necessary measure to compute the bank rate. The last one is a comparative ratio between the cost of debt of a firm *i* and the average of its industry. We tried to compute it manually by dividing the interest expense of a firm by its previous year's level of debt. However, we had to drop a large proportion of data. Their results showed no significant impact of these variables, leading us to exclude them from our model.

6.2 Results

Table 6.2: Determinants of the Decision to Go Public

	Pr(List = 1)	
	Est.	St.d
ROA	-0.889*	0.440
Reveneue	0.0967**	0.0373
Capex	1.70e-12	1.61e-11
Growth	0.115**	0.0400
Financial Leverage	0.0332	0.337
Output Share of Labor	-0.00000654	0.0000463
Output Share of Capital	1.85e-08	0.000000276
Output Share of Assets	-0.502***	0.127
Age	-0.00439	0.00541

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Note: The regression is based on 69 703 observations

The table above reports the maximum likelihood estimates of our probit regression and the corresponding standard errors. Unlike Pagano et al. (2022), we do not differentiate between independent companies and subsidiaries of listed companies nor distinguish between IPOs of independent companies and carve out.

Our reported results reveal the importance of the company size in dictating whether a company will become publically traded. The company's financing needs proxied by the CAPEX (investment) and growth appear to be important in the company's probability of getting listed. However, the coefficient for the investment is not statistically significant, while the one for growth is only significant at 10 percent. The probability of a firm getting listed is decreasing in the firm's financial leverage and age. However, the coefficients for the two are statistically insignificant. The firm's labor and capital productivity have a negative and a positive effect on the probability of an IPO occurring, respectively. But those effects are not statistically significant. In the other hand, the productivity of assets significantly increases the probability of a firm going listed. Finally, returns (ROA) seem to have a negative effect on an IPO significant at 10 percent.

Our reported results reveal the importance of the company size in dictating whether a company will become publically traded. The company's financing needs proxied by the CAPEX (investment) and growth are essential in the company's probability of getting listed. However, the coefficient for the investment is not statistically significant, while the one for growth is only significant at 10 percent. The probability of a firm getting listed decreases in the firm's financial leverage and age. However, the coefficient for the two is statistically insignificant. The firm's labor and capital productivity have a negative and a positive effect on the probability of an IPO occurring. However, those effects are not statistically significant. Conversely, the productivity of assets significantly increases the probability of a firm going listed. Comparing to Pagano et al. (2022)'s findings, we found similar effects of revenue, CAPEX, growth, and leverage on the probability of getting listed.

Interestingly, the returns hurt a potential IPO significantly at 10 percent. This finding contradicts Pagano et al. (2022)'s findings, which reported the returns' effect as positive. However, as Pagano et al. (2022) points out, the relationship between the returns and the probability of being listed is ambiguous. On one side, a high ROA can indicate that the firm does not need external validity, decreasing the probability of becoming listed. On the other hand, it can suggest a transitory peak of the firms' return, making it more appealing for them to become listed before their returns start decreasing again.

Our model has some limitations. The most relevant one is that the sample we used is unbalanced regarding how many companies get listed (56 firms) and the ones that can get listed but don't (55 324 firms). The thing that will be harmed the most is the explanatory power of our model, but it might also introduce some biases to our estimates.

As mentioned, we controlled for year and industry-fixed effects (unreported). We plotted the numbers of new IPOs per year (reported in the appendix), and it seemed that there is no particular year with an exceptionally high number of listings. The results from controlling for years in our regression confirm this conclusion. We had no significant effect on the probability of firms getting listed all the years. In figure 6.1, we plotted the IPOs in our sample by industry[4].

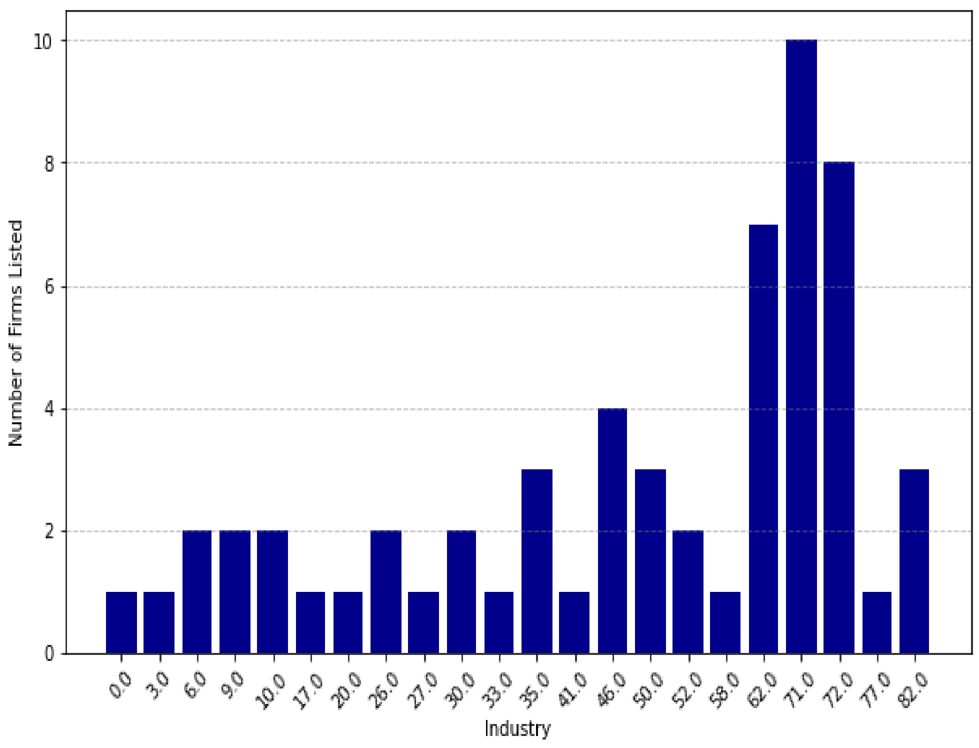

Figure 6.1: IPOs by Industry

Most of the IPOs happened in three industries: (1) the rental of machinery and equipment, (2) computers and related activities, and (3) air transport. The results from controlling for the industries' fixed effect, i.e., the coefficients for the dummy variable designating the industry, are all negative. However, only five of them were

[4]The industry classification can be found in: https://www.ssb.no/en/klass/klassifikasjoner/6

significant. The related industries are (1) fishing, (2) coal mining, (3) manufacturing of vehicles, (4) collection and purification of water, and (5) wholesale trade excluding vehicles. We suspect that these industries affect the probability of their firms getting listed due to some conditions the stock exchange implies for these companies, such as ESG-related ones. However, we must note that the model omitted the fixed effects of industries with no listings in our sample, which makes those results not robust.

We also run the model including firm-specific effects, which we believe have a significant effect on our estimates. Pagano et al. (2022) explained that practitioners talk about cultural resistance as a potential source of such effects. Some entrepreneurs, for example, will be more conservative when it comes to getting their businesses listed on a stock exchange. Our hypothesis was correct, and the output share of assets was negative and significant at 0.01%, while the growth remained positive but on a lower significance level. The results are reported in the appendix.

After examining how the Return on Assets (ROA) affects Norwegian firms' decision to become publically traded, we wanted to investigate how such a decision impacts the ROA. To do this, we conducted a quadratic discontinuity regression analysis.

First, we calculated the difference between the year of the observed ROA and the year of listing, which we called the "year after listing" variable. When this variable equals zero, it designates the year the company went public. If it is one, it designates one year after the firm went public, and so on. Negative values designate years before the event of going public occurs. For example, -1 represents one year before the listing.

Next, we created bins based on the "year after listing" variable, where each bin represents a year around the IPO's year. We calculated the means of the bins, resulting in the average observed ROA a year around the IPO.

In our discontinuity regression, we used ROA as our dependent variable and 'year after listing' as our independent variable. We also limit the observations to the ones between minus seven and seven since the ones outside this margin are very few, which introduces biases. The results from our regression are summarized and presented graphically in figure 6.2.

The x-axis presents 'year after listing, 'and the y-axis presents the mean of observed ROA for the firms in the corresponding year before or after the IPO. The dots are the scatter plot for the two presented variables; the red curve is the fitted curve from our quadratic regression for observations before IPO, the blue curve is the fitted curve from our quadratic regression for observations after IPO, and the grey bands represent the 90 percent confidence intervals.

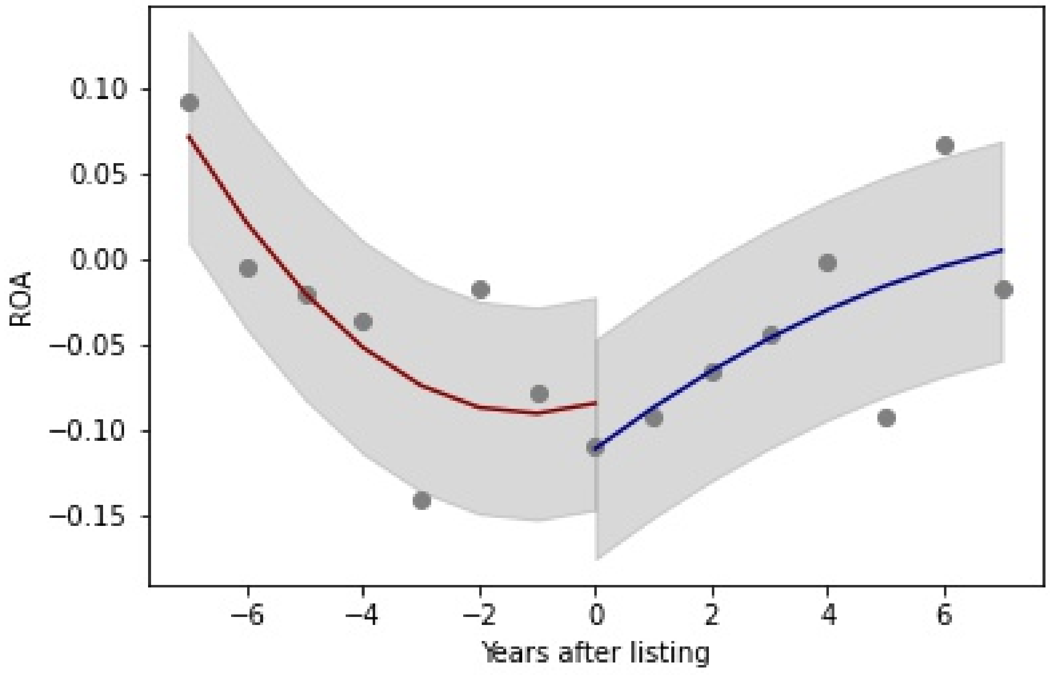

Figure 6.2: Quadratic Discontinuity Regression

The results show that the ROA decreases when a company goes listed. However, this result is statistically insignificant. What's interesting, on the other hand, is that the ROA has a decreasing path before the firms got listed and an increasing path afterward.

Another intriguing aspect to explore is the potential for firms to access cheaper debt following an initial public offering (IPO). (**?**; Basil(1988)) has suggested that this is one of the advantages of becoming a publicly traded company. Rajan (1992) has further argued that access to more affordable capital may arise due to increased bargaining power when negotiating with banks. However, as our thesis primarily focuses on returns, investigating this specific aspect falls outside its scope. Therefore, we consider it an area for further research and exploration.

7 Conclusion

In conclusion, our study examined returns as a practical approach to understanding wealth disparities using micro data from Norwegian private and public firms. The first contribution of our thesis is highlighting that returns of Norwegian private and public firms are dispersed and persistent. In addition, Norwegian firms face significant and fat-tailed changes in output, and that imperfect pass-through of input choices and their comovement with changes in the output growth rate. Our thesis' second contribution is that firm size and productivity positively influence returns. The third and last contribution is that returns, firm growth, and asset productivity predicted a firm's likelihood of becoming publically traded. These findings provide valuable insights for investors, policymakers, and market participants. We hope these findings are intriguing for further research.

References

Arellano, C., Bai, Y., and Kehoe, P. (2016). Financial frictions and fluctuations in volatility. *NBER Working Paper No. 22990.*

Bach, L., Calvet, L. E., and Sodini, P. (2020). Rich pickings? risk, return, and skill in household wealth. *American Economic Review*, 110(9):2703–47.

Bartelsman, E. and Doms, M. (2000). Understanding productivity: Lessons from longitudinal microdata. *Journal of Economic Literature*, 38(3):569–594.

Benhabib, J., Bisin, A., and Luo, M. (2017). Earnings inequality and other determinants of wealth inequality. *The American Economic Review*, 107(5):593–597.

Benhabib, J., Bisin, A., and Luo, M. (2019). Wealth distribution and social mobility in the us: A quantitative approach. *American Economic Review*, 109(5):1623–1647.

Benhabib, J., Bisin, A., and Zhu, S. (2011). The distribution of wealth and fiscal policy in economies with finitely lived agents. *Econometrica*, 79(1):123–157.

Boar, C., Gorea, D., and Midrigan, V. (2023). Why are returns to private business wealth so dispersed? (29705).

Børs, O. (2022). Oslo rulebook ii - issuer rules. page 25.

Capon, N., Farley, J., and Hoenig, S. (1990). A meta-analysis of financial performance. *Management Science*, 36:1143–1159.

Chemmanur, T. J. and Fulghieri, P. (1999). A theory of the going-public decision. *The Review of Financial Studies*, 12(2):249–279.

Cooper, R. W. and Haltiwanger, J. C. (2006). On the nature of capital adjustment costs. *The Review of Economic Studies*, 73(3):611–633.

Dash, C. S. K., Ajit, K. B., Satchidananda, D., and Ashish, G. (2023). An outliers detection and elimination framework in classification task of data mining. *Decision Analytics Journal Volume 6, March 2023, 100164.*

Deloof, M. (2003). Does working capital management affect profitability of belgian firms? *Journal of Business Finance Accounting*, 30:573–588.

Dkhili, H. and Dhiab, L. B. (2018). The relationship between economic freedom and fdi versus economic growth: Evidence from the gcc countries. *Journal of Risk and Financial Management*, 11(4).

Donangelo, A., Gourio, F., Kehrig, M., and Palacios, M. (2019). The cross-section of labor leverage and equity returns. *Journal of Financial Economics*, 132(2):497–518.

Doğan, M. (2013). Does firm size affect the firm profitability? evidence from turkey. *Research Journal of Finance and Accounting*, 4:53–59.

Fagereng, A., Guiso, L., Malacrino, D., and Pistaferri, L. (2018). Heterogeneity and persistence in returns to wealth. *Econometrica, Vol. 88, No. 1 (January, 2020), 115–170.*

Goddard, J., Tavakoli, M., and Wilson, J. (2005). Determinants of profitability in european manufacturing and services: Evidence from a dynamic panel data. *Applied Financial Economics*, 15:1269–1282.

Gopniath, G., Kalemli-Özcan, S., Karabarbounis, L., and Villegas-Sanches, C. (2017). Capital allocation and productivity in south europe. *The Quarterly Journal of Economics*, 132(4):1915–1967.

Jensen, M. C. (1986). Agency costs of free cash flow, corporate finance, and takeovers. *The American Economic Review*, 76(2):323–329.

Jones, S. and Hensher, D. (2004). Predicting firm financial distress: A mixed logit model. *Accounting Review - ACCOUNT REV*, 79:1011–1038.

(LOV-1997-06-13-45)., L. (1997). Public limited liability companies act. https://lovdata.no/referanse/hjemmel?dokID=NL/lov/1997-06-13-45.

Moll, B. (2014). Productivity losses from financial frictions: Can self-financing undo capital misallocation. *American Economic Review, 104 (10): 3186-3221.*

Moskowitz, T. J. and Vissing-Jørgensen, A. (2002). The returns to entrepreneurial investment: A private equity premium puzzle? *American Economic Review*, 92(4):745–778.

Pagano, M., Panetta, F., and Zingales, L. (2022). Why do companies go public? an empirical analysis. *The Journal of Finance*, 53(1):27–64.

Piketty, T. (2014). *Capital in the Twenty-First Century*. Harvard University Press.

Rajan, R. G. (1992). Insiders and outsiders: The choice between informed and arm's-length debt. *The Journal of Finance*, 47(4):1367–1400.

Selling, I. T. and Stickney, P. C. (1989). The effects of business environment and strategy on a firm's rate of return on assets. *Financial Analysts Journal , Jan. - Feb., 1989, Vol. 45, No. 1 (Jan. - Feb., 1989), pp. 43-52+68.*

Sikveland, M., Tveterås, R., and Zhang, D. (2021). Profitability differences between public and private firms: The case of norwegian salmon aquaculture. *Aquaculture Economics Management*, 26:1–25.

Smith, M., Yagan, D., Zidar, O., and Zwick, E. (2019). Capitalists in the twenty-first century*. *The Quarterly Journal of Economics*, 134:1675–1745.

SSB (2020). Regnskap. https://www.ssb.no/data-til-forskning/utlan-av-data-til-forskere/variabellister/regnskap.

SSB (n.d). About businesses. https://www.ssb.no/en/statistikk-pa-oppdrag/informasjon-om-virksomheter.

Stanley, A. (2022). Global inequalities. page 25.

Appendix

A1 Additional Material for the Analyses

A1.1 Empirical Analysis

Here we include additional material from our empirical analysis. Specifically, we shoe the dispersion of returns for the publicly listed and non-listed companies.

A1.1.1 Rate of Returns

The table reports the distribution of the returns, by publicly listed and non-listed companies. Similarly to what was reported in chapter 4, we see that both these firm-types experience disperse returns. Moreover, we show the plot of the ROA bin means to the market shares.

Table A1.1: ROA: Listed vs Non-Listed Public Firms

	mean	std	p10	p25	p50	p75	p90	p95
Non-Listed Companies								
π/a	-0.06	0.48	-0.41	-0.13	0.02	0.15	0.28	0.39
$\overline{\pi/a}$	-0.14	0.51	-0.63	-0.17	-0.01	0.13	0.26	0.33
Listed Companies								
ROA	-0.08	0.65	-0.50	-0.09	0.03	0.16	0.33	0.50
$\overline{ROA}$	-0.11	0.42	-0.74	-0.22	0.02	0.10	0.26	0.37

A1.2 Determinants of Profitability

Here we include additional material from our empirical analysis. We include the summary statistics for the listed and non-listed companies, in addition to show the full regression tables including the standard errors. Furthermore, we plot the mean of these variables for each firm type: private, non-listed, and listed, over the sample period. These are shown in figure A1.4 and A1.3

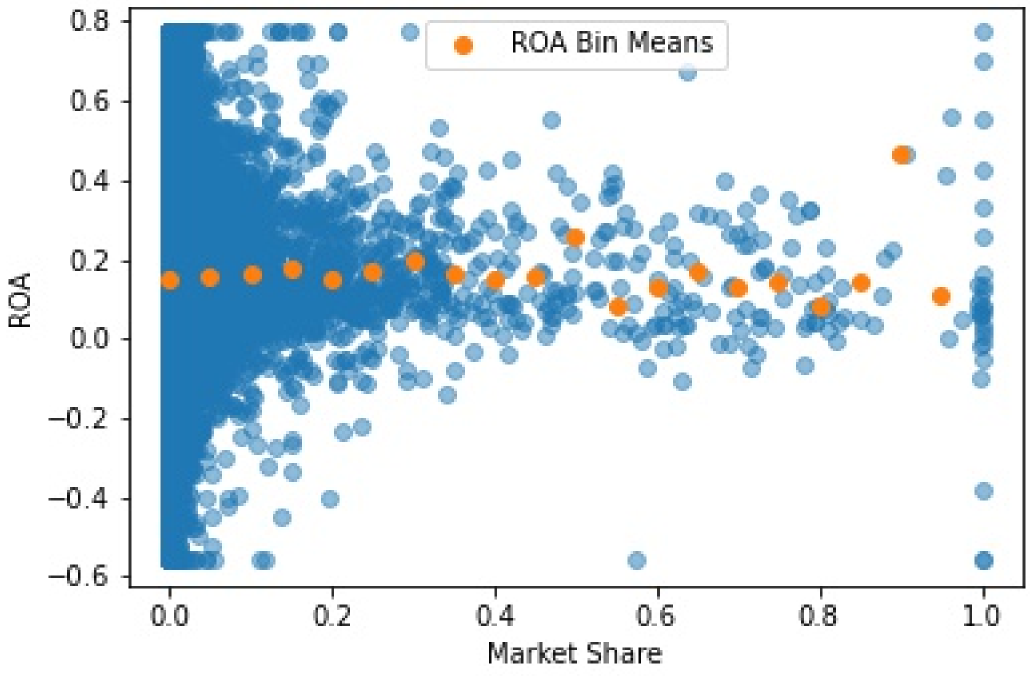

Figure A1.1: Scatterplot: Market Shares vs. ROA

Table A1.2: Summary Statistics from Chapter 5: Public Non-listed vs. Public Listed Companies

	mean	p10	p25	p50	p75	p90
Public Nonlisted Companies						
ROA	-0.01	-0.25	-0.05	0.05	0.11	0.19
Total Assets	20 622,19	340,81	833,07	2 296,09	12 892,87	28 628,13
Operating Income	1 553,26	-	25,37	282,31	1 178,25	3 185,37
Market Share	0.01	0.00	0.00	0.00	0.00	0.01
Output Share of Labor	2.63	-4.44	0.11	1.36	4.26	11.70
Output Share of Capital	5.14	-1.24	0.04	1.21	5.12	26.06
Output Share of Assets	0.15	-0.13	0.00	0.08	0.25	0.55
Operating Leverage	0.62	0.20	0.42	0.71	0.86	0.94
Working Capital	0.11	-0.26	-0.13	0.09	0.27	0.51
Liquidity Ratio	5.11	0.19	0.56	1.32	2.84	8.24
Financial Leverage	0.19	0.00	0.01	0.15	0.30	0.40
Public Listed Companies						
ROA	-0.00	-0.23	-0.03	0.04	0.11	0.22
Total Assets	113 076,65	1 623,66	3 844,48	9 587,06	48 655,06	185 526,34
Operating Income	38 729,37	1,43	74,76	434,52	2 104,77	8 909,86
Market Share	0.02	0.00	0.00	0.00	0.01	0.03
Output Share of Labor	7.14	-4.29	0.38	2.30	8.01	27.15
Output Share of Capital	19.54	-8.14	0.15	1.82	9.09	38.48
Output Share of Assets	0.06	-0.14	0.01	0.07	0.15	0.28
Operating Leverage	0.70	0.32	0.56	0.76	0.89	0.95
Working Capital	0.10	-0.20	-0.06	0.06	0.21	0.48
Liquidity Ratio	5.88	0.30	0.66	1.37	3.31	10.46
Financial Leverage	0.20	0.00	0.00	0.15	0.33	0.49

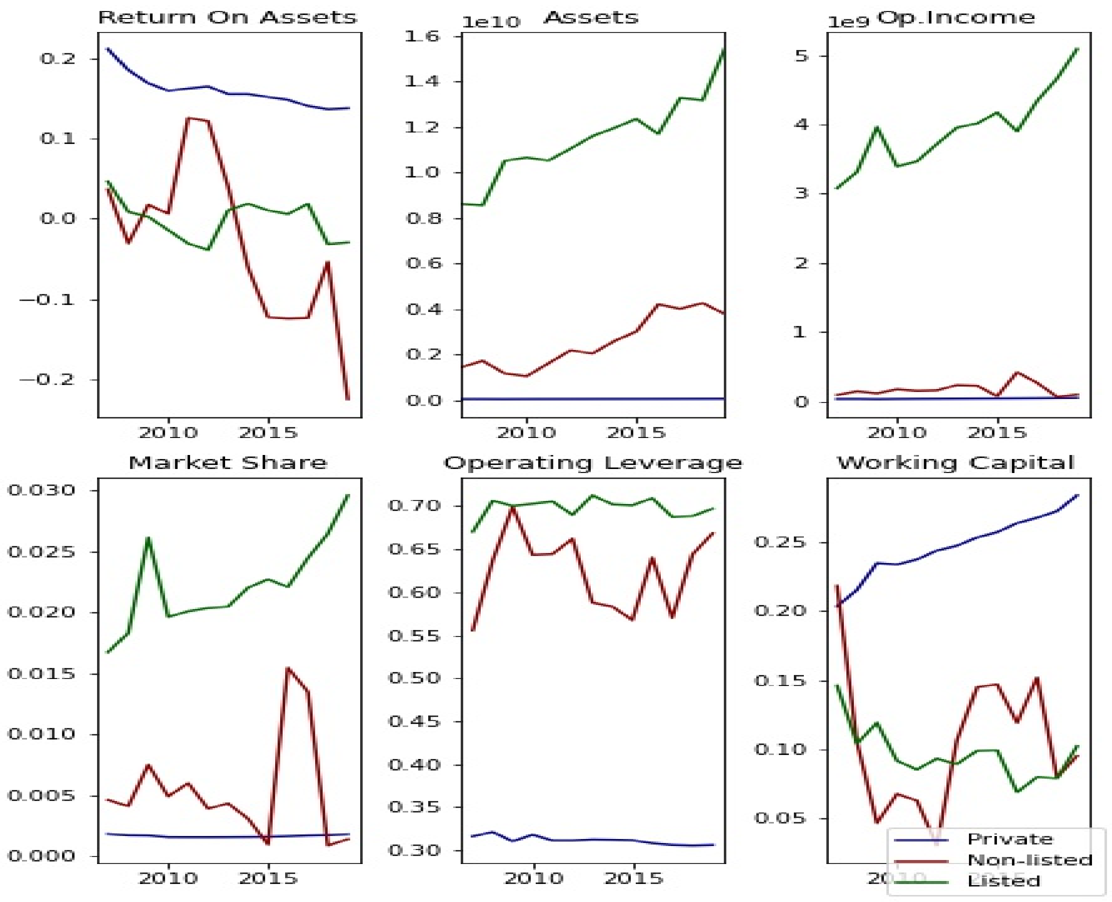

Figure A1.2: Mean of Profitability and Productivity Variables

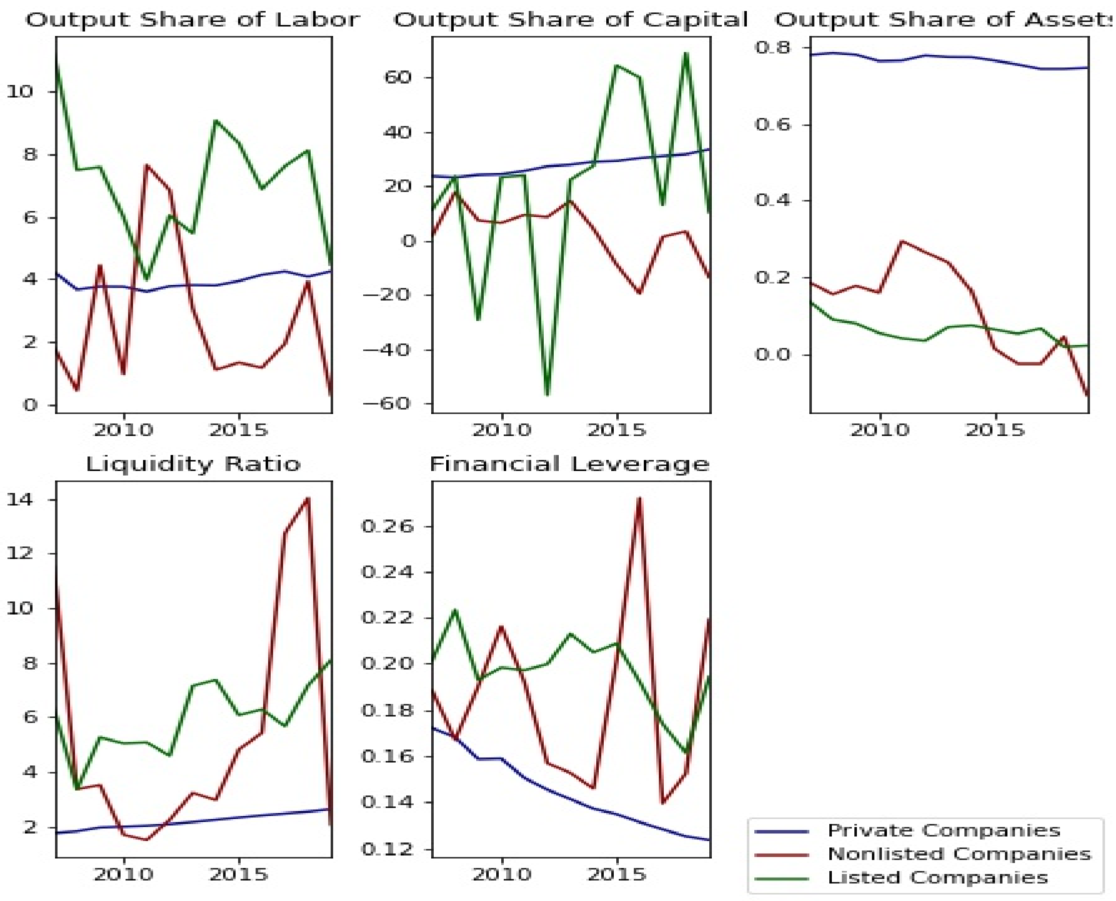

Figure A1.3: Mean of Profitability and Productivity Variables cont.

Table A1.3: Correlation Between Variables in the Full Sample

	ROA	Age	Market Share	Total Assets	Operating Income	Output	Operating Leverage	Working Capital	Financial Leverage	Liquidity Ratio	Output Share of Assets	Output Share of Capital	Output Share of Labor
ROA	1												
Age	-0.0543	1											
Market Share	0.0049	0.0612	1										
Total Assets	-0.0196	0.2993	0.1735	1									
Operating Income	0.1214	0.1920	0.1815	0.6786	1								
Output	0.1705	0.2174	0.1840	0.7746	0.8629	1							
Operating Leverage	-0.1182	0.0492	0.0121	0.2274	-0.2063	-0.0777	1						
Working Capital	0.0261	0.0652	-0.0214	-0.0423	-0.0267	-0.0462	-0.4671	1					
Financial Leverage	-0.1306	-0.0121	-0.0090	0.2000	0.0787	-0.0475	0.5800	-0.0911	1				
Liquidity Ratio	-0.0750	0.0629	-0.0172	0.0743	-0.2545	-0.1217	0.0237	0.2923	0.0392	1			
Output Share of Assets	0.3001	-0.1532	-0.0095	-0.4130	0.1702	0.2562	-0.4593	0.0048	-0.3749	-0.2994	1		
Output Share of Capital	0.1993	-0.0698	-0.0112	-0.1892	0.1784	0.2224	-0.7227	0.2766	-0.5253	-0.0950	0.6093	1	
Output Share of Labor	0.2288	0.0594	-0.0037	0.2226	-0.1776	-0.0209	0.3481	-0.0391	0.2438	0.2303	-0.3702	-0.2755	1

Note:This Correlation Matrix is for the Full Sample

A1.2.1 Result from the Regressions from Chapter 5

We report each model's full regression results, including the standard deviations. Additionally, we include sample 4 and 5, which are the regression run on non-listed and listed companies, respectively.

Table A1.4: Model A: All Companies

	Regression 1		Regression 2		Regression 3		Regression 4	
	Est.	St.d	Est.	St.d	Est.	St.d	Est.	St.d
Market Share	0.364***	0.0281	-0.0825**	0.0283	-0.232	0.277	-0.0985***	0.0283
Total Assets			0.0266***	0.000205				
Output					0.128***	0.00204		
Operating Income							0.0222***	0.000193
Output Share of Assets	0.0698***	0.000266	0.0792***	0.000274	0.196***	0.00288	0.0648***	0.000279
Output Share of Capital	0.00671***	0.000158	0.00394***	0.000158	0.0393***	0.00151	0.00672***	0.000160
Output Share of Labor	0.0732***	0.000180	0.0702***	0.000180	0.310***	0.00173	0.0776***	0.000184
Operating Leverage	0.00511***	0.00132	0.000104	0.00132	-1.472***	0.0127	0.0244***	0.00134
Working Capital	0.0619***	0.000807	0.0668***	0.000804	-1.280***	0.00777	0.0700***	0.000807
Financial Leverage	-0.0832***	0.00116	-0.104***	0.00116	1.586***	0.0112	-0.0989***	0.00117
Liquidity Ratio	-0.00112***	0.0000455	-0.00101***	0.0000452	0.00163***	0.000430	-0.000437***	0.0000484
Age	-0.000868***	0.0000274	-0.00177***	0.0000284	-0.00815***	0.000291	-0.00152***	0.0000281
Public	-0.0101	0.00732	-0.0581***	0.00731	-0.197**	0.0711	-0.0288***	0.00754
Constant	0.205***	0.00694	-0.173***	0.00755	-0.369***	0.0761	-0.121***	0.00799
Within R^2	0.267		0.284		0.0961		0.282	
Between R^2	0.241		0.238		0.112		0.242	
Overall R^2	0.248		0.251		0.138		0.253	

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Note: Regression based on 1074455 observations

Table A1.5: Model B: Private Companies

	Regression 1		Regression 2		Regression 3		Regression 4	
	Est.	St.d	Est.	St.d	Est.	St.d	Est.	St.d
Market Share	0.369***	0.0295	-0.105***	0.0297	-0.195	0.288	-0.119***	0.0297
Total Assets			0.0268***	0.000209				
Output					0.125***	0.00205		
Operating Income							0.0225***	0.000197
Output Share of Assets	0.0705***	0.000270	0.0799***	0.000279	0.193***	0.00291	0.0654***	0.000284
Output Share of Capital	0.00666***	0.000160	0.00391***	0.000160	0.0385***	0.00152	0.00669***	0.000163
Output Share of Labor	0.0720***	0.000181	0.0690***	0.000182	0.299***	0.00173	0.0764***	0.000185
Operating Leverage	-0.000679	0.00134	-0.00582***	0.00134	-1.511***	0.0127	0.0186***	0.00136
Working Capital	0.0558***	0.000815	0.0607***	0.000812	-1.319***	0.00779	0.0639***	0.000816
Financial Leverage	-0.0812***	0.00117	-0.102***	0.00118	1.597***	0.0112	-0.0971***	0.00118
Liquidity Ratio	-0.00105***	0.0000456	-0.000939***	0.0000453	0.00187***	0.000428	-0.000364***	0.0000485
Age	-0.000854***	0.0000280	-0.00175***	0.0000289	-0.00804***	0.000292	-0.00150***	0.0000287
Constant	0.210***	0.00702	-0.172***	0.00764	-0.301***	0.0760	-0.120***	0.00810
Within R^2	0.267		0.283		0.0980		0.282	
Between R^2	0.242		0.239		0.112		0.242	
Overall R^2	0.249		0.251		0.141		0.254	

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Note: Regression based on 1035430 observations

Table A1.6: Model C: Public Companies

	Regression 1		Regression 2		Regression 3		Regression 4	
	Est.	St.d	Est.	St.d	Est.	St.d	Est.	St.d
Market Share	0.336***	0.0939	0.190*	0.0953	0.0451	1.105	0.151	0.0949
Total Assets			0.0137***	0.000932				
Output					0.132***	0.0111		
Operating Income							0.0131***	0.000908
Output Share of Assets	0.0567***	0.00134	0.0638***	0.00144	0.326***	0.0164	0.0545***	0.00138
Output Share of Capital	0.00490***	0.000867	0.00215*	0.000893	0.0365***	0.0100	0.00457***	0.000884
Output Share of Labor	0.111***	0.00118	0.109***	0.00119	0.692***	0.0132	0.114***	0.00120
Operating Leverage	0.140***	0.00738	0.135***	0.00742	-0.549***	0.0841	0.156***	0.00749
Working Capital	0.197***	0.00528	0.205***	0.00530	-0.348***	0.0597	0.205***	0.00531
Financial Leverage	-0.140***	0.00668	-0.157***	0.00677	1.345***	0.0766	-0.154***	0.00676
Liquidity Ratio	-0.00279***	0.000450	-0.00274***	0.000450	-0.0152**	0.00497	-0.00217***	0.000483
Age	-0.00160***	0.000120	-0.00230***	0.000131	-0.0126***	0.00161	-0.00219***	0.000127
Constant	0.116**	0.0410	-0.0759	0.0436	-1.060*	0.529	-0.0858	0.0453
Within R^2	0.335		0.345		0.111		0.346	
Between R^2	0.279		0.277		0.146		0.282	
Overall R^2	0.281		0.280		0.126		0.288	

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Note: Regression based on 39025 observations

Table A1.7: Model D: Non-listed Companies

	Regression 1		Regression 2		Regression 3		Regression 4	
	Est	St.d	Est.	St.d	Est.	St.d	Est.	St.d
Market Share	3.344***	0.951	2.664**	0.927	0.410	5.820	3.162**	0.983
Total Assets			0.0344***	0.00837				
Output					0.0914	0.0532		
Operating Income							0.00928	0.00621
Output Share of Assets	0.0574***	0.00933	0.0710***	0.00958	0.0492	0.0658	0.0604***	0.0115
Output Share of Capital	0.00409	0.00496	0.000387	0.00490	0.00315	0.0312	0.00961	0.00547
Output Share of Labor	0.0254***	0.00575	0.0150*	0.00610	0.0338	0.0385	0.0234***	0.00610
Operating Leverage	0.0398	0.0569	-0.00623	0.0558	-1.489***	0.352	0.0189	0.0630
Working Capital	0.00100	0.0557	0.00192	0.0537	-1.325***	0.339	-0.0125	0.0581
Financial Leverage	-0.0223	0.0477	-0.0185	0.0460	0.683*	0.290	-0.00764	0.0504
Liquidity Ratio	-0.000416	0.00162	-0.000449	0.00156	0.00520	0.00987	0.000660	0.00176
Age	0.0000226	0.000618	-0.000525	0.000620	-0.00189	0.00397	-0.000349	0.000627
Constant	-0.0323	0.144	0	.	-0.368	1.419	0	.

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Note: Regression based on 259 observations

Table A1.8: Model E: Listed Companies

	Regression 1		Regression 2		Regression 3		Regression 4	
	Est	St.d	Est.	St.d	Est.	St.d	Est.	St.d
Market Share	0.0931	0.0573	0.101	0.0599	0.0235	0.323	0.0675	0.0583
Total Assets			-0.00137	0.00355				
Output					-0.0332	0.0194		
Operating Income							0.00262	0.00196
Output Share of Assets	0.0545***	0.00342	0.0541***	0.00356	0.158***	0.0229	0.0532***	0.00389
Output Share of Capital	0.00563**	0.00184	0.00567**	0.00184	0.0340***	0.00984	0.00599**	0.00203
Output Share of Labor	0.0230***	0.00262	0.0234***	0.00282	0.0548***	0.0151	0.0240***	0.00282
Operating Leverage	-0.0290	0.0258	-0.0294	0.0257	-0.960***	0.137	-0.0249	0.0266
Working Capital	0.0140	0.0225	0.0130	0.0225	-0.700***	0.120	0.0305	0.0230
Financial Leverage	-0.0607**	0.0199	-0.0580**	0.0205	0.348**	0.110	-0.0660**	0.0209
Liquidity Ratio	-0.000461	0.000482	-0.000489	0.000487	-0.00282	0.00258	-0.000576	0.000553
Age	0.0000932	0.000165	0.000104	0.000165	0.000377	0.000930	0.0000872	0.000160
Constant	0.214***	0.0524	0.240**	0.0855	1.926***	0.471	0.163*	0.0647

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$
Note: Regression based on 1 088 observations

Table A1.9: Probit, with firm-fixed effext

	Pr(List = 1)	
	Est	St.d
Roa	-0.489	(0.518)
Revenue	0.0760	(0.0562)
CAPEX	0.0336	(0.0547)
Growth	0.119*	(0.0465)
Financial Leverage	-0.152	(0.339)
Output Share of Labor	-0.00000634	(0.00000885)
Output Share of Capital	0.000000200	(0.000000147)
Output Share of Assets	-0.784***	(0.215)
Age	-0.00197	(0.00556)

A1.3 Determinants of the Decision to Go Public

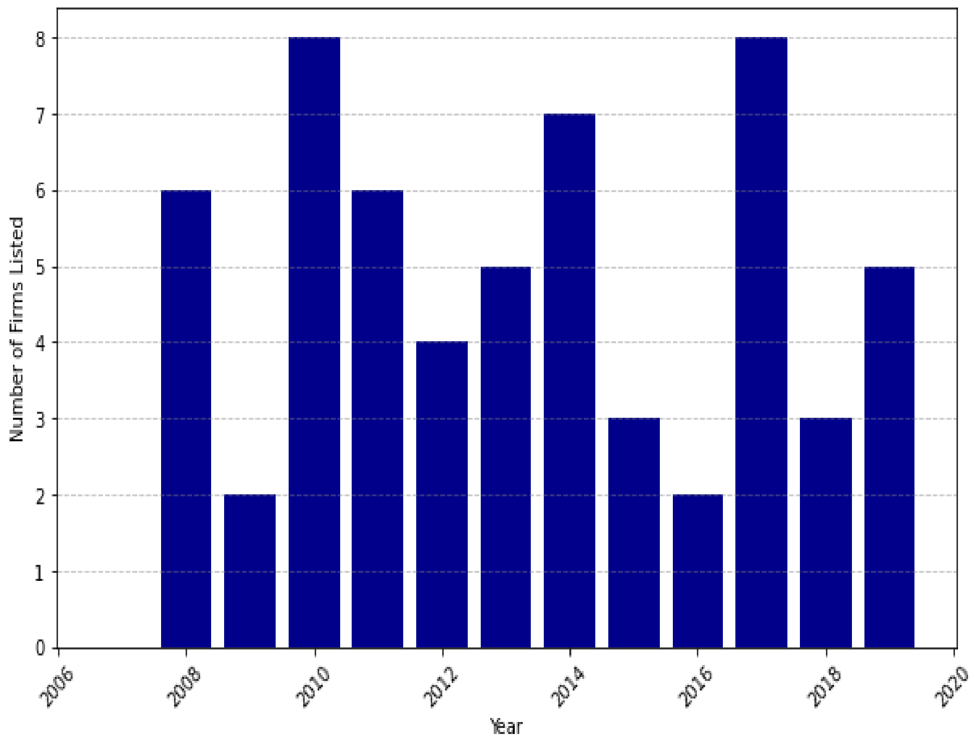

Figure A1.4: Number of New Listings

B1 Stata Codes

```
/// Empirical Analysis ///

use "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses - Copy\5_empirical_analysis\comovement.dta", clear

sort orgnr year


egen firm_id = group(orgnr)
xtset firm_id year

*** Generate profit_hat / output_win
gen pi_y_hat            = pi_hat_win / output_win
gen pi_y                = net_profit_win / output_win
winsor pi_y, p(.01) gen(profit_share)

*** Growth Rates
gen pi_y_g              = D.pi_y
gen pi_hat_g            = D.profit_share
gen log_labor_growth    = D.lg_emp_comp_win
gen log_capital_growth  = D.lg_capital_win

*** Regressions
* Growth Rate of Labor
eststo clear
eststo: qui xtreg log_labor_growth lg_output_growth if private == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_labor_growth lg_output_growth if public ==1, fe vce(cluster firm_id)
eststo: qui xtreg log_labor_growth lg_output_growth if public_nonlisted == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_labor_growth lg_output_growth if public_listed == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_labor_growth lg_output_growth, fe vce(cluster firm_id)
esttab, se keep(lg_output_growth) ///
mtitle ("Private Companies" "Public" "Public Nonlisted" "Public Listed" "Permanent Sample" )
*esttab using "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses - Copy\5_empirical_analysis\stata\logl.tex", se keep (lg_output_growth) ///

* Growth Rate of Capital
eststo clear
eststo: qui xtreg log_capital_growth lg_output_growth if private == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_capital_growth lg_output_growth if public == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_capital_growth lg_output_growth if public_nonlisted == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_capital_growth lg_output_growth if public_listed == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_capital_growth lg_output_growth, fe vce(cluster firm_id)

esttab, se keep(lg_output_growth) ///
mtitle ("Private Companies" "Public" "Public Nonlisted" "Public Listed" "Permanent Sample" )
*esttab using "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses - Copy\5_empirical_analysis\stata\logk.tex", se keep (lg_output_growth) ///

* Change in Profit Share
eststo clear
eststo: qui xtreg pi_y_g lg_output_growth if private == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_y_g lg_output_growth if public == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_y_g lg_output_growth if public_nonlisted == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_y_g lg_output_growth if public_listed == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_y_g lg_output_growth, fe vce(cluster firm_id)
esttab, se keep(lg_output_growth) ///
mtitle ("Private Companies" "Public" "Public Nonlisted" "Public Listed" "Permanent Sample" )
*esttab using "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses - Copy\5_empirical_analysis\stata\pi_y1.tex", se keep (lg_output_growth) ///

* Change in Profit Hat Share
eststo clear
eststo: qui xtreg pi_hat_g lg_output_growth if private == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_hat_g lg_output_growth if public == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_hat_g lg_output_growth if public_nonlisted == 1, fe vce(cluster firm_id)
```

```
eststo: qui xtreg pi_hat_g lg_output_growth if public_listed == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_hat_g lg_output_growth, fe vce(cluster firm_id)
esttab, se keep(lg_output_growth) ///
mtitle ("Private Companies" "Public" "Public Nonlisted" "Public Listed" "Permanent Sample" )
*esttab using "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses -
Copy\5_empirical_analysis\stata\\pi_hat1.tex", se keep (lg_output_growth) ///


/// Restrict:  |Δ log y| < 0.5

keep if abs(lg_output_growth) < 0.5

* Growth Rate of Labor
eststo clear
eststo: qui xtreg log_labor_growth lg_output_growth if private == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_labor_growth lg_output_growth if public ==1, fe vce(cluster firm_id)
eststo: qui xtreg log_labor_growth lg_output_growth if public_nonlisted == 1, fe vce(cluster
firm_id)
eststo: qui xtreg log_labor_growth lg_output_growth if public_listed == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_labor_growth lg_output_growth, fe vce(cluster firm_id)
esttab, se keep(lg_output_growth) ///
mtitle ("Private Companies" "Public" "Public Nonlisted" "Public Listed" "Permanent Sample" )
*esttab using "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses -
Copy\5_empirical_analysis\stata\logl2.tex", se keep (lg_output_growth) ///

* Growth Rate of Capital
eststo clear
eststo: qui xtreg log_capital_growth lg_output_growth if private == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_capital_growth lg_output_growth if public == 1, fe vce(cluster firm_id)
eststo: qui xtreg log_capital_growth lg_output_growth if public_nonlisted == 1, fe vce(cluster
firm_id)
eststo: qui xtreg log_capital_growth lg_output_growth if public_listed == 1, fe vce(cluster
firm_id)
eststo: qui xtreg log_capital_growth lg_output_growth, fe vce(cluster firm_id)

esttab, se keep(lg_output_growth) ///
mtitle ("Private Companies" "Public" "Public Nonlisted" "Public Listed" "Permanent Sample" )
*esttab using "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses -
Copy\5_empirical_analysis\stata\logk2.tex", se keep (lg_output_growth) ///

* Change in Profit Share
eststo clear
eststo: qui xtreg pi_y_g lg_output_growth if private == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_y_g lg_output_growth if public == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_y_g lg_output_growth if public_nonlisted == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_y_g lg_output_growth if public_listed == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_y_g lg_output_growth, fe vce(cluster firm_id)
esttab, se keep(lg_output_growth) ///
mtitle ("Private Companies" "Public" "Public Nonlisted" "Public Listed" "Permanent Sample" )
*esttab using "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses -
Copy\5_empirical_analysis\stata\pi_y12.tex", se keep (lg_output_growth) ///

* Change in Profit Hat Share
eststo clear
eststo: qui xtreg pi_hat_g lg_output_growth if private == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_hat_g lg_output_growth if public == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_hat_g lg_output_growth if public_nonlisted == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_hat_g lg_output_growth if public_listed == 1, fe vce(cluster firm_id)
eststo: qui xtreg pi_hat_g lg_output_growth, fe vce(cluster firm_id)
esttab, se keep(lg_output_growth) ///
mtitle ("Private Companies" "Public" "Public Nonlisted" "Public Listed" "Permanent Sample" )
*esttab using "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses -
Copy\5_empirical_analysis\stata\\pi_hat12.tex", se keep (lg_output_growth) ///


/// ROA PERSISTENCE ///

use "N:\durable\BIStudents\IssamSheikh\codes\Duplicates\5_analyses -
```

```
Copy\5_empirical_analysis\roa_persistence.dta", clear

egen firm_id = group(orgnr)
xtset firm_id year
sort firm_id year

gen roa_lag = L.ROA_win
regress ROA_win roa_lag if private == 1
regress ROA_win roa_lag if private == 0

xtreg ROA_win roa_lag if private == 1
xtreg ROA_win roa_lag if private == 0


/// Determinants of Returns ///
//// Profitability regressions ////
clear
cls

/// Read the dataset ///
use "N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_full.dta", clear

// Fix orgnr variables and set panel structure variables
egen org_ID = group(orgnr)
xtset org_ID year

gen public_listed = public - public_nonlisted

// Summary statistics for variables of interest

sort private
by private: summarize ROA_win age mrkt_share lg_total_assets lg_op_inc lg_output
operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win lg_output_a
lg_output_k lg_output_l


by private: pwcorr ROA_win age mrkt_share lg_total_assets lg_op_inc lg_output
operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win lg_output_a
lg_output_k lg_output_l


//// Regression 1.1 - All firms ////
eststo: xtreg ROA_win mrkt_share lg_output_a lg_output_k lg_output_l operating_leverage_win
working_capital financial_leverage_win liquidity_ratio_win  age i.year i.industry
eststo: xtreg ROA_win mrkt_share lg_total_assets lg_output_a lg_output_k lg_output_l
operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win  age i.year i.
industry
eststo: xtreg ROE_win mrkt_share lg_output lg_output_a lg_output_k lg_output_l
operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.
industry
eststo: xtreg ROA_win mrkt_share lg_op_inc lg_output_a lg_output_k lg_output_l
operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.
industry



//// Regression 1.2 - Private firms ////
eststo: xtreg ROA_win mrkt_share lg_output_a lg_output_k lg_output_l operating_leverage_win
working_capital financial_leverage_win liquidity_ratio_win  age i.year i.industry if private == 1
eststo: xtreg ROA_win mrkt_share lg_total_assets lg_output_a lg_output_k lg_output_l
operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win  age i.year i.
industry if private == 1
eststo: xtreg ROE_win mrkt_share lg_output lg_output_a lg_output_k lg_output_l
operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.
industry if private == 1
eststo: xtreg ROA_win mrkt_share lg_op_inc lg_output_a lg_output_k lg_output_l
operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.
industry if private == 1
```

```


//// Regression 1.3 - Public firms////
eststo: xtreg ROA_win mrkt_share lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win  age i.year i.industry if private == 0
eststo: xtreg ROA_win mrkt_share lg_total_assets lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win  age i.year i.industry if private == 0
eststo: xtreg ROE_win mrkt_share lg_output lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.industry if private == 0
eststo: xtreg ROA_win mrkt_share lg_op_inc lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.industry if private == 0


//// Regression 1.4 - Public Nonlisted ////
eststo clear
eststo: qui xtreg ROA_win mrkt_share lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win  age i.year i.industry if public_nonlisted == 1
eststo: qui xtreg ROA_win mrkt_share lg_total_assets lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win  age i.year i.industry if public_nonlisted == 1
eststo: qui xtreg ROE_win mrkt_share lg_output lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.industry if public_nonlisted == 1
eststo: qui xtreg ROA_win mrkt_share lg_op_inc lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.industry if public_nonlisted == 1
esttab, wide se drop(year, industry,)
esttab using "N:\durable\file-export\IssamSheikh\S2_profitability\nonlisted1.tex", wide se drop(i.year i.industry)


//// Regression 1.5 - Public Listed ////
eststo clear
eststo: qui xtreg ROA_win mrkt_share lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win  age i.year i.industry if public_listed == 1
eststo: qui xtreg ROA_win mrkt_share lg_total_assets lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win  age i.year i.industry if public_listed == 1
eststo: qui xtreg ROE_win mrkt_share lg_output lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.industry if public_listed == 1
eststo: qui xtreg ROA_win mrkt_share lg_op_inc lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age i.year i.industry if public_listed == 1
esttab, wide se keep(ROA_win mrkt_share lg_total_assets lg_op_inc lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age)
esttab using "N:\durable\file-export\IssamSheikh\S2_profitability\listed1.tex", wide se keep(ROA_win mrkt_share lg_total_assets lg_op_inc lg_output_a lg_output_k lg_output_l operating_leverage_win working_capital financial_leverage_win liquidity_ratio_win age), replace


/// PROBIT ///

clear
cls

/// Read the dataset ///
use "N:/durable/BIStudents/IssamSheikh/data/clean/reg2/reg2_full.dta"

// Fix orgnr variables and set panel structure variables
egen org_ID = group(orgnr)
xtset org_ID year

```

```
//// Probit ////
eststo: xtprobit to_listed L.ROA_win L.lg_rev L.lg_capex growth_win L.financial_leverage output_l
output_k age i.year i.industry

```

B2 Python Codes

In [ ]:
```
pip install nbconvert
```

In [ ]:
```
# Install needed libraries
import pandas as pd
import pyarrow as pa
import pyarrow.parquet as pq
import nbconvert
```

In [ ]:
```
## Building variables dictionary
# Read the Excel file into a Pandas DataFrame
df = pd.read_excel('N:/durable/BIStudents/IssamSheikh/data/variables/ \
    var_dict \- Copy.xlsx')

# Convert the DataFrame to dictionaries
variable_map = dict(zip(df['old_var'], df['new_var']))
eng_def_map = dict(zip(df['old_var'], df['def_eng']))
no_def_map = dict(zip(df['old_var'], df['def_no']))
long_def_map = dict(zip(df['old_var'], df['def_long']))
```

In [ ]:
```
# Defining a dictionary application
def get_variable_info(variable_name, language, long=False):
    # Determine if variable_name is the old or new variable name
    if variable_name in variable_map:
        old_var = variable_name
        new_var = variable_map[old_var]
    elif variable_name in set(variable_map.values()):
        new_var = variable_name
        old_var = list(variable_map.keys())[list(variable_map.values())\
            .index(new_var)]
    else:
        raise ValueError("Invalid variable name")

    # Determine the definition based on the specified language
    if language == 'eng':
        definition = eng_def_map[old_var]
        if long:
            long_definition = long_def_map[old_var]
    elif language == 'nor':
        definition = no_def_map[old_var]
        if long:
            long_definition = no_long_def_map[old_var]
    else:
        raise ValueError("Invalid language")

    if long:
        return new_var, definition, long_definition
    else:
        return new_var, definition
```

In [ ]:
```
# Example
get_variable_info("orgnr", "nor", long=False)
```

```
In [ ]: # Accounting firms data (Regnskap-yearly)
        # Creating a yearly accounting datasets
        # Give a name to path
        folder_raw = 'N:/durable/BIStudents/IssamSheikh/data/raw/regn_end/'

        # Reading the datasets from all the years and naming them regn_end_year
        years = range(2005, 2020)
        for year in years:
            file_name = f'{folder_raw}W19_1210_REGN_END_{year}.dta'
            df_name = f'regn_end_{year}'
            globals()[df_name] = pd.read_stata(file_name).rename(columns=variable_map)
            globals()[df_name].to_parquet(f'{folder_raw}regn_end_{year}.parquet',\
                 index=False)
```

```
In [ ]: # Shareholder registry data (Aksjeselskaper-yearly)
        # Creating a yearly shareholder registry datasets
        # Give a name to path
        folder_raw = 'N:/durable/BIStudents/IssamSheikh/data/raw/aksjeselskaper/'

        # Reading the datasets from all the years and naming them akjseselskaper_year
        years = range(2004, 2020)
        for year in years:
            file_name = f'{folder_raw}W19_1210_AKSJESELSKAPER_{year}.dta'
            df_name = f'aksjeselskaper_{year}'
            globals()[df_name] = pd.read_stata(file_name).rename(columns=variable_map)
            globals()[df_name].to_parquet(f'{folder_raw}aksjeselskaper_{year}.parquet', \
                index=False)
```

```
In [ ]: # Reading latest VOF data (202112)

        df = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer\
            /W19_1210_VOF_FORETAK_202112.dta')
        df = df.rename(columns=variable_map)

        # Save pandas dataframe as parquet file
        # Convert datafram to Apach arrow table
        table = pa.Table.from_pandas(df)

        # Write table to a parquet file with Brotli compression
        output_file = 'N:/durable/BIStudents/IssamSheikh/data/raw/vof_fore/VOF_\
            202112.parquet'
        pq.write_table(table, output_file)
```

```
In [ ]: # Panel creation: Accounting firms data (Regnskap)
        # Give a name to paths
        folder_raw = 'N:/durable/BIStudents/IssamSheikh/data/raw/regn_end/'

        # Creating a panel from datasets
        years = range(2005, 2020)
        df_name = [pd.read_stata(f'{folder_raw}W19_1210_REGN_END_{yr}.dta') for \
            yr in years]

        # Assign a year identifier to each dataset
        for i, panel_data in enumerate(df_name):
```

```
    panel_data['year'] = years[i]

# Concatenate the datasets vertically and setting a hierarchical index
panel_data = pd.concat(df_name) #.set_index(['year','w19_1210_lopenr_foretak'])

# Rename variables using dictionary
panel_data = panel_data.rename(columns=variable_map)

# Save pandas dataframe as parquet file
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(panel_data)

# Write table to a parquet file with Brotli compression
output_file = 'N:/durable/BIStudents/IssamSheikh/data/raw/regn_end/regn_end_\
    all.parquet'
pq.write_table(table, output_file)
```

In [ ]:

```
# Panel creation: shareholder registry (Aksjeselskaper)
# Give a name to paths
folder_raw = 'N:/durable/BIStudents/IssamSheikh/data/raw/aksjeselskaper/'

# Creating a panel from datasets
years = range(2004, 2020)
df_name = [pd.read_stata(f'{folder_raw}W19_1210_AKSJESELSKAPER_{yr}.dta') \
    for yr in years]

# Assign a year identifier to each dataset
for i, panel_data_1 in enumerate(df_name):
    panel_data_1['year'] = years[i]

# Concatenate the datasets vertically and setting a hierarchical index
panel_data_1 = pd.concat(df_name) #.set_index(['year','w19_1210_lopenr_foretak'])

# Rename variables using dictionary
panel_data_1 = panel_data_1.rename(columns=variable_map)

# Save pandas dataframe as parquet file
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(panel_data_1)

# Write table to a parquet file with Brotli compression
output_file = 'N:/durable/BIStudents/IssamSheikh/data/raw/aksjeselskaper\
    /aksjeselskaper_all.parquet'
pq.write_table(table, output_file)
```

In [ ]:

```
# Read VOF datasets
df05 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_200512.dta')
df06 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_200612.dta')
df07 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_200712.dta')
df08 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_200812.dta')
df09 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_200912.dta')
df10 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_201012.dta')
df11 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer\
    /W19_1210_VOF_FORETAK_201112.dta')
df12 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_201212.dta')
df13 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_201312.dta')
df14 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_201412.dta')
df15 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_201512.dta')
df16 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_201612.dta')
df17 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_201712.dta')
df18 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_201812.dta')
df19 = pd.read_stata('N:/durable/LaborMarketOutcomes/Data/Backup/STATA/VOF_aksjer/\
    W19_1210_VOF_FORETAK_201912.dta')
```

```
In [ ]: # Keep only orgnr and industry
df05 = df05[['w19_1210_lopenr_orgnr', 'naering1']]
df06 = df06[['w19_1210_lopenr_orgnr', 'naering1']]
df07 = df07[['w19_1210_lopenr_orgnr', 'naering1']]
df08 = df08[['w19_1210_lopenr_orgnr', 'naering1']]
df09 = df09[['w19_1210_lopenr_orgnr', 'naering1']]
df10 = df10[['w19_1210_lopenr_orgnr', 'naering1']]
df11 = df11[['w19_1210_lopenr_orgnr', 'naering1']]
df12 = df12[['w19_1210_lopenr_orgnr', 'naering1']]
df13 = df13[['w19_1210_lopenr_orgnr', 'naering1']]
df14 = df14[['w19_1210_lopenr_orgnr', 'naering1']]
df15 = df15[['w19_1210_lopenr_orgnr', 'naering1']]
df16 = df16[['w19_1210_lopenr_orgnr', 'naering1']]
df17 = df17[['w19_1210_lopenr_orgnr', 'naering1']]
df18 = df18[['w19_1210_lopenr_orgnr', 'naering1']]
df19 = df19[['w19_1210_lopenr_orgnr', 'naering1']]

# Concate dataframes
dfs = [df05, df06, df07, df08, df09, df10, df11, df12, df13, df14, \
    df15, df16, df17, df18, df19]
df = pd.concat(dfs)
print('Number of total observations:    ',  df.shape[0])

# Drop rows with empy industry
df_1 = df[df['naering1'] != '']
print('Number of droper rows in step 1:  ',df.shape[0] - df_1.shape[0])

# Remove duplicated rows
df_2 = df_1.drop_duplicates(subset='w19_1210_lopenr_orgnr', keep='last')
print('Number of droper rows in step 2: ',df_1.shape[0] - df_2.shape[0])

# Rename variables using dictionary
df_2 = df_2.rename(columns=variable_map)
print('Number of rows in final df:       ', df_2.shape[0])
print('Number of unique variables: ')
df_2.nunique()

# Save pandas dataframe as parquet file
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(df_2)

# Write table to a parquet file with Brotli compression
output_file = 'N:/durable/BIStudents/IssamSheikh/data/raw/vof_fore\
    /industry_all.parquet'
pq.write_table(table, output_file)
```

In [ ]:
```
# Creating parquets of VOF_FORETAK_201912 and VOF_VIRKSOMHET_201912
df_vof_foretak              = pd.read_stata('N:/durable/LaborMarketOutcomes/Data\
    /Backup/STATA/VOF_aksjer/W19_1210_VOF_FORETAK_201912.dta')
df_vof_foretak              = df_vof_foretak.rename(columns=variable_map)
df_vof_virksomhet           = pd.read_stata('N:/durable/LaborMarketOutcomes/Data\
    /Backup/STATA/VOF_aksjer/W19_1210_VOF_VIRKSOMHET_201912.dta')
df_vof_virksomhet           = df_vof_virksomhet.rename(columns=variable_map)

# Save pandas dataframe as parquet file
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(df_vof_foretak)

# Write table to a parquet file with Brotli compression
output_file = 'N:/durable/BIStudents/IssamSheikh/data/raw/vof_fore/VOF_FORETAK\
    _201912.parquet'
pq.write_table(table, output_file)

table1 = pa.Table.from_pandas(df_vof_virksomhet)

# Write table to a parquet file with Brotli compression
output_file = 'N:/durable/BIStudents/IssamSheikh/data/raw/vof_fore/VOF_\
    VIRKSOMHET_201912.parquet'
pq.write_table(table1, output_file)
```

In [ ]:

In [ ]:
```

```

In [ ]:
```
#import needed libraries
import pandas as pd
```

In [ ]:
```
# Give names to path
folder_raw = 'N:/durable/BIStudents/IssamSheikh/data/raw/regn_end/'

#reading the datasets from all the years and naming them regn_year
years = range(2005, 2020)
for year in years:
    file_name = f"{folder_raw}regn_end_{year}.parquet"
    df_name = f"regn_{year}"
    globals()[df_name] = pd.read_parquet(file_name)
    globals()[df_name] = globals()[df_name].rename(columns={'w19_1210_lopenr\
        _foretak': 'oregnnr'})
#after running this code we can use: regn_2005 for example to call the dataset from
```

In [ ]:
```
#give names to paths
folder_variables = 'N:/durable/BIStudents/IssamSheikh/data/variables/'

# Load the list of variables from the Brønnøysundregistrene csv file
variables_BR = pd.read_csv(folder_variables + 'Brønnøysundregistrene_\
    variables_list.csv')

# Convert the csv file into a list
variables_BR = variables_BR['Variable'].tolist()

# Find the missing variables in the 2005 dataframe
missing_variables = [var for var in variables_BR if var not in regn_2005.columns]
non_missing_variables = [var for var in variables_BR if var in regn_2005.columns]

# Get the number of missing variables
num_missing = len(missing_variables)
non_missing = len(variables_BR) - num_missing

#print(f"Number of missing variables: {num_missing}")
#print(f"Missing variables: {missing_variables}")
print(f"Number of non missing variables: {non_missing}")
print(f"Non-missing variables: {non_missing_variables}")
```

In [ ]:
```
# Give names to paths
folder_variables = 'N:/durable/BIStudents/IssamSheikh/data/variables/'

# Load the list of variables from the Skatteetaten csv file
variables_SK = pd.read_csv(folder_variables + 'aksjeselskaper_variables_list.csv')

# Convert the csv file into a list
variables_SK = variables_SK['Variable'].tolist()

# Find the missing variables in the 2005 dataframe
missing_variables = [var for var in variables_SK if var not in regn_2005.columns]
non_missing_variables = [var for var in variables_SK if var in regn_2005.columns]
```

```
# Get the number of missing variables3
num_missing = len(missing_variables)
non_missing = len(variables_SK) - num_missing

#print(f"Number of missing variables: {num_missing}")
#print(f"Missing variables: {missing_variables}")
print(f"Number of non missing variables: {non_missing}")
#print(f"Non-missing variables: {non_missing_variables}")
```

In [ ]:
```
# Comparing datasets
def compare_dataframes(df1, df2):
    # Get the set of variables in each data frame
    set1 = set(df1.columns)
    set2 = set(df2.columns)
    # Get the set of variables that are different
    diff = set1.symmetric_difference(set2)
    return diff

# Loop over all the years and compare the variables in each data frame to \
# the variables in the next one
for i in range(2005, 2017):
    df1_name = f"regn_{i}"
    df2_name = f"regn_{i + 1}"
    df1 = globals()[df1_name]
    df2 = globals()[df2_name]
    diff = compare_dataframes(df1, df2)
    if len(diff) > 0:
        print(f"Variables missing in {df1_name} compared to {df2_name}: {diff}")
    else:
        print(f"{df1_name} and {df2_name} have the same variables")
```

In [ ]:
```
# Counting the number of variables from Skatteetaten &
# Brønnøysundregistrene and the numbers of variables in the dataframes
print('Lenght of the variables list in the dataframe:' + \
    str(len(regn_2005.columns)))
print('Lenght of the variables list (Skatteetaten):' + \
    str(len(variables_SK)))
print('Lenght of the variables list (Brønnøysundregistrene):' +\
     str(len(variables_BR)))
```

In [ ]:
```
# Calculating & comparing operating income
operating_income = regn_2005.salgsinnt + regn_2005.p3300 + regn_2005.p3400 \
    + regn_2005.leieinnt + regn_2005.gev_anl + regn_2005.andreinnt

# Check if the sum of returns and returns equal
result = operating_income == regn_2005.p9000
print("Number of False values in acctual and calculated: ", (result == False).sum()
```

In [ ]:
```
# Calculating & comparing total costs
total_costs = (regn_2005.varekost + regn_2005.p4295 + regn_2005.lonn \
    + regn_2005.avskriv + regn_2005.ovrkost)

# check if the sum of costs and returns is equal to profit
```

```
result = total_costs == regn_2005.p9010
print("Number of False values in acctual and calculated: ", (result == False).sum()
```

In [ ]:
```
# Calculating & comparing salaries
salaries = (regn_2005.p5000 + regn_2005.p5400 + \
    regn_2005.personal)

result = salaries == regn_2005.lonn
print("Number of False values in acctual and calculated: ", (result == False).sum()
```

In [ ]:
```
# Calculating & comparing other costs
salaries = ( regn_2005.p4500 + regn_2005.p6100 + regn_2005.p6200 +\
     regn_2005.lokale + regn_2005.matriell + regn_2005.repved +\
         regn_2005.p6700 +regn_2005.bilkost + regn_2005.reise + \
            regn_2005.reklame + regn_2005.tap_anl + regn_2005.p7830 +\
                 regn_2005.andrdrkost)

result = salaries == regn_2005.ovrkost
print("Number of False values in acctual and calculated: ", (result == False).sum()
```

In [ ]:
```
#give names to paths
folder_raw = 'N:/durable/BIStudents/IssamSheikh/data/raw/aksjeselskaper/'

# Reading the datasets from all the years and naming them aksje_year
years = range(2004, 2020)
for year in years:
    file_name = f"{folder_raw}aksjeselskaper_{year}.parquet"
    df_name = f"aksje_{year}"
    globals()[df_name] = pd.read_parquet(file_name)
    #globals()[df_name] = globals()[df_name].columns.str.strip()
#after running this code we can use: aksje_2005 for example to
# call the dataset from 2005...
```

In [ ]:
```
# Load the list of variables from the akjeseleskaper variables csv file
variables_akje = pd.read_csv(folder_variables + \
    'aksjeselskaper_variables_list.csv')

# Convert the csv file into a list
variables_akje = variables_akje['Variable'].tolist()

# Find the missing variables in the 2005 dataframe
missing_variables = [var for var in variables_akje if\
     var not in aksje_2015.columns]
non_missing_variables = [var for var in variables_akje \
    if var in aksje_2015.columns]

# Get the number of missing variables
num_missing = len(missing_variables)
non_missing = len(variables_akje) - num_missing

print(f"Number of missing variables: {num_missing}")
#print(f"Missing variables: {missing_variables}")
print(f"Number of non missing variables: {non_missing}")
print(f"Non-missing variables: {non_missing_variables}")
```

```
In [ ]: # Comparing datasets
        def compare_dataframes(df1, df2):
            # Get the set of variables in each data frame
            set1 = set(df1.columns)
            set2 = set(df2.columns)
            # Get the set of variables that are different
            diff = set1.symmetric_difference(set2)
            return diff

        # Loop over all the years and compare the variables in
        # each data frame to the variables in the next one
        for i in range(2005, 2017):
            df1_name = f"aksje_{i}"
            df2_name = f"aksje_{i + 1}"
            df1 = globals()[df1_name]
            df2 = globals()[df2_name]
            diff = compare_dataframes(df1, df2)
            if len(diff) > 0:
                print(f"Variables missing in {df1_name} compared to {df2_name}: {diff}")
            else:
                print(f"{df1_name} and {df2_name} have the same variables")
```

```
In [ ]: # Counting the number of variables from akjeseleskaper and the numbers of variables
        print('Lenght of the variables list in the dataframe:' + \
            str(len(aksje_2005.columns)))
        print('Lenght of the variables list (akjeseleskaper):' + \
            str(len(variables_akje)))
```

In [ ]:

```

```

In [ ]:

```python
# Install needed libraries
import pandas as pd
import pyarrow as pa
import pyarrow.parquet as pq
```

In [ ]:

```python
## Building variables dictionary
# Read the Excel file into a Pandas DataFrame
df = pd.read_excel('N:/durable/BIStudents/IssamSheikh/data/variables/\
    var_dict - Copy.xlsx')

# Convert the DataFrame to dictionaries
variable_map = dict(zip(df['old_var'], df['new_var']))
eng_def_map = dict(zip(df['old_var'], df['def_eng']))
no_def_map = dict(zip(df['old_var'], df['def_no']))
long_def_map = dict(zip(df['old_var'], df['def_long']))
```

In [ ]:

```python
def get_variable_info(variable_name, language, long=False):
    # Determine if variable_name is the old or new variable name
    if variable_name in variable_map:
        old_var = variable_name
        new_var = variable_map[old_var]
    elif variable_name in set(variable_map.values()):
        new_var = variable_name
        old_var = list(variable_map.keys())[list(variable_map.values()).\
            index(new_var)]
    else:
        raise ValueError("Invalid variable name")

    # Determine the definition based on the specified language
    if language == 'eng':
        definition = eng_def_map[old_var]
        if long:
            long_definition = long_def_map[old_var]
    elif language == 'nor':
        definition = no_def_map[old_var]
        if long:
            long_definition = no_long_def_map[old_var]
    else:
        raise ValueError("Invalid language")

    if long:
        return new_var, definition, long_definition
    else:
        return new_var, definition
```

In [ ]:

```python
# Example
get_variable_info("reg_stat", "eng", long=True)
```

In [ ]:

```

```

In [ ]:
```
pip install nbconvert
```

In [ ]:
```
# Import needed libraries
import pandas as pd
import numpy as np
import pyarrow.parquet as pq
pd.set_option('mode.use_inf_as_na', True)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline
from tabulate import tabulate
from scipy.stats.mstats import winsorize
import pyarrow as pa

# Display 6 columns for viewing pupuses
pd.set_option('display.max_columns', 10)
# Reduce decimals points to 2
pd.options.display.float_format = '{:,.2f}'.format
```

In [ ]:
```
# Load and Read Data Parquet
df = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/raw\
    /regn_end/regn_end_all.parquet')

# Counting number of rows before cleaning
n_row_before = len(df)
print('Number of rows in df (Original): ' + str(n_row_before))
```

Basic Cleaning

In [ ]:
```
# Step 1: Drop rows with missing variables & duplicates on year and orgnr
# df.dropna(inplace=True)
df.drop_duplicates(subset=['year', 'orgnr'], keep='last', inplace=True)

# Counting number of rows before droping rows with missing variables
n_row_after_1 = df.shape[0]
print('Number of rows in df (Original):          ' + \
     str(n_row_before))
print('Number of rows of the dataframe (New):    ' + \
    str(n_row_after_1))
print('Number of dropped rows:                   ' + \
    str(n_row_before - n_row_after_1))
```

In [ ]:
```
# Step 2: Check for reporting inconsistencies
# Step 2.1: Drop rows where total assets are missing or negative
df = df[df['total_assets']>=0]
n_row_after_2 = df.shape[0]
print('Number of rows in df (Original):          ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):    ' + \
    str(n_row_after_2))
```

```
print('Number of dropped rows:                   ' + \
    str(n_row_after_1 - n_row_after_2))
```

In [ ]:
```
# Step 2.2: Drop rows where employment is missing or negative:
df = df[df['emp_comp'] > 0]
n_row_after_3 = df.shape[0]
print('Number of rows in df (Original):          ' +\
     str(n_row_before))
print('Number of rows of the dataframe (New):    ' + \
    str(n_row_after_3))
print('Number of dropped rows:                   ' + \
    str(n_row_after_2 - n_row_after_3))
```

In [ ]:
```
# Step 2.3: Drop rows where sales are missing or negative:
df = df[df['rev'] >= 0]
n_row_after_4 = df.shape[0]
print('Number of rows in df (Original):          ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):    ' + \
    str(n_row_after_4))
print('Number of dropped rows:                   ' + \
    str(n_row_after_3 - n_row_after_4))
```

In [ ]:
```
# Step 2.4: Drop rows where fixed assets are missing or negative:
df = df[df['tan_fixed_assets'] >= 0]
df = df[df['int_assets'] >= 0]
df = df[df['tot_fixe_assets'] >= 0]
n_row_after_5 = df.shape[0]
print('Number of rows in df (Original):          ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):    ' + \
    str(n_row_after_5))
print('Number of dropped rows:                      ' + \
    str(n_row_after_4 - n_row_after_5))
```

In [ ]:
```
df['tot_fixe_assets'].describe()
```

In [ ]:
```
# Step 2.5: Drop rows where values of materials (COGS) are
#missing or negative
df = df[df['cogs'] >= 0]
n_row_after_6 = df.shape[0]
print('Number of rows in df (Original):          ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):    ' + \
    str(n_row_after_6))
print('Number of dropped rows:                      ' + \
    str(n_row_after_5 - n_row_after_6))
```

In [ ]:
```
# Step 2.6: Drop rows where operating revenues are
#missing or negative
df = df[df['op_inc'] >= 0]

n_row_after_7 = df.shape[0]
```

```
print('Number of rows in df (Original):         ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):   ' + \
    str(n_row_after_7))
print('Number of dropped rows:                      ' + \
    str(n_row_after_6 - n_row_after_7))
```

```
In [ ]: # Step 2.6: Drop rows where long_term liabilities are missing or negative
        df = df[df['lt_liab'] >= 0]

        n_row_after_8 = df.shape[0]

        print('Number of rows in df (Original):         ' + \
            str(n_row_before))
        print('Number of rows of the dataframe (New):   ' + \
            str(n_row_after_8))
        print('Number of dropped rows:                     ' + \
            str(n_row_after_7 - n_row_after_8))
```

Balance sheet internal inconsistency

```
In [ ]: # Step 3: Check for Reporting Inconsistencies
        # Step 3.1: Checking if the Fixed Assets Add Up
        # PP&E + Tangible Fixed Assets + Capital Assets - Total Fixed Assets
        df['fa_diff'] = (df['int_assets'] + df['tan_fixed_assets'] +\
             df['cap_assets']) - df['tot_fixe_assets']

        # Step 3.1.1: Dropping Rows with Fixed Assets Reporting Inconsistencies
        df = df[(df['fa_diff'] == 0)]

        # Step 3.1.2 Deleting the "fa_ratio" Column
        df = df.drop('fa_diff', axis=1)

        # Counting nr of Dropped Rows:
        n_row_after_9 = df.shape[0]
        print('Number of rows in df (Original):         ' + \
            str(n_row_before))
        print('Number of rows of the dataframe (New):   ' + \
            str(n_row_after_9))
        print('Number of dropped rows:                     ' + \
            str(n_row_after_8 - n_row_after_9))
```

```
In [ ]: # Step 3.2: Checking if the Current Assets Add Up
            # (Inventories + Claims + Securities and Other
            # Financial Current Assets + Cash and Bank Deposists) - Total Current Assets
        df['ca_diff'] = (df['inventory'] + df['claims'] + df['securities']\
             + df['cash']) - df['tot_curr_assets']

        # Step 3.2.1: Dropping Rows with Current Assets Reporting Inconsistencies
        df = df[(df['ca_diff'] == 0)]

        # Step 3.2.2: Deleting the "ca_diff" Column
        df = df.drop('ca_diff', axis = 1)
```

```
# Counting nr of Dropped Rows:
n_row_after_10 = df.shape[0]
print('Number of rows in df (Original):         ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):   ' + \
    str(n_row_after_10))
print('Number of dropped rows:                        '\
     + str(n_row_after_9 - n_row_after_10))
```

```
In [ ]: # Step 3.3: Checking if the Assets Add Up
            # (Inventories + Claims + Securities and Other Financial Current Assets + Cash
        df['a_diff'] = (df['tot_fixe_assets'] + df['tot_curr_assets']) -\
             df['total_assets']

        # Step 3.3.1: Dropping Rows with Assets Reporting Inconsistencies
        df = df[(df['a_diff'] == 0)]

        # Step 3.3.2: Deleting the "a_diff" Column
        df = df.drop('a_diff', axis = 1)

        # Counting nr of Dropped Rows:
        n_row_after_11 = df.shape[0]
        print('Number of rows in df (Original):         ' + \
            str(n_row_before))
        print('Number of rows of the dataframe (New):   ' + \
            str(n_row_after_11))
        print('Number of dropped rows:                        ' \
            + str(n_row_after_10 - n_row_after_11))
```

```
In [ ]: # Step 3.4: Checking if the Long Term Liabilities Add Up
            # (Convertible Loans and Bond Loans + Long-term Debt
            # to Credit Institutions + Debt to Businesses in the Same Enterprise +
            #  Other Long-term Liabilitites) - Long-term Liabilities
        df['ltl_diff'] = (df['con_loan_bn'] + df['lon_debt_ins'] + \
            df['gje_til_tid'] + df['oth_lt_liab']) - df['lt_liab']

        # Step 3.4.1: Dropping Rows with Long-term Liabilities Reporting Inconsistencies
        df = df[(df['ltl_diff'] == 0)]

        # Step 3.4.2: Deleting the "ltl_diff" Column
        df = df.drop('ltl_diff', axis = 1 )

        # Counting nr of Dropped Rows:
        n_row_after_12= df.shape[0]
        print('Number of rows in df (Original):         ' + \
            str(n_row_before))
        print('Number of rows of the dataframe (New):   ' + \
            str(n_row_after_12))
        print('Number of dropped rows:                        ' + \
            str(n_row_after_11 - n_row_after_12))
```

```
In [ ]: # Step 3.5: Checking if the Current Liabilities Add Up
            # (Convertible Loans and Bond Loans + Overdraft + Trade Creditors +
            # Payable Tax + Proposed Dividends + Advance From Customers
```

```
    # + Debt to Companies in the Same Group + Salary Due, Holiday Pay etc. +
    #  Other Short Term Liabilites) - Current Liabilites
df['cl_diff'] = (df['con_loan_cl'] + df['overdraft'] + df['acc_pay'] + \
    df['tax_pay'] +
                    df['pro_div_liab'] + df['adv_from_cus'] + \
                        df['deb_to_grp'] +
                    df['sal_pay'] + df['oth_st_liab']) - df['curr_liab']

# Step 3.5.1: Dropping Rows with Current Liabilities
#  Reporting Inconsistencies
df = df[(df['cl_diff'] == 0)]

# Step 3.5.2: Deleting the "cl_diff" Column
df = df.drop('cl_diff', axis = 1)

# Counting nr of Dropped Rows:
n_row_after_13 = df.shape[0]
print('Number of rows in df (Original):          ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):    ' +\
     str(n_row_after_13))
print('Number of dropped rows:                      ' + \
    str(n_row_after_12 - n_row_after_13))
```

In [ ]:
```
# Step 3.6: Checking if the Shareholders Equity Add Up
    # Deposit Capital + Retained Earnings - Equity
df['eq_diff'] = (df['dep_cap'] + df['ret_earn']) - \
    df['equity']

# Step 3.6.1: Dropping Rows with Current Assets \
# Reporting Inconsistencies
df = df[(df['eq_diff'] == 0)]

# Step 3.6.2: Deleting the "eq_diff" Column
df = df.drop('eq_diff', axis = 1)

# Counting nr of Dropped Rows:
n_row_after_14 = df.shape[0]
print('Number of rows in df (Original):          ' +\
     str(n_row_before))
print('Number of rows of the dataframe (New):    ' +\
     str(n_row_after_14))
print('Number of dropped rows:                     ' + \
    str(n_row_after_13 - n_row_after_14))
```

In [ ]:
```
# Step 3.7: Checking if the Shareholders Equity and Liabilities Add Up
    # Equity + Liabilities - Total Equity and Liabilities
df['eqli_diff'] = (df['equity'] + df['liab'] - df['total_equi_lia'])

# Step 3.7.1: Dropping Rows with Reporting Inconsistencies
df = df[(df['eqli_diff'] == 0)]

# Step 3.7.2: Deleting the "eqli_diff" Column
df = df.drop('eqli_diff', axis = 1)
```

```
# Counting nr of Dropped Rows:
n_row_after_15 = df.shape[0]
print('Number of rows in df (Original):          ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):    ' + \
    str(n_row_after_15))
print('Number of dropped rows:                        ' \
    + str(n_row_after_14 - n_row_after_15))
```

```
In [ ]: # Calculating the ratio of observations dropped
a = int(((n_row_before-n_row_after_13) / n_row_before)*100)
b = int(100-(((n_row_before-n_row_after_13) / n_row_before)*100))
print('Proportion kept: ' + str(b) + ' %')
print('Proportion lost: ' + str(a) + ' %')
```

```
In [ ]: # Step 4: Accounting Rate of Return
# Step 4.1: Computing wealth (Equity) = \
# Total Assets - Total Liabilities
df['equity'] = (df['total_assets'] - df['liab'])

# Step 4.2: Drop rows with negative Equity
df = df[df['equity'] >= 0]

# Counting nr of dropped rows
n_row_after_16 = df.shape[0]
print('Number of rows in df (Original):         ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):   ' +\
     str(n_row_after_16))
print('Number of dropped rows:                   ' +\
     str(n_row_after_15 - n_row_after_16))
```

```
In [ ]: # Step 4.3: Computing ROE
df['ROE'] = df['net_profit'] / df['equity']

# Step 4.4: Drop rows with an infinit ROE
df.replace([np.inf, -np.inf], np.nan, inplace=True)
df = df.dropna(subset=['ROE'])

# Counting nr of dropped rows
n_row_after_17 = df.shape[0]
print('Number of rows in df (Original):         ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):   ' + \
    str(n_row_after_17))
print('Number of dropped rows:                       ' + \
    str(n_row_after_16 - n_row_after_17))
```

```
In [ ]: # Calculating the ratio of observations dropped
a = int(((n_row_before-n_row_after_14) / n_row_before)*100)
b = int(100-(((n_row_before-n_row_after_14) / n_row_before)*100))
print('Proportion kept: ' + str(b) + ' %')
print('Proportion lost: ' + str(a) + ' %')
```

```
In [ ]: # Keeping only the companies with observations with more than 9 years
        df1 = df
        df1['firm_years'] = df1.groupby('orgnr')['year'].\
            transform('nunique')
        df1 = df1[df1['firm_years']>=10]
        df1 = df1[df1['year']>=2007]
        x = len(df1)

        # Counting nr of dropped rows
        n_row_after_16 = len(df1)
        print('Number of rows in df (Original):         ' + \
            str(n_row_before))
        print('Number of rows of the dataframe (New):    ' + \
            str(n_row_after_16))
        print('Number of dropped rows:                  ' + \
            str(n_row_after_15 - n_row_after_16))
```

```
In [ ]: # DROP COMPANIES WITH OBSERVATIONS FROM ONLY FEW YEARS

        # Step 1: Identify the minimum and maximum years in the DataFrame
        #min_year = 2007
        #max_year = 2019

        # Step 2: Check if any year is missing for each company
        #missing_years = df.groupby('orgnr')['year'].\
        # apply(lambda x: len(x) != (max_year - min_year + 1))

        # Step 3: Create a boolean mask to identify the rows with\
        #  missing years
        #rows_with_missing_years = df['orgnr'].isin(missing_years\
        # [missing_years].index)

        # Step 4: Drop the rows with missing years from the DataFrame
        #df1 = df[~rows_with_missing_years]

        # Counting nr of dropped rows
        #n_row_after_16 = len(df1)c
        #print('Number of rows in df (Original):         ' + str(n_row_before))
        #print('Number of rows of the dataframe (New):    ' + str(n_row_after_16))
        #print('Number of dropped rows:                  ' + str(n_row_after_15 - n_row_afte
```

```
In [ ]: # Calculating the ratio of observations dropped
        a = int(((n_row_before-n_row_after_16) / n_row_before)*100)
        b = int(100-(((n_row_before-n_row_after_16) / n_row_before)*100))
        print('Proportion kept: ' + str(b) + ' %')
        print('Proportion lost: ' + str(a) + ' %')
```

```
In [ ]: # Checking for duplicates in orgnr and year
        duplicates_df = df.duplicated(subset=['orgnr','year'])
        duplicates_df1 = df1.duplicated(subset=['orgnr','year'])

        print('The number of duplicates in orgnr and year in df is:  ',\
             duplicates_df.sum())
```

```
print('The number of duplicates in orgnr and year in df1 is: ', \
    duplicates_df1.sum())
```

In [ ]:
```
# Save pandas dataframe as parquet file
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(df)

# Write table to a parquet file with Brotli compression

output_file = 'N:/durable/BIStudents/IssamSheikh/data/clean\
    /clean_regn_end_all.parquet'
pq.write_table(table, output_file)
```

In [ ]:
```
# Save pandas dataframe as parquet file
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(df1)

# Write table to a parquet file with Brotli compression

output_file = 'N:/durable/BIStudents/IssamSheikh/data/clean\
    /clean_regn_end.parquet'
pq.write_table(table, output_file)
```

In [ ]:
```
# Save dataframes as Stata files
#df.to_stata('N:/durable/BIStudents/IssamSheikh/codes/5_regressions/\
# stata/clean_regn_end_all.dta')
#df1.to_stata('N:/durable/BIStudents/IssamSheikh/codes/5_regressions\
# /stata/clean_regn_end.dta')
```

In [ ]:

```

```

In this notebook we will

1. Compute the firm's age variable
2. Differenciate between public and private firms

In [ ]:

```python
# Import needed libraries
import pandas as pd
from tabulate import tabulate
import pyarrow as pa
import pyarrow.parquet as pq
```

In [ ]:

```python
# Step 1: Reading dataset with all the years
df_aksje_rw = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data\
    /raw/aksjeselskaper/aksjeselskaper_all.parquet')

# Counting number of rows of the original dataframe
n_row_before = df_aksje_rw.shape[0]
print('Number of rows in df (Original):         ' +\
     str(n_row_before))
```

In [ ]:

```python
# Step 2: Only keep the variables of interest
df_aksje_or = df_aksje_rw[['year','orgnr','est_date','sto_ex','org_form']]

# Drop duplicates
df_aksje = df_aksje_or.drop_duplicates()

# Counting number of rows after droping duplicates
n_row_after1 = df_aksje.shape[0]
print('Number of rows in df (Original):         ' + \
    str(n_row_before))
print('Number of rows of the dataframe (New):   ' + \
    str(n_row_after1))
print('Number of dropped rows:                   ' +\
     str(n_row_before - n_row_after1))
```

In [ ]:

```python
# Drop nans
df_aksje = df_aksje.dropna(subset=['year','orgnr','est_date','org_form'])

n_row_after2 = df_aksje.shape[0]
print('Number of rows in df (Original):         ' +\
     str(n_row_before))
print('Number of rows of the dataframe (New):   ' + \
    str(n_row_after2))
print('Number of dropped rows:                  ' + \
    str(n_row_after1 - n_row_after2))
```

In [ ]:

```python
# Step 4: Create the Dummy Variables For 'public_listed',
# 'public_nonlisted' and 'private'
# Transform the sto_ex: 1 if listed, 0 if not.
df_aksje['sto_ex'] = df_aksje['sto_ex'].replace(({'':0, '1':1}))
```

```
df_aksje['sto_ex'] = df_aksje['sto_ex'].replace(({'OB':1}))
df_aksje['sto_ex'] = df_aksje['sto_ex'].replace(({'OA':1}))

# Transform the sto_ex to "public_listed"
df_aksje['public_listed'] = df_aksje['sto_ex'].apply(lambda x: 1 if x == 1 else 0)

# Transform the org_form: 1 if ASA (public_nonlisted), 0 if not (private)
df_aksje['public_nonlisted'] = df_aksje.apply(lambda row: 0 if \
    row['public_listed'] == 1 else (1 if row['org_form'] == 'ASA' else 0), axis = 1
df_aksje['private']          = df_aksje.apply(lambda row: 0 if \
    row['public_listed'] == 1 else (1 if row['org_form'] != 'ASA' else 0), axis = 1
df_aksje['public']           = df_aksje.apply(lambda row: 0 if \
    row['private'] == 1 else 1, axis=1)

# Drop sto_ex and org_form from the dataframe
df_aksje = df_aksje.drop('sto_ex', axis = 1)
df_aksje = df_aksje.drop('org_form', axis = 1)

# Count number of different profiles
print('# of private firms:', df_aksje['private'].sum())
print('# of public firms:    ', df_aksje['public'].sum())
print('    non_listed firms: ', df_aksje['public_nonlisted'].sum())
print('    listed firms:     ', df_aksje['public_listed'].sum())
```

```
In [ ]: # Step 8.4: Check if the Dummy Variables Are Mutually Exclusive, and
        #which rows where it is violated
        violation = df_aksje[(df_aksje['public_nonlisted'] == 1) & \
            ((df_aksje['private'] == 1) | (df_aksje['public_listed'] == 1))]
        if not violation.empty:
            print("Something is wrong in the following rows:")
            #print(violation)
        else:
            print("The variables: 'private', 'public_listed', and \
                'public_nonlisted' are all mutually exclusive.")
```

```
In [ ]: # Step 5: Computing the age variable
            # Convert "dt_bankrupt" to a String Format and Only
            # Keep the First 4 Characters
        df_aksje['est_date'] = df_aksje['est_date'].str[:4]

            # Convert to Numerical Type
        df_aksje['est_date'] = pd.to_numeric(df_aksje['est_date'])

            # The Age Variable
        df_aksje['age'] = df_aksje['year'] - df_aksje['est_date']
```

```
In [ ]: # Checking for duplicates in orgnr and year
        dup_df_aksje = df_aksje.duplicated(subset=['orgnr','year'])
        print('The number of duplicates in orgnr and year \
            in df_aksje is:  ', dup_df_aksje.sum())
```

```
In [ ]: # Droping duplicates in orgnr and year
        df_aksje = df_aksje.drop_duplicates(subset=['orgnr','year'], keep='last')

        # Counting number of rows after droping duplicates in orgnr and year
```

```
n_row_after3 = df_aksje.shape[0]
print('Number of rows in df (Original):         ' + str(n_row_before))
print('Number of rows of the dataframe (New):   ' + str(n_row_after3))
print('Number of dropped rows:                  ' \
    + str(n_row_after2 - n_row_after3))
```

```
In [ ]: # Some observations
        #print('# of private firms (based on sto_ex):   ', \
        # df_aksje[df_aksje['sto_ex']==0]['orgnr'].nunique())
        #print('# of private firms (based on org_form): ', \
        # df_aksje[df_aksje['public']==0]['orgnr'].nunique())
        #print('# of public firms (based on sto_ex):    ', \
        # df_aksje[df_aksje['sto_ex']==1]['orgnr'].nunique())
        #print('# of public firms (based on org_form):  ', \
        # df_aksje[df_aksje['public']==1]['orgnr'].nunique())
        print('Average firm age: ', df_aksje['age'].mean())

            # Number of private and public companies based on sto_ex and org_form
        c1 = df_aksje[df_aksje['private']==1].groupby('year').size()
        c2 = df_aksje[df_aksje['public_listed']==1].groupby('year').size()
        c3 = df_aksje[df_aksje['public_nonlisted']==1].groupby('year').size()

            # Concatenate the DataFrames horizontally
        result_df = pd.concat([c1, c2, c3], axis=1)

            # Print the resulting table
        print(tabulate(result_df,  headers = ["private", "public_listed",\
             'public_nonlisted']))
```

```
In [ ]: print('The number of duplicated on orgnr and year in\
         df_akje_raw is: ',df_aksje.duplicated(subset=['orgnr','year']).sum())
```

```
In [ ]: # Saving the clean version of aksjeselskaper data
        table = pa.Table.from_pandas(df_aksje)
        output_file = 'N:/durable/BIStudents/IssamSheikh/data\
            /clean/clean_aksje_all.parquet'
        pq.write_table(table, output_file)
```

```
In [ ]:
```

```
In [ ]: # In this notebook I will try to retrieve data about
        #bankrupty, mergers, and aquisitions, and sector code
```

```
In [ ]: # Import needed libraries
        import pandas as pd
        import numpy as np
        import pyarrow as pa
        import pyarrow.parquet as pq
        from tabulate import tabulate

        # Display 6 columns for viewing pupuses
        pd.set_option('display.max_columns', 10)
        # Reduce decimals points to 2
        pd.options.display.float_format = '{:,.2f}'.format
```

```
In [ ]: # Read dataset
        df_vof_raw = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/\
            raw/vof_fore/VOF_202112.parquet')

        # Counting number of rows before cleaning
        n_row_before = len(df_vof_raw)
        print('Number of rows in df (Original): ' + str(n_row_before))
```

```
In [ ]: df_vof_raw['orgnr'].nunique()
```

```
In [ ]: # Only keep variables of interest
        df_vof_or = df_vof_raw[['orgnr', 'dt_bankrupt', 'dt_merger', \
            'dt_accq', 'act_primary']]

        # Drop duplicates
        df_vof_or = df_vof_or.drop_duplicates()

        # Counting number of rows after droping duplicates
        n_row_after1 = df_vof_raw.shape[0]
        print('Number of rows in df (Original):          ' + str(n_row_before))
        print('Number of rows of the dataframe (New):    ' + str(n_row_after1))
        print('Number of dropped rows:                   ' +\
             str(n_row_before - n_row_after1))
```

```
In [ ]: # Replace empty strings with NaN
        df_vof = df_vof_or.replace('', np.nan)
```

```
In [ ]: # Fixing the dates
        # Convert dt_bankrupt, dt_merger, and dt_accq to a string
        # format and only keep only the first 4 characters
        df_vof['dt_bankrupt']      = df_vof['dt_bankrupt'].str[:4]
        df_vof['dt_merger']        = df_vof['dt_merger'].str[:4]
        df_vof['dt_accq']          = df_vof['dt_accq'].str[:4]
        df_vof['paper']            = df_vof['act_primary'].str.split\
            ('.').str[0] ## Accessing the first two numbers of the activity code
```

```
# Convert to numerical type
df_vof['dt_bankrupt']       = pd.to_numeric(df_vof['dt_bankrupt'])
df_vof['dt_merger']         = pd.to_numeric(df_vof['dt_merger'])
df_vof['dt_accq']           = pd.to_numeric(df_vof['dt_accq'])
df_vof['paper']             = pd.to_numeric(df_vof['paper'])
```

In [ ]:
```
# Only keep data from 2005 and on
df_vof.loc[df_vof['dt_bankrupt'] < 2005, 'dt_bankrupt'] = np.nan
df_vof.loc[df_vof['dt_merger'] < 2005  , 'dt_merger'] = np.nan
df_vof.loc[df_vof['dt_accq'] < 2005    , 'dt_accq'] = np.nan
```

In [ ]:
```
# Drop NaNs
df_vof = df_vof.dropna(subset=['dt_bankrupt', 'dt_merger', 'dt_accq', \
    'act_primary', 'paper'], how='all')

# Counting number of rows before droping rows with missing variables
n_row_after_1 = df_vof.shape[0]
print('Number of rows in df (Original):          ' + \
    tr(n_row_before))
print('Number of rows of the dataframe (New):    ' + \
    str(n_row_after_1))
print('Number of dropped rows:                   ' + \
    str(n_row_before - n_row_after_1))
```

In [ ]:
```
# Creating dummy variables for bankrupty, mergers, and accq\
 (=1 if took place and 0 otherwise)
df_vof['bankrupt'] = df_vof['dt_bankrupt'].notna().astype(int)
df_vof['merger']   = df_vof['dt_merger'].notna().astype(int)
df_vof['accq']     = df_vof['dt_accq'].notna().astype(int)
df_vof['paper']    = (~df_vof['paper'].isin([64, 65, 66, 68, 85, 86])).\
    astype(int) # Gives 1 if as in paper, 0 otherwise
```

In [ ]:
```
# Some observations
print('# of firms that went bankrupt since 2005:        ',\
 df_vof['bankrupt'].sum())
print('# of firms that got aquired since 2005:          ', \
df_vof['accq'].sum())
print('# of firms that merged since 2005:               ', \
df_vof['merger'].sum())
print('# of firms in paper since 2005:                  ', \
df_vof['paper'].sum())
```

In [ ]:
```
# Some observations on yearly basis
    # Number of private and public companies based on sto_ex and org_form
c1 = df_vof[df_vof['bankrupt']==1].groupby('dt_bankrupt').size()
c2 = df_vof[df_vof['merger']==1].groupby('dt_merger').size()
c3 = df_vof[df_vof['accq']==1].groupby('dt_accq').size()
    # Concatenate the DataFrames horizontally
result_df = pd.concat([c1, c2, c3], axis=1)
#averages = result_df.mean().rename('avg')
#result_df = result_df.append(averages)
```

```
    # Print the resulting table
print(tabulate(result_df,  headers = ["Bankrupties", "Mergers", 'Aquisitions']))
```

In [ ]:
```
# Create different dataframes for each action
df1 = (df_vof[['orgnr', 'dt_bankrupt', 'bankrupt']]).dropna()
df2 = (df_vof[['orgnr', 'dt_accq', 'accq']]).dropna()
df3 = (df_vof[['orgnr', 'dt_merger', 'merger']]).dropna()
df4 = (df_vof[['orgnr',  'act_primary', 'paper']]).dropna()
# Counting rows
print('# of firms that went bankrupt since 2005:    ', len(df1))
print('# of firms that got aquired since 2005:      ', len(df2))
print('# of firms that merged since 2005:           ', len(df3))
print('# of firms when following the paper:         ', len(df4))
```

In [ ]:
```
# Convert datafram to Apach arrow table
table1 = pa.Table.from_pandas(df1)
table2 = pa.Table.from_pandas(df2)
table3 = pa.Table.from_pandas(df3)
table4 = pa.Table.from_pandas(df4)
# Write table to a parquet file with Brotli compression
output_file1 = 'N:/durable/BIStudents/IssamSheikh/data/clean/\
    bankruptcies_since_2005.parquet'
output_file2 = 'N:/durable/BIStudents/IssamSheikh/data/clean/\
    acquisitions_since_2005.parquet'
output_file3 = 'N:/durable/BIStudents/IssamSheikh/data/clean/\
    mergers_since_2005.parquet'
output_file4 = 'N:/durable/BIStudents/IssamSheikh/data/clean/\
    paper_companies.parquet'
pq.write_table(table1, output_file1)
pq.write_table(table2, output_file2)
pq.write_table(table3, output_file3)
pq.write_table(table4, output_file4)
```

In [ ]:

```
In [ ]: pip install nbconvert
```

```
In [ ]: # Import needed libraries
        import pandas as pd
        import numpy as np
        import pyarrow as pa
        import pyarrow.parquet as pq
```

```
In [ ]: # Step 1: Read the datasets:
        df_aksje_raw        = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/raw\
            /aksjeselskaper/aksjeselskaper_all.parquet')
        df_vof_raw          = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/raw\
            /vof_fore/VOF_202112.parquet')
```

```
In [ ]: # Step 2: Keep the Variables of Interest and Cut the Dataframe
        # So It Starts From 2007

        # Aksje
        df_aksje_categories = df_aksje_raw[['orgnr', 'year', 'sto_ex', 'org_form']]
        df_aksje_categories = df_aksje_categories[(df_aksje_categories['year'] >= 2007)]
```

```
In [ ]: # Step 4: Create the Dummy Variables For 'public_listed',
        #'public_nonlisted' and 'private'

        # Transform the sto_ex: 1 if listed, 0 if not.
        df_aksje_categories['sto_ex']           = df_aksje_categories['sto_ex'].\
            replace(({'':0, '1':1}))
        df_aksje_categories['sto_ex']           = df_aksje_categories['sto_ex'].\
            replace(({'OB':1}))
        df_aksje_categories['sto_ex']           = df_aksje_categories['sto_ex'].\
            replace(({'OA':1}))

        # Transform the sto_ex to "public_listed"
        df_aksje_categories['public_listed']    = df_aksje_categories['sto_ex'].\
            apply(lambda x: 1 if x == 1 else 0)

        # Transform the org_form: 1 if ASA (public_nonlisted), 0 if not (private)
        df_aksje_categories['public_nonlisted'] = df_aksje_categories.apply\
            (lambda row: 0 if row['public_listed'] == 1 else (1 if row['org_form']\
                 == 'ASA' else 0), axis = 1)
        df_aksje_categories['private']          = df_aksje_categories.apply\
            (lambda row: 0 if row['public_listed'] == 1 else (1 if row['org_form']\
                 != 'ASA' else 0), axis = 1)

        # Drop sto_ex and org_form from the dataframe
        df_aksje_categories                     = df_aksje_categories.\
            drop('sto_ex', axis = 1)
        df_aksje_categories                     = df_aksje_categories.\
            drop('org_form', axis = 1)
```

```
In [ ]: # Step 5: Add Additional Rows
        max_years                       = df_aksje_categories.\
            groupby('orgnr')['year'].max()
```

```
extra_rows                      = pd.DataFrame({'orgnr':\
     max_years.index, 'year': max_years + 1})

extra_rows['public_listed']     = 0
extra_rows['public_nonlisted']  = 0
extra_rows['private']           = 0
extra_rows['bnkrpt']            = 0
df_aksje_categories             = pd.concat([df_aksje_categories\
    , extra_rows], ignore_index=True)

# Test
# print(df_aksje_categories[df_aksje_categories['orgnr'] == 'F004425633'])
```

```
In [ ]: # Check if the dummies are all equal to 0
        f1 = df_aksje_categories[df_aksje_categories[['private', \
            'public_listed', 'public_nonlisted']].nunique(axis=1) == 1]

        # These are a result of the extra rows added.
```

```
In [ ]: # Step 6: Check if the Dummy Variables Are Mutually Exclusive

        violation = df_aksje_categories[(df_aksje_categories\
            ['public_listed'] == 1) & ((df_aksje_categories['public_nonlisted'] == 1)\
                 | (df_aksje_categories['private'] == 1))]
        if not violation.empty:
            print("Something is wrong in the following rows:")
            print(violation)
        else:
            print("All good")

        violation1 = df_aksje_categories[(df_aksje_categories['public_nonlisted'] == 1)\
             & (df_aksje_categories['private'] == 1)]
        if not violation1.empty:
            print("Something is wrong in the following rows:")
            print(violation1)
        else:
            print("All good")
```

```
In [ ]: # Step 7: The Bankruptcy Dummy
        # Keep the variables of interest
        df_vof_raw                   = df_vof_raw[['orgnr', 'dt_bankrupt']]

        # Replace empty strings with NaNs, and delete the rows
        df_vof_raw                   = df_vof_raw.replace('', np.nan)
        df_vof_raw                   = df_vof_raw.dropna(subset=['dt_bankrupt'])

        # Convert dt_bankrupt to a string format and only keep only the first
        # 4 characters, and then convert to numerical
        df_vof_raw['dt_bankrupt']    = df_vof_raw['dt_bankrupt'].str[:4]
        df_vof_raw['dt_bankrupt']    = pd.to_numeric(df_vof_raw['dt_bankrupt'])

        # Cut the dataframe so it only have observations from 2007 and up
        df_vof_raw                   = df_vof_raw[(df_vof_raw['dt_bankrupt'] >= 2007)]

        # Creating dummy variables for bankrupty, mergers, and accq (=1 if took
# place and 0 otherwise)
df_vof_raw['bankrupt']      = df_vof_raw['dt_bankrupt'].notna().astype(int)
```

In [ ]:
```
# Step 8: Merge the datasets
merge_cat               = df_aksje_categories.merge(df_vof_raw, on = \
    'orgnr', how = 'left')

# Gives bkrtp = 1 if the date of bankruptcy is consistent with the
# year, otherwise it gives 0
merge_cat['bankrupt']   = merge_cat['dt_bankrupt'].eq(\
    merge_cat['year']).astype(int)

# Test
# print(merge_cat[merge_cat['orgnr'] == 'F004425633'])
```

In [ ]:
```
f2  = merge_cat[merge_cat[['private', 'public_listed', \
'public_nonlisted', 'bankrupt']].nunique(axis=1) == 1]
```

In [ ]:
```
# Step 8.1 Check if the rows where dummy variabels are all 0

f2  = merge_cat[merge_cat[['private', 'public_listed', \
    'public_nonlisted', 'bankrupt']].nunique(axis=1) == 1]

# This is the organization I was talking about: \
# merge_cat[merge_cat['orgnr']== 'F000005421']
# We have only 2007 (and 2008 with the added rows), \
# and it went bankrupt in 2010.

# Step 8.2: Delete f2 From merge_cat

# before deleting
before = merge_cat.shape[0]

# deleting
df_merged = merge_cat[~merge_cat['orgnr'].isin(f2['orgnr'])]

after = df_merged.shape[0]

print( " Number of rows deleted when  dropping the organizations: ", before-after)
```

In [ ]:
```
# Step 8.3: Check the dummy variables now:

f3 = df_merged[df_merged[['private', 'public_listed',\
     'public_nonlisted', 'bankrupt']].nunique(axis=1) == 1]

# None!!
```

In [ ]:
```
# Step 8.4: Check if any of profile, list or nonlisted are 1 when bankrupt is 1.

private     = df_merged[(df_merged['bankrupt'] == 1) & \
    ((df_merged['private']))]
listed      = df_merged[(df_merged['bankrupt'] == 1) & \
    ((df_merged['public_listed']))]
nonlisted   = df_merged[(df_merged['bankrupt'] == 1) & \
    ((df_merged['public_nonlisted']))]
```

```
# All these are empty -- we're good!!
```

In [ ]:
```
# Step 9: Create a parquet of merge_cat to use for profitability analyses
table = pa.Table.from_pandas(df_merged)

# Write table to a parquet file with Brotli compression
output_file = 'N:/durable/BIStudents/IssamSheikh/codes/Sheikh/\
    test_for_bankruptcy_reg/parquets/profiles_deleted_zeros.parquet'
pq.write_table(table, output_file)
```

In [ ]:
```
# Drop est_dat and dt_bankrupt when everything is done.
```

In [ ]:
```
# Step 10: Dropping 'est_date' and 'dt_bankrupt' From the Datasets
# I'll copy the dataframe just in case something goes wrong

lifecycle = merge_cat[['year', 'orgnr', 'public_listed', 'public_nonlisted',\
     'private', 'bankrupt']]
```

In [ ]:
```
# Step 10.1: Creating dummies for the 'paths'

lifecycle['pri_to_bank']                = 0
lifecycle['pri_to_nonlist']             = 0
lifecycle['pri_to_list']                = 0
lifecycle['nonlist_to_list']            = 0
lifecycle['nonlist_to_pri']             = 0
lifecycle['nonlist_to_bank']            = 0
lifecycle['list_to_nonlist']            = 0
lifecycle['list_to_pri']                = 0
lifecycle['list_to_bank']               = 0
```

In [ ]:
```
# Step 10.2: Need to group the transitions by organization number:

pri_to_bank                 = lifecycle.groupby('orgnr').apply(lambda x: \
    x[(x['private'].shift() == 1) & (x['private'] == 0) & (x['bankrupt'].\
        shift() == 0) & (x['bankrupt'] == 1)].index.to_list())
pri_to_pub_nonlist          = lifecycle.groupby('orgnr').apply(lambda x: \
    x[(x['private'].shift() == 1) & (x['private'] == 0) & (x['public_nonlisted']\
        .shift() == 0) & (x['public_nonlisted'] == 1)].index.to_list())
pri_to_pub_list             = lifecycle.groupby('orgnr').apply(lambda x:\
     x[(x['private'].shift() == 1) & (x['private'] == 0) & (x['public_listed'].\
        shift() == 0) & (x['public_listed'] == 1)].index.to_list())
pub_nonlist_to_pub_list     = lifecycle.groupby('orgnr').apply(lambda x:\
     x[(x['public_nonlisted'].shift() == 1) & (x['public_nonlisted'] == 0)\
         & (x['public_listed'].shift() == 0) & (x['public_listed'] == 1)]\
            .index.to_list())
pub_nonlist_to_pri          = lifecycle.groupby('orgnr').apply(lambda \
    x: x[(x['public_nonlisted'].shift() == 1) & (x['public_nonlisted'] \
        == 0) & (x['private'].shift() == 0) & (x['private'] == 1)]\
            .index.to_list())
pub_nonlist_to_bank         = lifecycle.groupby('orgnr').apply(lambda \
    x: x[(x['public_nonlisted'].shift() == 1) & (x['public_nonlisted'] \
        == 0) & (x['bankrupt'].shift() == 0) & (x['bankrupt'] == 1)]\
            .index.to_list())
pub_list_to_nonlist         = lifecycle.groupby('orgnr').apply(lambda\
     x: x[(x['public_listed'].shift() == 1) & (x['public_listed'] == 0)\
         & (x['public_nonlisted'].shift() == 0) & (x['public_nonlisted'] \
            == 1)].index.to_list())
pub_list_to_pri             = lifecycle.groupby('orgnr').apply(lambda \
    x: x[(x['public_listed'].shift() == 1) & (x['public_listed'] == 0) \
        & (x['private'].shift() == 0) & (x['private'] == 1)].index.to_list())
pub_list_to_bank            = lifecycle.groupby('orgnr').apply(lambda \
    x: x[(x['public_listed'].shift() == 1) & (x['public_listed'] == 0) & \
        (x['bankrupt'].shift() == 0) & (x['bankrupt'] == 1)].index.to_list())
```

```
In [ ]: # Step 10.3: Need to flatten the grouped transitions:
        # The grouped transitions are nested lists, meaning they are made up by other subl
        # By flattening them we concatenated the sublists together into a single, one-dimen

        pri_to_bank                 = [idx for sublist in\
             pri_to_bank for idx in sublist]
        pri_to_pub_nonlist          = [idx for sublist in \
            pri_to_pub_nonlist for idx in sublist]
        pri_to_pub_list             = [idx for sublist in \
            pri_to_pub_list for idx in sublist]
        pub_nonlist_to_pub_list     = [idx for sublist in \
            pub_nonlist_to_pub_list for idx in sublist]
        pub_nonlist_to_pri          = [idx for sublist in\
             pub_nonlist_to_pri for idx in sublist]
        pub_nonlist_to_bank         = [idx for sublist in \
            pub_nonlist_to_bank for idx in sublist]
        pub_list_to_nonlist         = [idx for sublist in \
            pub_list_to_nonlist for idx in sublist]
        pub_list_to_pri             = [idx for sublist in \
            pub_list_to_pri for idx in sublist]
        pub_list_to_bank            = [idx for sublist in \
            pub_list_to_bank for idx in sublist]
```

```
In [ ]: # Step 10.4: Set the dummy variables to 1 for when the transition take place

        lifecycle.loc[pri_to_bank,             'pri_to_bank']         = 1
        lifecycle.loc[pri_to_pub_nonlist,      'pri_to_nonlist']      = 1
        lifecycle.loc[pri_to_pub_list,         'pri_to_list']         = 1
        lifecycle.loc[pub_nonlist_to_pub_list, 'nonlist_to_list']     = 1
        lifecycle.loc[pub_nonlist_to_pri,      'nonlist_to_pri']      = 1
        lifecycle.loc[pub_nonlist_to_bank,     'nonlist_to_bank']     = 1
        lifecycle.loc[pub_list_to_nonlist,     'list_to_nonlist']     = 1
        lifecycle.loc[pub_list_to_pri,         'list_to_pri']         = 1
        lifecycle.loc[pub_list_to_bank,        'list_to_bank']        = 1
```

```
In [ ]: # Create a variable frm private to public
        lifecycle['pri_to_pub'] = np.where((lifecycle['pri_to_nonlist'] == 1) |\
             (lifecycle['pri_to_list'] == 1), 1, 0)
```

```
In [ ]: # Step 10: Save lifecycle as a parquet
        table = pa.Table.from_pandas(lifecycle)

        # Write table to a parquet file with Brotli compression
```

```
output_file = 'N:/durable/BIStudents/IssamSheikh/data/lifecycle.parquet'
pq.write_table(table, output_file)
```

```
In [ ]: pip install nbconvert
```

```
In [ ]: # Import needed libraries
        import pandas as pd
```

```
In [ ]: # Read datasets
        df_regn = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data\
            /clean/clean_regn_end_all.parquet')
        df_aksje = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data\
            /clean/clean_aksje_all.parquet')
        df_vof = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data\
            /clean/clean_VOF_202112.parquet')
```

```
In [ ]: # Perform matching analysis regn VS aksje
        # Get unique years from both dataframes
        years = pd.concat([df_regn['year'], df_aksje['year']]).unique()

        # Initialize empty lists to store the counts
        regn_only_counts = []
        aksje_only_counts = []
        both_counts = []

        # Loop over the years
        for year in years:
            # Filter the data for the current year
            df_regn_year = df_regn[df_regn['year'] == year]
            df_aksje_year = df_aksje[df_aksje['year'] == year]

            # Count the number of rows in df_regn only
            regn_only_count = len(df_regn_year[~df_regn_year['orgnr'].\
                isin(df_aksje_year['orgnr'])])
            regn_only_counts.append(regn_only_count)

            # Count the number of rows in df_aksje only
            aksje_only_count = len(df_aksje_year[~df_aksje_year['orgnr'].\
                isin(df_regn_year['orgnr'])])
            aksje_only_counts.append(aksje_only_count)

            # Count the number of rows in both df_regn and df_aksje
            both_count = len(df_regn_year[df_regn_year['orgnr'].\
                isin(df_aksje_year['orgnr'])])
            both_counts.append(both_count)

        # Create a new dataframe with the results
        results_df = pd.DataFrame({'Year': years,
                                   'df_regn_only': regn_only_counts,
                                   'df_aksje_only': aksje_only_counts,
                                   'Both Dataframes': both_counts})

        # Set the 'Year' column as the index
        results_df.set_index('Year', inplace=True)

        print(results_df)
```

In [ ]:
```
# Perform matching analysis regn VS vof
regn_only = df_regn[~df_regn['orgnr'].isin(df_vof['orgnr'])]['orgnr'].nunique()
vof_only = df_vof[~df_vof['orgnr'].isin(df_regn['orgnr'])]['orgnr'].nunique()
both = df_regn[df_regn['orgnr'].isin(df_vof['orgnr'])]['orgnr'].nunique()

# Create a new dataframe with the results
results_df = pd.DataFrame({'df_regn only': [regn_only],
                           'df_vof only': [vof_only],
                           'Both': [both]})

print(results_df)
```

In [ ]:
```
# Perform matching analysis aksje VS vof
aksje_only = df_aksje[~df_aksje['orgnr'].isin(df_vof['orgnr'])]['orgnr'].nunique()
vof_only = df_vof[~df_vof['orgnr'].isin(df_aksje['orgnr'])]['orgnr'].nunique()
both = df_aksje[df_aksje['orgnr'].isin(df_vof['orgnr'])]['orgnr'].nunique()

# Create a new dataframe with the results
results_df = pd.DataFrame({'df_regn only': [aksje_only],
                           'df_vof only': [vof_only],
                           'Both': [both]})

print(results_df)
```

```
In [ ]:
```

```
In [ ]: # Importing needed libraries
        import pandas as pd
        import pyarrow as pa
        import pyarrow.parquet as pq
        import numpy as np
        from tabulate import tabulate
```

```
In [ ]: # Reading all data sets
        df_regn     = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/\
            clean/clean_regn_end_all.parquet')
        df_aksje    = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/\
            clean/clean_aksje_all.parquet')
        df_vof      = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/\
            clean/bankruptcies_since_2005.parquet')
        df_life     = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/\
            lifecycle.parquet')
        df_industry = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/\
            raw/vof_fore/industry_all.parquet')
```

```
In [ ]: # Restrict the dataframes to 2007 and on
        df_regn = df_regn[df_regn['year']>=2007]
        df_aksje = df_aksje[df_aksje['year']>=2007]
        df_vof = df_vof[(df_vof['dt_bankrupt']>=2007) & (df_vof['dt_bankrupt']<=2019)]
        df_life = df_life[(df_life['year']>=2007) & (df_life['year']<=2019)]

        # Sort using orgnr and year
        df_regn = df_regn.sort_values(['orgnr', 'year'])
        df_aksje = df_aksje.sort_values(['orgnr', 'year'])
        df_vof = df_vof.sort_values('orgnr')
        df_life = df_life.sort_values(['orgnr', 'year'])
```

```
In [ ]: # Counting the number of rows in different dataframes
        print('Number of rows in df_regn (Original) :  ' + str(len(df_regn)))
        print('Number of rows in df_aksje (Original):  ' + str(len(df_aksje)))
        print('Number of rows in df_vof (Original)  :    ' + str(len(df_vof)))
        print('Number of rows in df_life (Original) :  ' + str(len(df_life)))
```

```
In [ ]: print('Number of unique orgnrs in df_regn:   ' + str(df_regn['orgnr'].nunique()))
        print('Number of unique orgnrs in df_aksje:  ' + str(df_aksje['orgnr'].nunique()))
        print('Number of unique orgnrs in df_vof:     ' + str(df_vof['orgnr'].nunique()))
        print('Number of unique orgnrs in df_life:   ' + str(df_life['orgnr'].nunique()))
```

```
In [ ]: # Merge df_regn and and df_aksje
        df_merg1 = df_regn.merge(df_aksje, on=['orgnr','year'], how='inner')

        # Counting the number of rows
        print('Number of rows in df_merg1: ' + str(len(df_merg1)))
```

```
In [ ]: # Expected df size after adding new rows
        #print('Expected rows after adding extra rows: ' + str(df_merg1['orgnr'].nunique()
In [ ]: # Find the maximum year for each firm
        #max_years = df_regn.groupby('orgnr')['year'].max()
        #max_years = max_years.reset_index(name='max_year')

        # Create a new DataFrame with additional rows
        #new_rows = []
        #for row in max_years.itertuples(index=False):
        #    orgnr = row.orgnr
        #    max_year = row.max_year
        #    for year in range(max_year + 1, 2020):  # Range from max_year + 1 to 2019
        #        new_rows.append({'orgnr': orgnr, 'year': year})

        # Create a DataFrame with all years from 2005 to 2019
        #all_years = pd.DataFrame({'orgnr': np.repeat(df_regn['orgnr'].unique(), 15),
        #                          'year': np.tile(np.arange(2005, 2020), df_regn['orgnr'].

        # Merge the original DataFrame with the additional rows
        #df_regn0 = pd.merge(df_regn, all_years, on=['orgnr', 'year'], how='right')

        # Counting the number of rows in df_merge
        #print('Number of rows in df_regn: ' + str(len(df_regn0)))
```

```
In [ ]: # Merge df_merge2 with df_vof on orgnr
        df_merg2 = df_merg1.merge(df_vof, on='orgnr', how='left')

        # Set firms' profile to 0 in the year of bankruptcy
        df_merg2.loc[df_merg2['bankrupt'] == 1, ['private', \
            'public_nonlisted', 'public_listed']] = 0

        # Counting numeber of rows in df_merge and the number of bankruptcies
        print('Number of rows in df_merg1: ' + str(len(df_merg2)))
        print('Number of of bankruptcies:    ' + str((df_merg2['bankrupt'].sum())))
```

```
In [ ]: # Keep only variables of interest in life
        df_life1 = df_life[['orgnr', 'year', 'pri_to_pub', 'to_listed']]
```

```
In [ ]: # Adding the lifecycle data
        df_merg3 = df_merg2.merge(df_life1, on=['year', 'orgnr'], how='left')
        # Counting the number of rows and bankruptcies
        print('Number of rows in df_merg:    ' + str(len(df_merg3)))
        print('Number of of bankruptcies:    ' + str((df_merg3['bankrupt'].sum())))
        print('Number of firms going public:      ' + str((df_merg3['pri_to_pub'].sum())))
        print('Number of firms going listed:      ' + str((df_merg3['to_listed'].sum())))
```

```
In [ ]: # Some observations
        #print('Average firm age: ', df_merg5['age'].mean())

            # Number of private and public companies based on sto_ex and org_form
        c1 = df_merg3[df_merg3['private']==1].groupby('year').size()
        c2 = df_merg3[df_merg3['public_listed']==1].groupby('year').size()
        c3 = df_merg3[df_merg3['public_nonlisted']==1].groupby('year').size()

            # Number of transitions
        c4 = df_merg3[df_merg3['pri_to_pub']==1].groupby('year').size()
```

```
c5 = df_merg3[df_merg3['to_listed']==1].groupby('year').size()

    # Concatenate the DataFrames horizontally
result_df = pd.concat([c1, c2, c3, c4, c5], axis=1)

    # Print the resulting table
print(tabulate(result_df,  headers = ["private", "public_listed",\
     'public_nonlisted', 'pri_to_pub', 'to_listed']))
```

In [ ]:
```
df_industry['orgnr'].nunique()
```

In [ ]:
```
# Introduce industry
df_industry['industry'] = df_industry['act_primary'].str[:2]
df_industry['industry'] = df_industry['industry'].astype(int)
df_merg4 = df_merg3.merge(df_industry, on='orgnr', how='left')
#df_merg4 = df_merg4.dropna(subset=['industry'])
```

In [ ]:
```
tst = df_merg4[['orgnr', 'year', 'industry']]
tst[tst['orgnr'] == 'F000000017']
```

In [ ]:
```
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(df_merg4)

# Write table to a parquet file with Brotli compression
output_file = 'N:/durable/BIStudents/IssamSheikh/data/clean/\
    regn_aksje_vof_life_merged_all1.parquet'
pq.write_table(table, output_file)
```

In [ ]:
```
pip install nbconvert
```

In [ ]:
```
# Importing needed libraries
import pandas as pd
import pyarrow as pa
import pyarrow.parquet as pq
import numpy as np
from tabulate import tabulate
```

In [ ]:
```
# Reading all data sets
df_regn     = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/clean\
    /clean_regn_end.parquet')
df_aksje    = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/clean\
    /clean_aksje_all.parquet')
df_vof      = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data/clean\
    /bankruptcies_since_2005.parquet')
df_life     = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data\
    /lifecycle.parquet')
df_industry = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/data\
    /raw/vof_fore/industry_all.parquet')
```

In [ ]:
```
# Restrict the dataframes to 2007 and on
df_regn = df_regn[df_regn['year']>=2007]
df_aksje = df_aksje[df_aksje['year']>=2007]
df_vof = df_vof[(df_vof['dt_bankrupt']>=2007) & (df_vof['dt_bankrupt']<=2019)]
df_life = df_life[(df_life['year']>=2007) & (df_life['year']<=2019)]

# Sort using orgnr and year
df_regn = df_regn.sort_values(['orgnr', 'year'])
df_aksje = df_aksje.sort_values(['orgnr', 'year'])
df_vof = df_vof.sort_values('orgnr')
df_life = df_life.sort_values(['orgnr', 'year'])
```

In [ ]:
```
# Counting the number of rows in different dataframes
print('Number of rows in df_regn (Original) :  ' + str(len(df_regn)))
print('Number of rows in df_aksje (Original):  ' + str(len(df_aksje)))
print('Number of rows in df_vof (Original)  :    ' + str(len(df_vof)))
print('Number of rows in df_life (Original) :  ' + str(len(df_life)))
```

In [ ]:
```
print('Number of unique orgnrs in df_regn:   ' + str(df_regn['orgnr'].nunique()))
print('Number of unique orgnrs in df_aksje:  ' + str(df_aksje['orgnr'].nunique()))
print('Number of unique orgnrs in df_vof:     ' + str(df_vof['orgnr'].nunique()))
print('Number of unique orgnrs in df_life:   ' + str(df_life['orgnr'].nunique()))
```

In [ ]:
```
# Merge df_regn and and df_aksje
df_merg1 = df_regn.merge(df_aksje, on=['orgnr','year'], how='inner')

# Counting the number of rows
print('Number of rows in df_merg1: ' + str(len(df_merg1)))
```

In [ ]:
```
# Expected df size after adding new rows
#print('Expected rows after adding extra rows: ' + \
# str(df_merg1['orgnr'].nunique() * 15))
```

In [ ]:
```
# Find the maximum year for each firm
#max_years = df_regn.groupby('orgnr')['year'].max()
#max_years = max_years.reset_index(name='max_year')

# Create a new DataFrame with additional rows
#new_rows = []
#for row in max_years.itertuples(index=False):
#     orgnr = row.orgnr
#     max_year = row.max_year
#     for year in range(max_year + 1, 2020):  # Range from max_year\
#   + 1 to 2019
#         new_rows.append({'orgnr': orgnr, 'year': year})

# Create a DataFrame with all years from 2005 to 2019
#all_years = pd.DataFrame({'orgnr': np.repeat(df_regn['orgnr'].\
# unique(), 15),
#                           'year': np.tile(np.arange(2005, 2020), \
# df_regn['orgnr'].nunique())})

# Merge the original DataFrame with the additional rows
#df_regn0 = pd.merge(df_regn, all_years, on=['orgnr', 'year'], how='right')

# Counting the number of rows in df_merge
#print('Number of rows in df_regn: ' + str(len(df_regn0)))
```

In [ ]:
```
# Merge df_merge2 with df_vof on orgnr
df_merg2 = df_merg1.merge(df_vof, on='orgnr', how='left')

# Set firms' profile to 0 in the year of bankruptcy
df_merg2.loc[df_merg2['bankrupt'] == 1, ['private', \
    'public_nonlisted', 'public_listed']] = 0

# Counting numeber of rows in df_merge and the number of bankruptcies
print('Number of rows in df_merg1: ' + str(len(df_merg2)))
print('Number of of bankruptcies:    ' + str((df_merg2['bankrupt'].sum())))
```

In [ ]:
```
# Keep only variables of interest in life
df_life1 = df_life[['orgnr', 'year', 'pri_to_pub', 'to_listed']]
```

In [ ]:
```
# Adding the lifecycle data
df_merg3 = df_merg2.merge(df_life1, on=['year', 'orgnr'], how='left')
# Counting the number of rows and bankruptcies
print('Number of rows in df_merg:    ' + str(len(df_merg3)))
print('Number of of bankruptcies:    ' + str((df_merg3['bankrupt'].sum())))
print('Number of firms going public:     ' + str((df_merg3['pri_to_pub'].sum())))
print('Number of firms going listed:     ' + str((df_merg3['to_listed'].sum())))
```

In [ ]:
```
# Some observations
#print('Average firm age: ', df_merg5['age'].mean())

    # Number of private and public companies based on sto_ex and org_form
c1 = df_merg3[df_merg3['private']==1].groupby('year').size()
c2 = df_merg3[df_merg3['public_listed']==1].groupby('year').size()
```

```
c3 = df_merg3[df_merg3['public_nonlisted']==1].groupby('year').size()

    # Number of transitions
c4 = df_merg3[df_merg3['pri_to_pub']==1].groupby('year').size()
c5 = df_merg3[df_merg3['to_listed']==1].groupby('year').size()

    # Concatenate the DataFrames horizontally
result_df = pd.concat([c1, c2, c3, c4, c5], axis=1)

    # Print the resulting table
print(tabulate(result_df,  headers = ["private", "public_listed",\
     'public_nonlisted', 'pri_to_pub', 'to_listed']))
```

In [ ]:
```
# Introduce industry
df_industry['industry'] = df_industry['act_primary'].str[:2]
df_industry['industry'] = df_industry['industry'].astype(int)
df_merg4 = df_merg3.merge(df_industry, on='orgnr', how='left')
#df_merg4 = df_merg4.dropna(subset=['industry'])
```

In [ ]:
```
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(df_merg4)

# Write table to a parquet file with Brotli compression
output_file = 'N:/durable/BIStudents/IssamSheikh/data/clean/\
    regn_aksje_vof_life_merged1.parquet'
pq.write_table(table, output_file)
```

In [ ]:
```

```

Create All the Variables We Need

In [ ]:
```
# Import needed libraries
import pandas as pd
import numpy as np
import pyarrow.parquet as pq
pd.set_option('mode.use_inf_as_na', True)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline
from tabulate import tabulate
from scipy.stats.mstats import winsorize
import pyarrow as pa

# Display 6 columns for viewing pupuses
pd.set_option('display.max_columns', 10)
# Reduce decimals points to 2
pd.options.display.float_format = '{:,.2f}'.format
```

In [ ]:
```
# Read Permanent and Full Sample
full_sample                = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/da
    clean/regn_aksje_vof_life_merged_all1.parquet')
permanent_sample           = pd.read_parquet('N:/durable/BIStudents/IssamSheikh/da
    clean/regn_aksje_vof_life_merged1.parquet')
```

Financial measures

In [ ]:
```
# EBITDA
full_sample['EBIT'] = full_sample['net_profit'] + full_sample['int_expe'] + \
    full_sample['tax'] + full_sample['dep']
permanent_sample['EBIT'] = permanent_sample['net_profit'] + \
    permanent_sample['int_expe'] + permanent_sample['tax'] + permanent_sample['dep'
```

In [ ]:
```
# ROA
permanent_sample['ROA'] = permanent_sample['EBIT']/\
    permanent_sample['total_assets']
full_sample['ROA']      = full_sample['EBIT']/\
    full_sample['total_assets']
```

In [ ]:
```
# ROE
full_sample['ROE']      = full_sample['EBIT'] / full_sample['equity']
permanent_sample['ROE'] = permanent_sample['EBIT'] / permanent_sample['equity']
```

In [ ]:
```
# Working Capital Ratio
permanent_sample['working_capital']      = \
    ( permanent_sample['tot_curr_assets'] - permanent_sample['curr_liab'] ) /\
 permanent_sample['total_assets']
full_sample['working_capital']           = \
    ( full_sample['tot_curr_assets'] - full_sample['curr_liab'] ) /\
         full_sample['total_assets']
```

In [ ]:
```
# Financial Leverage Ratio
full_sample['financial_leverage']     = full_sample['lt_liab']\
     / full_sample['total_assets']
permanent_sample['financial_leverage']     = permanent_sample['lt_liab']\
     / permanent_sample['total_assets']
```

In [ ]:
```
# Liquidity Ratio
permanent_sample['liquidity_ratio']      = (permanent_sample['tot_curr_assets']\
    -permanent_sample['inventory']) / permanent_sample['curr_liab']
full_sample['liquidity_ratio']           = (full_sample['tot_curr_assets']-\
    full_sample['inventory']) / full_sample['curr_liab']
```

In [ ]:
```
# Cash, Deposits and Securities Ratio
full_sample['cash_resources_to_assets']     = (full_sample['cash'] \
    + full_sample['securities']) / full_sample['total_assets']
permanent_sample['cash_resources_to_assets']     = (permanent_sample['cash']\
     + permanent_sample['securities']) / permanent_sample['total_assets']
```

In [ ]:
```
# Output
permanent_sample['output'] = (permanent_sample['net_profit']\
     + permanent_sample['emp_comp'] + permanent_sample['int_expe'] \
        + permanent_sample['dep'])
full_sample['output'] = (full_sample['net_profit'] + \
    full_sample['emp_comp'] + full_sample['int_expe'] + full_sample['dep'])
```

In [ ]:
```
# Logged output
full_sample['lg_output']  = np.log(full_sample['output'])

permanent_sample['lg_output'] = np.log(permanent_sample['output'])
```

In [ ]:
```
# Logged OP_inc and Assets
full_sample['lg_assets'] = np.log(full_sample['total_assets'])
permanent_sample['lg_assets'] = np.log(permanent_sample['total_assets'])
```

In [ ]:
```
# Capital
permanent_sample['capital'] = permanent_sample['int_assets'] +\
     permanent_sample['tan_fixed_assets']
permanent_sample = permanent_sample[permanent_sample['capital'] > 0]
full_sample['capital'] = full_sample['int_assets'] + \
    full_sample['tan_fixed_assets']
full_sample = full_sample[full_sample['capital'] > 0]
```

In [ ]:
```
# Operating Leverage
permanent_sample['operating_leverage']   = permanent_sample['tot_fixe_assets']\
     / permanent_sample['total_assets']
full_sample['operating_leverage']        = full_sample['tot_fixe_assets'] \
    / full_sample['total_assets']
```

Other

```
In [ ]: # Market shares
            # Computing operating income by industry per year
        full_sample['op_inc_ind_yr'] = full_sample.groupby(['industry', 'year'])\
            ['op_inc'].transform('sum')

            # Computing the market share by orgnr and year
        full_sample['mrkt_share'] = full_sample['op_inc'] / \
            full_sample['op_inc_ind_yr']

        # Market shares
            # Computing operating income by industry per year
        permanent_sample['op_inc_ind_yr'] = permanent_sample.groupby(\
            ['industry', 'year'])['op_inc'].transform('sum')

            # Computing the market share by orgnr and year
        permanent_sample['mrkt_share'] = permanent_sample['op_inc'] \
            / permanent_sample['op_inc_ind_yr']
```

```
In [ ]: # CAPEX
        permanent_sample['capex'] = permanent_sample.groupby('orgnr')\
            ['tan_fixed_assets'].diff() + permanent_sample['dep']
        # CAPEX
        full_sample['capex'] = full_sample.groupby('orgnr')\
            ['tan_fixed_assets'].diff() + full_sample['dep']
```

```
In [ ]: # Growth of sales
        full_sample['growth'] = full_sample.groupby('orgnr')\
            ['rev'].pct_change()
        # Growth of sales
        permanent_sample['growth'] = permanent_sample.groupby\
            ('orgnr')['rev'].pct_change()
```

```
In [ ]: # Interest Rate
        #permanent_sample['int_rate'] = permanent_sample['int_expe']
        # / permanent_sample['lt_liab']
        #full_sample['int_rate'] = full_sample['int_expe'] /
        # full_sample['lt_liab']

        # Create df with interast rates
        #int_rates = full_sample[['orgnr', 'year', 'industry', 'int_rate']]
        #int_rates.dropna(inplace=True)
        #int_rates.reset_index(drop=True)

            # Winsorize
        #int_rates['int_rate_win'] = winsorize(int_rates['int_rate'], (0.01, 0.05))
            # Compute industry interest rate
        #int_rates['ind_int_rate'] = int_rates.groupby('industry')['int_rate_win'].mean()

        #int_rates['ind_int_rate'].describe()
```

In [ ]:
```
# Cost of borrowing by industry
#ind_int_rate = full_sample.groupby('industry')
# ['int_rate_win'].mean()
#full_sample = full_sample.merge(ind_int_rate,
# on='industry', how='left')
#full_sample.rename(columns={'int_rate_win_x':
# 'int_rate_win'}, inplace='True')
#full_sample.rename(columns={'int_rate_win_y':
# 'ind_int_rate'}, inplace='True')

# Cost of borrowing by industry
#ind_int_rate = permanent_sample.groupby('industry')
# ['int_rate_win'].mean()
#permanent_sample = permanent_sample.merge(ind_int_rate,
# on='industry', how='left')
#permanent_sample.rename(columns={'int_rate_win_x': '
# int_rate_win'}, inplace='True')
#permanent_sample.rename(columns={'int_rate_win_y':
# 'ind_int_rate'}, inplace='True')
```

In [ ]:
```
# Sort the DataFrame by 'orgnr' and 'year'
full_sample = full_sample.sort_values(['orgnr', 'year'])

# Calculate the lagged 'lt_liab' for each firm
full_sample['lag_lt_liab'] = full_sample.groupby('orgnr')['lt_liab'].shift()

# Divide 'int_expe' by the lagged 'lt_liab' for each firm
full_sample['int_expe_divided'] = full_sample['int_expe']\
     / full_sample['lag_lt_liab']
```

In [ ]:
```
# Bank rate
#permanent_sample['bank_rate'] = permanent_sample['int_rate']
#  / permanent_sample['int_rate'].mean()
# Bank rate
#full_sample['bank_rate'] = full_sample['int_rate']
# / permanent_sample['int_rate'].mean()
```

Productivity measures

In [ ]:
```
# Full Sample
full_sample['lg_output']  = np.log(full_sample['output'])

    # Output to total_assets, capital, and emp_comp
full_sample['output_a'] = full_sample['output'] / full_sample['total_assets']
full_sample['output_k'] = full_sample['output'] / full_sample['capital']
full_sample['output_l'] = full_sample['output'] / full_sample['emp_comp']

full_sample['a_output'] = full_sample['total_assets'] / full_sample['output']
full_sample['k_output'] = full_sample['capital'] / full_sample['output']
full_sample['l_output'] = full_sample['emp_comp'] / full_sample['output']

# Permanent Sample
```

```
    # Output to total_assets, capital, and emp_comp
permanent_sample['output_a'] = permanent_sample['output'] \
    / permanent_sample['total_assets']
permanent_sample['output_k'] = permanent_sample['output'] \
    / permanent_sample['capital']
permanent_sample['output_l'] = permanent_sample['output'] \
    / permanent_sample['emp_comp']

permanent_sample['a_output'] = permanent_sample['total_assets']\
     / permanent_sample['output']
permanent_sample['k_output'] = permanent_sample['capital'] \
    / permanent_sample['output']
permanent_sample['l_output'] = permanent_sample['emp_comp'] \
    / permanent_sample['output']
```

In [ ]:
```
# Logging the emp_comp and output: Permanent Sample

# Log emp_comp
permanent_sample['log_emp_comp']                \
    = np.log(permanent_sample['emp_comp'])
# Log the output variable
permanent_sample['log_output']                \
    = np.log(permanent_sample['output'])
# Calculating the output growth
permanent_sample['log_output_growth']         \
    = permanent_sample.groupby('orgnr')['log_output'].diff()

# Normalizing the growth, so it is between -1 and 1:
min_val_y                                       \
    =  permanent_sample['log_output'].min()
max_val_y                                      \
     =  permanent_sample['log_output'].max()
permanent_sample['norm_output_growth']          \
    = (permanent_sample['log_output'] - min_val_y) /\
         (max_val_y - min_val_y) * 2 - 1

# Normalizing the emp_comp
min_val_l                                       = \
     permanent_sample['log_emp_comp'].min()
max_val_l                                       =  \
    permanent_sample['log_emp_comp'].max()
permanent_sample['norm_emp_comp']               = \
    (permanent_sample['log_emp_comp'] - min_val_l) /\
         (max_val_l - min_val_l) * 2 - 1
```

In [ ]:
```
# Logging the emp_comp and output: Permanent Sample

# Log emp_comp
full_sample['log_emp_comp']                = \
    np.log(full_sample['emp_comp'])
# Log the output variable
full_sample['log_output']                = \
    np.log(full_sample['output'])
# Calculating the output growth
full_sample['log_output_growth']         = \
    full_sample.groupby('orgnr')['log_output'].diff()

# Normalizing the growth, so it is between -1 and 1:
min_val_y                                               = \
     full_sample['log_output'].min()
max_val_y                                               = \
     full_sample['log_output'].max()
full_sample['norm_output_growth']                  = \
    (full_sample['log_output'] - min_val_y) / \
        (max_val_y - min_val_y) * 2 - 1

# Normalizing the emp_comp
min_val_l                                               =  \
    full_sample['log_emp_comp'].min()
max_val_l                                               = \
     full_sample['log_emp_comp'].max()
full_sample['norm_emp_comp']                       = \
    (full_sample['log_emp_comp'] - min_val_l) / \
        (max_val_l - min_val_l) * 2 - 1
```

In [ ]:
```
# Profit Share
permanent_sample['profit_share'] = \
    permanent_sample['net_profit']/permanent_sample['output']
full_sample['profit_share'] = \
    full_sample['net_profit']/full_sample['output']
```

In [ ]:
```
# Creating a counterfactual profit share
permanent_sample.sort_values(['orgnr', 'year'], inplace=True)

# time series average for each firm
permanent_sample['time_series_average_labor_share']  \
      = permanent_sample.groupby('orgnr')['l_output'].\
        transform('mean')
permanent_sample['time_series_average_profit_share']  \
    = permanent_sample.groupby('orgnr')['profit_share'].\
        transform('mean')

# the "new" labor cost
permanent_sample['new_labor_cost']                     \
     = permanent_sample['output']*permanent_sample\
        ['time_series_average_labor_share']


# the π hat: Net profit + old labor cost - new labor cost
permanent_sample['pi_hat']                             \
     = permanent_sample['net_profit'] + permanent_sample\
        ['emp_comp'] - permanent_sample['new_labor_cost']
```

In [ ]:
```
# Creating a counterfactual profit share
full_sample.sort_values(['orgnr', 'year'], inplace=True)

# time series average for each firm
full_sample['time_series_average_labor_share']     = \
    full_sample.groupby('orgnr')['l_output'].\
        transform('mean')
```

```
full_sample['time_series_average_profit_share']  = \
    full_sample.groupby('orgnr')['profit_share'].\
        transform('mean')

# the "new" labor cost
full_sample['new_labor_cost']                    = \
    full_sample['output']*full_sample\
        ['time_series_average_labor_share']


# the π hat: Net profit + old labor cost - new labor cost
full_sample['pi_hat']                           \
     = full_sample['net_profit'] + full_sample['emp_comp']\
         - full_sample['new_labor_cost']
```

In [ ]:
```
# Deviation of Labor and Profit Shares from Time Series Mean
permanent_sample['labor_deviation']    = \
    permanent_sample['l_output'] - permanent_sample\
        ['time_series_average_labor_share']
permanent_sample['profit_deviation']  = \
    permanent_sample['profit_share'] - permanent_sample\
        ['time_series_average_profit_share']
```

In [ ]:
```
# Deviation of Labor and Profit Shares from Time Series Mean
full_sample['labor_deviation']    = full_sample['l_output']\
     - full_sample['time_series_average_labor_share']
full_sample['profit_deviation']  = full_sample['profit_share']\
     - full_sample['time_series_average_profit_share']
```

In [ ]:
```
permanent_sample.loc[permanent_sample['private'] == 1, 'type'] = 'private'
permanent_sample.loc[permanent_sample['public'] == 1, 'type'] = 'public'
permanent_sample.loc[permanent_sample['public_listed'] == 1, 'type'] = 'listed'
permanent_sample.loc[permanent_sample['public_nonlisted'] == 1, \
    'type'] = 'nonlisted'

full_sample.loc[full_sample['private'] == 1, 'type'] = 'private'
full_sample.loc[full_sample['public'] == 1, 'type'] = 'public'
full_sample.loc[full_sample['public_listed'] == 1, 'type'] = 'listed'
full_sample.loc[full_sample['public_nonlisted'] == 1, 'type'] = 'nonlisted'
```

In [ ]:
```
# Save pandas dataframe as parquet file
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(permanent_sample)

# Write table to a parquet file with Brotli compression

output_file = 'N:/durable/BIStudents/IssamSheikh/data/\
    clean/permanent_sample.parquet'
pq.write_table(table, output_file)
```

In [ ]:
```
# Save pandas dataframe as parquet file
# Convert datafram to Apach arrow table
table = pa.Table.from_pandas(full_sample)

# Write table to a parquet file with Brotli compression
```

```
output_file = 'N:/durable/BIStudents/IssamSheikh/data\
    /clean/full_sample.parquet'
pq.write_table(table, output_file)
```

In [ ]:
```
pip install nbconvert
```

In [ ]:
```
# Import needed libraries
import pandas as pd
import numpy as np
import pyarrow.parquet as pq
pd.set_option('mode.use_inf_as_na', True)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline
from tabulate import tabulate
from scipy.stats.mstats import winsorize
import pyarrow as pa

# Display 6 columns for viewing pupuses
pd.set_option('display.max_columns', 10)
# Reduce decimals points to 2
pd.options.display.float_format = '{:,.2f}'.format
```

In [ ]:
```
# Read datasets:
permanent_sample     = pd.read_parquet\
    ('N:/durable/BIStudents/IssamSheikh/data/clean\
        /permanent_sample.parquet')
full_sample          = pd.read_parquet\
    ('N:/durable/BIStudents/IssamSheikh/data/clean\
        /full_sample.parquet')
```

In [ ]:
```
permanent_sample['public'] = permanent_sample['public_listed']\
     + permanent_sample['public_nonlisted']
full_sample['public'] = full_sample['public_listed'] \
    + full_sample['public_nonlisted']
```

In [ ]:
```
# Empirical Analysis
df_private      = permanent_sample[permanent_sample['private'] == 1]
df_nonlist      =  permanent_sample[permanent_sample['public_nonlisted'] == 1]
df_list         =  permanent_sample[permanent_sample['public_listed'] == 1]
df_public       = permanent_sample[permanent_sample['public'] == 1]
```

In [ ]:
```
df_private_full      = full_sample[full_sample['private'] == 1]
df_nonlist_full      =  full_sample[full_sample['public_nonlisted'] == 1]
df_list_full         =  full_sample[full_sample['public_listed'] == 1]
df_public_full       = full_sample[full_sample['public'] == 1]
df_private_large_full     = full_sample[(full_sample['private'] == 1) \
    & (full_sample['total_assets'] > 3000000000)]
```

In [ ]:
```
# Create parquets for all subsamples
table1 = pa.Table.from_pandas(df_private )
table2 = pa.Table.from_pandas(df_nonlist)
table3 = pa.Table.from_pandas(df_list)
table4 = pa.Table.from_pandas(df_public)
table5 = pa.Table.from_pandas(df_private_full)
table6 = pa.Table.from_pandas(df_nonlist_full)
```

```
table7 = pa.Table.from_pandas(df_list_full)
table8 = pa.Table.from_pandas(df_public_full)
table9 = pa.Table.from_pandas(df_private_large_full)
```

```
In [ ]: output_file1 = 'N:/durable/BIStudents/IssamSheikh/data/clean/final/\
            permanent_sample/df_private.parquet'
        output_file2 = 'N:/durable/BIStudents/IssamSheikh/data/clean/final/\
            permanent_sample/df_nonlist.parquet'
        output_file3 = 'N:/durable/BIStudents/IssamSheikh/data/clean/final/\
            permanent_sample/df_list.parquet'
        output_file4 = 'N:/durable/BIStudents/IssamSheikh/data/clean/final/\
            permanent_sample/df_public.parquet'
        output_file5 = 'N:/durable/BIStudents/IssamSheikh/data/clean/final/\
            full_sample/df_private_full.parquet'
        output_file6 = 'N:/durable/BIStudents/IssamSheikh/data/clean/final/\
            full_sample/df_nonlist_full.parquet'
        output_file7 = 'N:/durable/BIStudents/IssamSheikh/data/clean/final/\
            full_sample/df_list_full.parquet'
        output_file8 = 'N:/durable/BIStudents/IssamSheikh/data/clean/final/\
            full_sample/df_public_full.parquet'
        output_file9 = 'N:/durable/BIStudents/IssamSheikh/data/clean/final/\
            full_sample/df_private_large_full.parquet'
```

```
In [ ]: pq.write_table(table1, output_file1)
        pq.write_table(table2, output_file2)
        pq.write_table(table3, output_file3)
        pq.write_table(table4, output_file4)
        pq.write_table(table5, output_file5)
        pq.write_table(table6, output_file6)
        pq.write_table(table7, output_file7)
        pq.write_table(table8, output_file8)
        pq.write_table(table9, output_file9)
```

```
In [ ]: pip install nbconvert
```

```
In [ ]: import pandas as pd
        import numpy as np
        from scipy.stats.mstats import winsorize
        from IPython.display import display
        import pyarrow as pa
        import pyarrow.parquet as pq
        from tabulate import tabulate
        import matplotlib.pyplot as plt
        from scipy.stats.mstats import winsorize

        # Display 6 columns for viewing pupuses
        pd.set_option('display.max_columns', 10)
        # Reduce decimals points to 2
        pd.options.display.float_format = '{:,.2f}'.format
```

```
In [ ]: # Read the data
        df_permanent = pd.read_parquet('N:/durable/BIStudents/IssamSheikh\
            /data/clean/permanent_sample.parquet')
        df_full      = pd.read_parquet('N:/durable/BIStudents/IssamSheikh\
            /data/clean/full_sample.parquet')
```

Generate subsamples with big private firms and public firms

```
In [ ]: # Generating publc firms data
        df_pub = df_full[df_full['public'] == 1]
        df_pri = df_full[df_full['private'] == 1]

        # Restricting df_pri so it only contains big firms
        df_pri = df_pri[df_pri['total_assets'] > 3000000000]

        # Variables of Interest
        df_pub = df_pub[['orgnr', 'year', 'rev', 'net_profit', 'total_assets', 'ROE', \
            'ROA', 'output_l', 'output_k', 'output_a',\
                         'financial_leverage', 'liquidity_ratio']]
        df_pri = df_pri[['orgnr', 'year', 'rev', 'net_profit', 'total_assets', 'ROE',\
             'ROA', 'output_l', 'output_k', 'output_a', \
                         'financial_leverage', 'liquidity_ratio']]
```

Winsorizing

Big Private

```
In [ ]: df_pri['total_assets_win'] = winsorize(df_pri['total_assets'], \
            (0.01, 0.01))
```

```
df_pri['ROA_win']           = winsorize(df_pri['ROA'], (0.01, 0.01))
df_pri['output_l_win'] = winsorize(df_pri['output_l'], (0.01, 0.01))
df_pri['output_k_win'] = winsorize(df_pri['output_k'], (0.01, 0.01))
df_pri['output_a_win'] = winsorize(df_pri['output_a'], (0.01, 0.01))
df_pri['financial_leverage_win'] = winsorize(df_pri['financial_leverage'], \
    (0.01, 0.01))
df_pri['liquidity_ratio_win'] = winsorize(df_pri['liquidity_ratio'], \
    (0.01, 0.01))
```

Public

```
In [ ]: df_pub['total_assets_win'] = winsorize(df_pub['total_assets'], \
            (0.01, 0.01))
        df_pub['ROA_win']           = winsorize(df_pub['ROA'], (0.01, 0.01))
        df_pub['output_l_win'] = winsorize(df_pub['output_l'], (0.01, 0.01))
        df_pub['output_k_win'] = winsorize(df_pub['output_k'], (0.01, 0.01))
        df_pub['output_a_win'] = winsorize(df_pub['output_a'], (0.01, 0.01))
        df_pub['financial_leverage_win'] = winsorize(df_pub['financial_leverage'], \
            (0.01, 0.01))
        df_pub['liquidity_ratio_win'] = winsorize(df_pub['liquidity_ratio'], \
            (0.01, 0.01))
```

Create Table for Comparison

```
In [ ]: bigpri = pd.DataFrame({
        'mean': [df_pri['total_assets_win'].mean(),         \
            df_pri['ROA_win'].mean(),         df_pri['output_l_win'].mean(),         \
                df_pri['output_k_win'].mean(),         df_pri['output_a_win'].mean(),
                    df_pri['financial_leverage_win'].mean(),                 df_pri['liquid
                        ratio_win'].mean()],
        'st.d': [df_pri['total_assets_win'].std(),          \
            df_pri['ROA_win'].std(),          df_pri['output_l_win'].std(),          \
                df_pri['output_k_win'].std(),          df_pri['output_a_win'].std(),
                    df_pri['financial_leverage_win'].std(),                  df_pri['liquid
                        ratio_win'].std()],
        'p10'  : [df_pri['total_assets_win'].quantile(.10),  \
            df_pri['ROA_win'].quantile(.10), df_pri['output_l_win'].quantile(.10), \
                df_pri['output_k_win'].quantile(.10), df_pri['output_a_win'].quantile(.10),
                    df_pri['financial_leverage_win'].quantile(.10),          df_pri['liquid
                        ratio_win'].quantile(.10)],
        'p25'  : [df_pri['total_assets_win'].quantile(.25),  \
            df_pri['ROA_win'].quantile(.25), df_pri['output_l_win'].quantile(.25), \
                df_pri['output_k_win'].quantile(.25), df_pri['output_a_win'].quantile(.25),
                    df_pri['financial_leverage_win'].quantile(.25),          df_pri['liquid
                        ratio_win'].quantile(.25)],
        'p50'  : [df_pri['total_assets_win'].quantile(.50),  \
            df_pri['ROA_win'].quantile(.50), df_pri['output_l_win'].quantile(.50), \
                df_pri['output_k_win'].quantile(.50), df_pri['output_a_win'].quantile(.50),
                    df_pri['financial_leverage_win'].quantile(.50),          df_pri['liquid
                        _ratio_win'].quantile(.50)],
        'p75'  : [df_pri['total_assets_win'].quantile(.75),  \
            df_pri['ROA_win'].quantile(.75), df_pri['output_l_win'].quantile(.75), \
        df_pri['output_k_win'].quantile(.75), df_pri['output_a_win'].quantile(.75),
             df_pri['financial_leverage_win'].quantile(.75),          df_pri['liqui
                y_ratio_win'].quantile(.75)],
'p90' : [df_pri['total_assets_win'].quantile(.90),  \
    df_pri['ROA_win'].quantile(.90), df_pri['output_l_win'].quantile(.90), \
        df_pri['output_k_win'].quantile(.90), df_pri['output_a_win'].quantile(.90),
            df_pri['financial_leverage_win'].quantile(.90),          df_pri['liquid
                _ratio_win'].quantile(.90)],
}, index = ['Assets', 'ROA_win', 'y/l', 'y/k', 'y/a', \
    'Financial Leverage', \
    'Liquidity Ratio'])

bigpub    = pd.DataFrame({
'mean': [df_pub['total_assets_win'].mean(),          \
    df_pub['ROA_win'].mean(),        df_pub['output_l_win'].mean(),        \
        df_pub['output_k_win'].mean(),        df_pub['output_a_win'].mean(),
            df_pub['financial_leverage_win'].mean(),                 df_pub['liquid
                ratio_win'].mean()],
'st.d': [df_pub['total_assets_win'].std(),           \
    df_pub['ROA_win'].std(),         df_pub['output_l_win'].std(),          \
        df_pub['output_k_win'].std(),         df_pub['output_a_win'].std(),
            df_pub['financial_leverage_win'].std(),                  df_pub['liquid
                ratio_win'].std()],
'p10' : [df_pub['total_assets_win'].quantile(.10),  \
    df_pub['ROA_win'].quantile(.10), df_pub['output_l_win'].quantile(.10), \
        df_pub['output_k_win'].quantile(.10), df_pub['output_a_win'].quantile(.10),
            df_pub['financial_leverage_win'].quantile(.10),          df_pub['liquid
                ratio_win'].quantile(.10)],
'p25' : [df_pub['total_assets_win'].quantile(.25),  \
    df_pub['ROA_win'].quantile(.25), df_pub['output_l_win'].quantile(.25), \
        df_pub['output_k_win'].quantile(.25), df_pub['output_a_win'].quantile(.25),
            df_pub['financial_leverage_win'].quantile(.25),          df_pub['liquid
                ratio_win'].quantile(.25)],
'p50' : [df_pub['total_assets_win'].quantile(.50),  \
    df_pub['ROA_win'].quantile(.50), df_pub['output_l_win'].quantile(.50), \
        df_pub['output_k_win'].quantile(.50), df_pub['output_a_win'].quantile(.50),
            df_pub['financial_leverage_win'].quantile(.50),          df_pub['liquid
                _ratio_win'].quantile(.50)],
'p75' : [df_pub['total_assets_win'].quantile(.75),  \
    df_pub['ROA_win'].quantile(.75), df_pub['output_l_win'].quantile(.75), \
        df_pub['output_k_win'].quantile(.75), df_pub['output_a_win'].quantile(.75),
             df_pub['financial_leverage_win'].quantile(.75),          df_pub['liqui
                y_ratio_win'].quantile(.75)],
'p90' : [df_pub['total_assets_win'].quantile(.90),  \
    df_pub['ROA_win'].quantile(.90), df_pub['output_l_win'].quantile(.90), \
        df_pub['output_k_win'].quantile(.90), df_pub['output_a_win'].quantile(.90),
            df_pub['financial_leverage_win'].quantile(.90),          df_pub['liquid
                _ratio_win'].quantile(.90)],
}, index = ['Assets', 'ROA_win', 'y/l', 'y/k', 'y/a', \
    'Financial Leverage', \
    'Liquidity Ratio'])

# Export to Latex:

bigprilat                = bigpri.to_latex(index=True)
title                    = 'bigpri'
```

```
extension                = '.tex'
path                     = 'N:/durable/file-export/IssamSheikh/\
    S3_listing/tables/' + title + extension
with open(path, 'w') as f:
    f.write(bigprilat)

bigpublat                = bigpub.to_latex(index=True)
title                    = 'bigpub'
extension                = '.tex'
path                     = 'N:/durable/file-export/IssamSheikh/\
    S3_listing/tables/' + title + extension
with open(path, 'w') as f:
    f.write(bigpublat)
```

In [ ]:
```
print(bigpub.to_markdown())
```

In [ ]:
```

```

```
In [ ]: pip install nbconvert
```

```
In [ ]: # Import needed libraries
        import pandas as pd
        import numpy as np
        import pyarrow.parquet as pq
        pd.set_option('mode.use_inf_as_na', True)
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.preprocessing import MinMaxScaler
        %matplotlib inline
        from tabulate import tabulate
        from scipy.stats.mstats import winsorize
        import pyarrow as pa
        import re

        # Display 6 columns for viewing pupuses
        pd.set_option('display.max_columns', 10)
        # Reduce decimals points to 2
        pd.options.display.float_format = '{:,.10f}'.format
```

```
In [ ]: # Read the data
        df_permanent = pd.read_parquet('N:/durable/BIStudents/\
            IssamSheikh/data/clean/permanent_sample.parquet')
        df_private  = pd.read_parquet('N:/durable/BIStudents\
            /IssamSheikh/data/clean/final/permanent_sample/df_private.parquet')
        df_nonlist  = pd.read_parquet('N:/durable/BIStudents/\
            IssamSheikh/data/clean/final/permanent_sample/df_nonlist.parquet')
        df_list     = pd.read_parquet('N:/durable/BIStudents/\
            IssamSheikh/data/clean/final/permanent_sample/df_list.parquet')
        df_public   = pd.read_parquet('N:/durable/BIStudents/\
            IssamSheikh/data/clean/final/permanent_sample/df_public.parquet')
```

```
In [ ]: df_permanent['public'] = df_permanent['public_listed'] +\
         df_permanent['public_nonlisted']
```

Winsorizing

Level variables

Permanent Sample

```
In [ ]: ## output, labor, capital, equity, net profit, pi_hat
        # Winsorize Output
        df_permanent['output_win'] = winsorize(df_permanent['output'], \
            (0.01, 0.01))
        df_permanent['lg_output_win'] = np.log(df_permanent['output_win'])\

        df_permanent['lg_output_growth']        = df_permanent.groupby('orgnr')\
    ['lg_output_win'].diff()
#df_permanent['lg_output_growth'].hist()

# Winsorize Labor
df_permanent['emp_comp_win'] = winsorize(df_permanent\
    ['emp_comp'], (0.01, 0.01))
df_permanent['lg_emp_comp_win'] = np.log(df_permanent\
    ['emp_comp_win'])
#df_permanent['lg_emp_comp_win'].hist()

# Winsorize Capital
df_permanent['capital_win'] = winsorize(df_permanent\
    ['capital'], (0.01, 0.01))
df_permanent['lg_capital_win'] = np.log(df_permanent\
    ['capital_win'])
#df_permanent['lg_capital_win'].hist()

# Winsorize Equity
df_permanent['equity_win'] = winsorize(df_permanent\
    ['equity'], (0.01, 0.01))
df_permanent['lg_equity_win'] = np.log(df_permanent\
    ['equity_win'])
#df_permanent['lg_equity_win'].hist()

# Winsorize Net profit
df_permanent['net_profit_win'] = winsorize(df_permanent\
    ['net_profit'], (0.01, 0.01))
df_permanent['lg_net_profit_win'] = np.log(df_permanent\
    ['net_profit_win'])
#df_permanent['lg_net_profit_win'].hist()

# Winsorize Counterfactual profit share
df_permanent['pi_hat_win'] = winsorize(df_permanent\
    ['pi_hat'], (0.01, 0.01))
df_permanent['lg_pi_hat_win'] = np.log(df_permanent\
    ['pi_hat_win'])
#df_permanent['pi_hat'].hist()

# Winsorize total assets
df_permanent['total_assets_win'] = winsorize(df_permanent\
    ['total_assets'], (0.01, 0.01))
df_permanent['total_assets_win'].hist()
```

Private firms

In [ ]:
```
## output, labor, capital, equity, net profit, pi_hat
# Winsorize Output
df_private['output_win'] = winsorize(df_private['output'], \
    (0.01, 0.01))
df_private['lg_output_win'] = np.log(df_private\
    ['output_win'])
df_private['lg_output_growth']        = df_private.groupby\
    ('orgnr')['lg_output_win'].diff()
#df_private['lg_output_win'].hist()
```

```
# Winsorize Labor
df_private['emp_comp_win'] = winsorize(df_private\
    ['emp_comp'], (0.01, 0.01))
df_private['lg_emp_comp_win'] = np.log(df_private\
    ['emp_comp_win'])
#df_private['lg_emp_comp_win'].hist()

# Winsorize Capital
df_private['capital_win'] = winsorize(df_private\
    ['capital'], (0.01, 0.01))
df_private['lg_capital_win'] = np.log(df_private\
    ['capital_win'])
#df_private['lg_capital_win'].hist()

# Winsorize Equity
df_private['equity_win'] = winsorize(df_private\
    ['equity'], (0.01, 0.01))
df_private['lg_equity_win'] = np.log(df_private[\
    'equity_win'])
#df_private['lg_equity_win'].hist()

# Winsorize Net profit\

df_private['net_profit_win'] = winsorize(df_privat\
    e['net_profit'], (0.01, 0.01))
df_private['lg_net_profit_win'] = np.log(df_private\
    ['net_profit_win'])
#df_private['lg_net_profit_win'].hist()

# Winsorize Counterfactual profit share
df_private['pi_hat_win'] = winsorize(df_private\
    ['pi_hat'], (0.01, 0.01))
df_private['lg_pi_hat_win'] = np.log(df_private\
    ['pi_hat_win'])
#df_private['pi_hat'].hist()
```

Public firms

```
In [ ]: # Winsorize Output
# output, labor, capital, equity, net profit
df_public['output_win'] = winsorize(df_public\
    ['output'], (0.01, 0.01))
df_public['lg_output_win'] = np.log(df_public\
    ['output_win'])
df_public['lg_output_growth']        = df_public.\
    groupby('orgnr')['lg_output_win'].diff()
#df_public['lg_output_win'].hist()

# Winsorize Labor
# output, labor, capital, equity, net profit
df_public['emp_comp_win'] = winsorize(df_public\
    ['emp_comp'], (0.01, 0.01))
df_public['lg_emp_comp_win'] = np.log(df_public\
    ['emp_comp_win'])
#df_public['lg_emp_comp_win'].hist()
```

```
# Winsorize Capital
# output, labor, capital, equity, net profit
df_public['capital_win'] = winsorize(df_public\
    ['capital'], (0.01, 0.01))
df_public['lg_capital_win'] = np.log(df_public\
    ['capital_win'])
#df_public['lg_capital_win'].hist()

# Winsorize Equity
# output, labor, capital, equity, net profit
df_public['equity_win'] = winsorize(df_public\
    ['equity'], (0.01, 0.01))
df_public['lg_equity_win'] = np.log(df_public\
    ['equity_win'])
#df_public['lg_equity_win'].hist()

# Winsorize Net profit
# output, labor, capital, equity, net profit
df_public['net_profit_win'] = winsorize(df_public\
    ['net_profit'], (0.01, 0.01))
df_public['lg_net_profit_win'] = np.log(df_public\
    ['net_profit'])
#df_public['lg_net_profit_win'].hist()
```

Listed firms

```
In [ ]: # Winsorize Output
# output, labor, capital, equity, net profit
df_list['output_win'] = winsorize(df_list['output'],\
     (0.01, 0.01))
df_list['lg_output_win'] = np.log(df_list\
    ['output_win'])
df_list['lg_output_growth']        = df_list.groupby\
    ('orgnr')['lg_output_win'].diff()
#df_list['lg_output_win'].hist()

# Winsorize Labor
# output, labor, capital, equity, net profit
df_list['emp_comp_win'] = winsorize(df_list['emp_comp']\
    , (0.01, 0.01))
df_list['lg_emp_comp_win'] = np.log(df_lis\
    t['emp_comp_win'])
#df_list['lg_emp_comp_win'].hist()

# Winsorize Capital
# output, labor, capital, equity, net profit
df_list['capital_win'] = winsorize(df_list['capital']\
    , (0.01, 0.01))
df_list['lg_capital_win'] = np.log(df_list['capital_win'])
#df_list['lg_capital_win'].hist()

# Winsorize Equity
# output, labor, capital, equity, net profit
df_list['equity_win'] = winsorize(df_list['equity'], \
    (0.01, 0.01))
df_list['lg_equity_win'] = np.log(df_lis\
    t['equity_win'])
#df_list['lg_equity_win'].hist()

# Winsorize Net profit
# output, labor, capital, equity, net profit
df_list['net_profit_win'] = winsorize(df_list\
    ['net_profit'], (0.01, 0.01))
df_list['lg_net_profit_win'] = np.log(df_list\
    ['net_profit'])
#df_list['lg_net_profit_win'].hist()
```

Non-listed firms

```
In [ ]: # Winsorize Output
# output, labor, capital, equity, net profit
df_nonlist['output_win'] = winsorize(\
    df_nonlist['output'], (0.01, 0.01))
df_nonlist['lg_output_win'] = np.log(\
    df_nonlist['output_win'])
df_nonlist['lg_output_growth']        = \
    df_nonlist.groupby('orgnr')['lg_output_win']\
        .diff()
#df_nonlist['lg_output_win'].hist()

# Winsorize Labor
# output, labor, capital, equity, net profit
df_nonlist['emp_comp_win'] = winsorize\
    (df_nonlist['emp_comp'], (0.01, 0.01))
df_nonlist['lg_emp_comp_win'] = np.log\
    (df_nonlist['emp_comp_win'])
#df_nonlist['lg_emp_comp_win'].hist()

# Winsorize Capital
# output, labor, capital, equity, net profit
df_nonlist['capital_win'] = winsorize(\
    df_nonlist['capital'], (0.01, 0.01))
df_nonlist['lg_capital_win'] = np.log\
    (df_nonlist['capital_win'])
#df_nonlist['lg_capital_win'].hist()

# Winsorize Equity
# output, labor, capital, equity, net profit
df_nonlist['equity_win'] = winsorize(\
    df_nonlist['equity'], (0.01, 0.01))
df_nonlist['lg_equity_win'] = np.log\
    (df_nonlist['equity_win'])
#df_nonlist['lg_equity_win'].hist()

# Winsorize Net profit
# output, labor, capital, equity, net profit
df_nonlist['net_profit_win'] = winsorize\
    (df_nonlist['net_profit'], (0.01, 0.01))
df_nonlist['lg_net_profit_win'] = np.log\
    (df_nonlist['net_profit'])
#df_nonlist['lg_net_profit_win'].hist()
```

Ratio variables

Returns

Permanent Sample

In [ ]:
```
## ROE, ROA
# Winsorize ROE
df_permanent['ROE_win'] = winsorize(\
    df_permanent['ROE'], (0.01, 0.01))
#df_permanent['ROE_win'].hist()

# Winsorize ROA
df_permanent['ROA_win'] = winsorize(\
    df_permanent['ROA'], (0.01, 0.01))
#df_permanent['ROA'].hist()
```

Private firms

In [ ]:
```
## ROE, ROA
# Winsorize ROE
df_private['ROE_win'] = winsorize(\
    df_private['ROE'], (0.01, 0.01))
#df_private['ROE_win'].hist()

# Winsorize ROA
df_private['ROA_win'] = winsorize(\
    df_private['ROA'], (0.01, 0.01))
#df_private['ROA'].hist()
```

Public firms

In [ ]:
```
## ROE, ROA
# Winsorize ROE
df_public['ROE_win'] = winsorize(\
    df_public['ROE'], (0.01, 0.01))
#df_public['ROE_win'].hist()

# Winsorize ROA
df_public['ROA_win'] = winsorize(d\
    f_public['ROA'], (0.01, 0.01))
#df_public['ROA_win'].hist()
```

Listed firms

In [ ]:
```
## ROE, ROA
# Winsorize ROE
df_list['ROE_win'] = winsorize(df_list\
    ['ROE'], (0.01, 0.01))
#df_list['ROE'].hist()

# Winsorize ROA
df_list['ROA_win'] = winsorize(df_list\
    ['ROA'], (0.01, 0.01))
#df_list['ROA_win'].hist()
```

Non-listed firms

```
In [ ]: ## ROE, ROA
        # Winsorize ROE
        df_nonlist['ROE_win'] = winsorize(\
            df_nonlist['ROE'], (0.01, 0.01))
        #df_nonlist['ROE'].hist()

        # Winsorize ROA
        df_nonlist['ROA_win'] = winsorize(\
            df_nonlist['ROA'], (0.01, 0.01))
        #df_nonlist['ROA_win'].hist()
```

Productivity

Permanent Sample

```
In [ ]: ## l_output, k_output, profit_share
        # Winsorize l_output
        df_permanent['l_output_win'] = winsorize(\
            df_permanent['l_output'], (0.01, 0.01))
        #df_permanent['l_output_win'].hist()

        # Winsorize k_output
        df_permanent['k_output_win'] = winsorize(\
            df_permanent['k_output'], (0.01, 0.01))
        #df_permanent['k_output'].hist()

        # Winsorize profit_share
        df_permanent['profit_share_win'] = winsorize(\
            df_permanent['profit_share'], (0.01, 0.01))
        #df_permanent['profit_share_win'].hist()
```

Private firms

```
In [ ]: ## l_output, k_output, profit_share
        # Winsorize l_output
        df_private['l_output_win'] = winsorize(df_private[\
            'l_output'], (0.01, 0.01))
        #df_private['l_output_win'].hist()

        # Winsorize k_output
        df_private['k_output_win'] = winsorize(df_private\
            ['k_output'], (0.01, 0.01))
        #df_private['k_output'].hist()
```

```
# Winsorize profit_share
df_private['profit_share_win'] = winsorize(df_private\
    ['profit_share'], (0.01, 0.01))
#df_private['profit_share_win'].hist()
```

Public firms

In [ ]:
```
## l_output, k_output, profit_share
# Winsorize l_output
df_public['l_output_win'] = winsorize(df_public\
    ['l_output'], (0.01, 0.01))
#df_public['l_output_win'].hist()

# Winsorize k_output
df_public['k_output_win'] = winsorize(df_public\
    ['k_output'], (0.01, 0.01))
#df_public['k_output_win'].hist()

# Winsorize profit_share
df_public['profit_share_win'] = winsorize(df_public\
    ['profit_share'], (0.01, 0.01))
#df_public['profit_share_win'].hist()
```

Listed firms

In [ ]:
```
## l_output, k_output, profit_share
# Winsorize l_output
df_list['l_output_win'] = winsorize(df_list\
    ['l_output'], (0.01, 0.01))
#df_list['l_output_win'].hist()

# Winsorize k_output
df_list['k_output_win'] = winsorize(df_list[\
    'k_output'], (0.01, 0.01))
#df_list['k_output_win'].hist()

# Winsorize profit_share
df_list['profit_share_win'] = winsorize(df_list\
    ['profit_share'], (0.01, 0.01))
#df_list['profit_share_win'].hist()
```

Non_listed firms

In [ ]:
```
## l_output, k_output, profit_share
# Winsorize l_output
df_nonlist['l_output_win'] = winsorize(df_nonlist\
    ['l_output'], (0.01, 0.01))
#df_nonlist['l_output_win'].hist()

# Winsorize k_output
df_nonlist['k_output_win'] = winsorize(df_nonlis\
    t['k_output'], (0.01, 0.01))
#df_nonlist['k_output_win'].hist()
```

```
# Winsorize profit_share
df_nonlist['profit_share_win'] = winsorize(df_nonlist\
    ['profit_share'], (0.01, 0.01))
#df_nonlist['profit_share_win'].hist()
```

In [ ]:

```
```

Tables

Tables for raw data

In [ ]:

```
# Permanent Sample:
summary_private1              = pd.DataFrame({
 'mean': [df_private['output'].mean(),        \
    df_private['emp_comp'].mean(),        df_private['capital'].mean(),\
        df_private['equity'].mean(),        df_private['net_profit'].mean()],
 'st.d': [df_private['output'].std(),         \
    df_private['emp_comp'].std(),         df_private['capital'].std(),\
 df_private['equity'].std(),         df_private['net_profit'].std()],
 'p10' : [df_private['output'].quantile(.10),\
     df_private['emp_comp'].quantile(.10), df_private['capital'].\
        quantile(.10), df_private['equity'].quantile(.10), \
            df_private['net_profit'].quantile(.10)],
 'p25' : [df_private['output'].quantile(.25),\
     df_private['emp_comp'].quantile(.25), df_private['capital']\
        .quantile(.25), df_private['equity'].quantile(.25), df_private\
            ['net_profit'].quantile(.25)],
 'p50' : [df_private['output'].quantile(.50), \
    df_private['emp_comp'].quantile(.50), df_private['capital'].\
        quantile(.50), df_private['equity'].quantile(.50), df_private\
            ['net_profit'].quantile(.50)],
 'p75' : [df_private['output'].quantile(.75), \
    df_private['emp_comp'].quantile(.75), df_private['capital'].\
        quantile(.75), df_private['equity'].quantile(.75), df_privat\
            e['net_profit'].quantile(.75)],
 'p90' : [df_private['output'].quantile(.90),\
     df_private['emp_comp'].quantile(.90), df_private['capital'].\
        quantile(.90), df_private['equity'].quantile(.90), df_private\
            ['net_profit'].quantile(.90)],
}, index = ['Output', 'Labor', 'Capital', 'Equity', 'Net Profit'])

summary_public1              = pd.DataFrame({
 'mean': [df_public['output'].mean(),        \
    df_public['emp_comp'].mean(),        df_public['capital'].mean(),\
        df_public['equity'].mean(),        df_public['net_profit'].mean()],
 'st.d': [df_public['output'].std(),         \
    df_public['emp_comp'].std(),         df_public['capital'].std(),\
 df_public['equity'].std(),         df_public['net_profit'].std()],
 'p10' : [df_public['output'].quantile(.10),\
     df_public['emp_comp'].quantile(.10), df_public['capital'].\
        quantile(.10), df_public['equity'].quantile(.10), \
            df_public['net_profit'].quantile(.10)],
 'p25' : [df_public['output'].quantile(.25),\
     df_public['emp_comp'].quantile(.25), df_public['capital']\
        .quantile(.25), df_public['equity'].quantile(.25), df_public\
            ['net_profit'].quantile(.25)],
 'p50' : [df_public['output'].quantile(.50), \
    df_public['emp_comp'].quantile(.50), df_public['capital'].\
        quantile(.50), df_public['equity'].quantile(.50), df_public\
            ['net_profit'].quantile(.50)],
 'p75' : [df_public['output'].quantile(.75), \
    df_public['emp_comp'].quantile(.75), df_public['capital'].\
        quantile(.75), df_public['equity'].quantile(.75), df_privat\
            e['net_profit'].quantile(.75)],
 'p90' : [df_public['output'].quantile(.90),\
     df_public['emp_comp'].quantile(.90), df_public['capital'].\
        quantile(.90), df_public['equity'].quantile(.90), df_public\
            ['net_profit'].quantile(.90)],
}, index = ['Output', 'Labor', 'Capital', 'Equity', 'Net Profit'])


# Export to Latex
summary_private_latperm     = summary_private1.to_latex(index=True)
title    = 'summary_private_raw'
extension               = '.tex'
path     = 'N:/durable/file-export\
    /IssamSheikh/S1_empirical_analysis/tables/raw/' + title + extension

with open(path, 'w') as f:
    f.write(summary_private_latperm)

summary_public_latperm      = summary_public1.to_latex(index=True)
title    = 'summary_public_raw'
extension               = '.tex'
path     = 'N:/durable/file-export/\
    IssamSheikh/S1_empirical_analysis/tables/raw/' + title + extension
with open(path, 'w') as f:
    f.write(summary_public_latperm)

# permanent_sample: Creating Table 1: Summary Statistics Listed vs Nonlisted:

summary_public_nonlisted = pd.DataFrame({
 'mean': [df_nonlist['output'].mean(),        \
    df_nonlist['emp_comp'].mean(),        df_nonlist['capital'].mean(),\
        df_nonlist['equity'].mean(),        df_nonlist['net_profit'].mean()],
 'st.d': [df_nonlist['output'].std(),         \
    df_nonlist['emp_comp'].std(),         df_nonlist['capital'].std(),\
 df_nonlist['equity'].std(),         df_nonlist['net_profit'].std()],
 'p10' : [df_nonlist['output'].quantile(.10),\
     df_nonlist['emp_comp'].quantile(.10), df_nonlist['capital'].\
        quantile(.10), df_nonlist['equity'].quantile(.10), \
            df_nonlist['net_profit'].quantile(.10)],
 'p25' : [df_nonlist['output'].quantile(.25),\
     df_nonlist['emp_comp'].quantile(.25), df_nonlist['capital']\
        .quantile(.25), df_nonlist['equity'].quantile(.25), df_nonlist\
            ['net_profit'].quantile(.25)],
 'p50' : [df_nonlist['output'].quantile(.50), \
    df_nonlist['emp_comp'].quantile(.50), df_nonlist['capital'].\
        quantile(.50), df_nonlist['equity'].quantile(.50), df_nonlist\
            ['net_profit'].quantile(.50)],
 'p75' : [df_nonlist['output'].quantile(.75), \
    df_nonlist['emp_comp'].quantile(.75), df_nonlist['capital'].\
        quantile(.75), df_nonlist['equity'].quantile(.75), df_privat\
            e['net_profit'].quantile(.75)],
 'p90' : [df_nonlist['output'].quantile(.90),\
     df_nonlist['emp_comp'].quantile(.90), df_nonlist['capital'].\
        quantile(.90), df_nonlist['equity'].quantile(.90), df_nonlist\
            ['net_profit'].quantile(.90)],
}, index = ['Output', 'Labor', 'Capital', 'Equity', 'Net Profit'])

summary_public_listed              = pd.DataFrame({
 'mean': [df_list['output'].mean(),        \
    df_list['emp_comp'].mean(),        df_list['capital'].mean(),\
        df_list['equity'].mean(),        df_list['net_profit'].mean()],
 'st.d': [df_list['output'].std(),         \
    df_list['emp_comp'].std(),         df_list['capital'].std(),\
 df_list['equity'].std(),         df_list['net_profit'].std()],
 'p10' : [df_list['output'].quantile(.10),\
     df_list['emp_comp'].quantile(.10), df_list['capital'].\
        quantile(.10), df_list['equity'].quantile(.10), \
            df_list['net_profit'].quantile(.10)],
 'p25' : [df_list['output'].quantile(.25),\
     df_list['emp_comp'].quantile(.25), df_list['capital']\
        .quantile(.25), df_list['equity'].quantile(.25), df_list\
            ['net_profit'].quantile(.25)],
 'p50' : [df_list['output'].quantile(.50), \
    df_list['emp_comp'].quantile(.50), df_list['capital'].\
        quantile(.50), df_list['equity'].quantile(.50), df_list\
            ['net_profit'].quantile(.50)],
 'p75' : [df_list['output'].quantile(.75), \
    df_list['emp_comp'].quantile(.75), df_list['capital'].\
        quantile(.75), df_list['equity'].quantile(.75), df_privat\
            e['net_profit'].quantile(.75)],
 'p90' : [df_list['output'].quantile(.90),\
     df_list['emp_comp'].quantile(.90), df_list['capital'].\
        quantile(.90), df_list['equity'].quantile(.90), df_list\
            ['net_profit'].quantile(.90)],
}, index = ['Output', 'Labor', 'Capital', 'Equity', 'Net Profit'])



# Export to Latex
summary_public_nonlistedlat  = summary_public_nonlisted.to_latex(index=True)
title   = 'summary_nonlist_raw'
extension               = '.tex'
path    = 'N:/durable/file-export/IssamSheikh\
    /S1_empirical_analysis/tables/raw/' + title + extension

with open(path, 'w') as f:
    f.write(summary_public_nonlistedlat)

summary_public_listedlat   = summary_public_listed.to_latex(index=True)
title   = 'summary_list_raw'
extension               = '.tex'
```

```
path     = 'N:/durable/file-export/IssamSheikh\
    /S1_empirical_analysis/tables/raw/' + title + extension

with open(path, 'w') as f:
    f.write(summary_public_listedlat)
```

Tables for winsorized data

```
In [ ]: # Permanent Sample:
summary_private1             = pd.DataFrame({
 'mean': [df_private['output_win'].mean(), \
           df_private['emp_comp_win'].mean(),      \
              df_private['capital_win'].mean(),      \
  df_private['equity_win'].mean(),        \
     df_private['net_profit_win'].mean()],
 'st.d': [df_private['output_win'].std(),     \
            df_private['emp_comp_win'].std(),   \
      df_private['capital_win'].std(),    \
     df_private['equity_win'].std(),       \
  df_private['net_profit_win'].std()],
 'p10' : [df_private['output_win'].quantile(.10),\
     df_private['emp_comp_win'].quantile(.10),\
         df_private['capital_win'].quantile(.10), \
            df_private['equity_win'].quantile(.10), \
df_private['net_profit_win'].quantile(.10)],
 'p25' : [df_private['output_win'].quantile(.25),\
     df_private['emp_comp_win'].quantile(.25),\
         df_private['capital_win'].quantile(.25),\
             df_private['equity_win'].quantile(.25)\
, df_private['net_profit_win'].\
    quantile(.25)],
 'p50' : [df_private['output_win'].quantile(.50),\
     df_private['emp_comp_win'].quantile(.50), \
        df_private['capital_win'].quantile(.50),\
             df_private['equity_win'].quantile(.50),\
 df_private['net_profit_win'].quantile(.50)],
 'p75' : [df_private['output_win'].quantile(.75),\
     df_private['emp_comp_win'].quantile(.75), \
        df_private['capital_win'].quantile(.75),\
             df_private['equity_win'].quantile(.75),\
 df_private['net_profit_win'].\
    quantile(.75)],
 'p90' : [df_private['output_win'].quantile(.90)\
    , df_private['emp_comp_win'].quantile(.90), \
        df_private['capital_win'].quantile(.90),\
             df_private['equity_win'].quantile(.90)\
, df_private['net_profit_win'].\
    quantile(.90)],
 }, index = ['Output', 'Labor', 'Capital', 'Equity', 'Net Profit_win'])

summary_public1              = pd.DataFrame({
 'mean': [df_public['output_win'].mean(), \
           df_public['emp_comp_win'].mean(),      \
              df_public['capital_win'].mean(),      \
  df_public['equity_win'].mean(),        \
     df_public['net_profit_win'].mean()],
 'st.d': [df_public['output_win'].std(),     \
            df_public['emp_comp_win'].std(),   \
      df_public['capital_win'].std(),    \
     df_public['equity_win'].std(),       \
  df_public['net_profit_win'].std()],
 'p10' : [df_public['output_win'].quantile(.10),\
     df_public['emp_comp_win'].quantile(.10),\
         df_public['capital_win'].quantile(.10), \
            df_public['equity_win'].quantile(.10), \
df_public['net_profit_win'].quantile(.10)],
 'p25' : [df_public['output_win'].quantile(.25),\
     df_public['emp_comp_win'].quantile(.25),\
         df_public['capital_win'].quantile(.25),\
             df_public['equity_win'].quantile(.25)\
, df_public['net_profit_win'].\
    quantile(.25)],
 'p50' : [df_public['output_win'].quantile(.50),\
     df_public['emp_comp_win'].quantile(.50), \
        df_public['capital_win'].quantile(.50),\
             df_public['equity_win'].quantile(.50),\
 df_public['net_profit_win'].quantile(.50)],
 'p75' : [df_public['output_win'].quantile(.75),\
     df_public['emp_comp_win'].quantile(.75), \
        df_public['capital_win'].quantile(.75),\
             df_public['equity_win'].quantile(.75),\
 df_public['net_profit_win'].\
    quantile(.75)],
 'p90' : [df_public['output_win'].quantile(.90)\
    , df_public['emp_comp_win'].quantile(.90), \
        df_public['capital_win'].quantile(.90),\
             df_public['equity_win'].quantile(.90)\
, df_public['net_profit_win'].\
    quantile(.90)],
 }, index = ['Output', 'Labor', 'Capital', 'Equity', 'Net Profit_win'])



# Export to Latex
summary_private_latperm     = summary_private1.to_latex(index=True)
title   = 'summary_private_win'
extension               = '.tex'
path    = 'N:/durable/file-export/IssamSheikh\
    /S1_empirical_analysis/tables/win/' + title + extension

with open(path, 'w') as f:
    f.write(summary_private_latperm)

summary_public_latperm      = summary_public1.to_latex(index=True)
title   = 'summary_public_win'
extension               = '.tex'
path    = 'N:/durable/file-export/IssamSheikh\
    /S1_empirical_analysis/tables/win/' + title + extension
with open(path, 'w') as f:
    f.write(summary_public_latperm)
```

```
# permanent_sample: Creating Table 1: Summary Statistics Listed vs Nonlisted:

summary_public_nonlisted             = pd.DataFrame({
 'mean': [df_public['output_win'].mean(), \
           df_public['emp_comp_win'].mean(),      \
              df_public['capital_win'].mean(),      \
  df_public['equity_win'].mean(),       \
     df_public['net_profit_win'].mean()],
 'st.d': [df_public['output_win'].std(),     \
            df_public['emp_comp_win'].std(),   \
      df_public['capital_win'].std(),    \
     df_public['equity_win'].std(),       \
  df_public['net_profit_win'].std()],
 'p10' : [df_public['output_win'].quantile(.10),\
     df_public['emp_comp_win'].quantile(.10),\
         df_public['capital_win'].quantile(.10), \
            df_public['equity_win'].quantile(.10), \
df_public['net_profit_win'].quantile(.10)],
 'p25' : [df_public['output_win'].quantile(.25),\
     df_public['emp_comp_win'].quantile(.25),\
         df_public['capital_win'].quantile(.25),\
             df_public['equity_win'].quantile(.25)\
, df_public['net_profit_win'].\
    quantile(.25)],
 'p50' : [df_public['output_win'].quantile(.50),\
     df_public['emp_comp_win'].quantile(.50), \
        df_public['capital_win'].quantile(.50),\
             df_public['equity_win'].quantile(.50),\
 df_public['net_profit_win'].quantile(.50)],
 'p75' : [df_public['output_win'].quantile(.75),\
     df_public['emp_comp_win'].quantile(.75), \
        df_public['capital_win'].quantile(.75),\
             df_public['equity_win'].quantile(.75),\
 df_public['net_profit_win'].\
    quantile(.75)],
 'p90' : [df_public['output_win'].quantile(.90)\
    , df_public['emp_comp_win'].quantile(.90), \
        df_public['capital_win'].quantile(.90),\
             df_public['equity_win'].quantile(.90)\
, df_public['net_profit_win'].\
    quantile(.90)],
 }, index = ['Output', 'Labor', 'Capital', 'Equity', 'Net Profit_win'])


summary_public_listed              = pd.DataFrame({
 'mean': [df_list['output_win'].mean(), \
           df_list['emp_comp_win'].mean(),      \
              df_list['capital_win'].mean(),      \
  df_list['equity_win'].mean(),       \
     df_list['net_profit_win'].mean()],
 'st.d': [df_list['output_win'].std(),     \
            df_list['emp_comp_win'].std(),   \
      df_list['capital_win'].std(),    \
     df_list['equity_win'].std(),       \
  df_list['net_profit_win'].std()],
 'p10' : [df_list['output_win'].quantile(.10),\
     df_list['emp_comp_win'].quantile(.10),\
         df_list['capital_win'].quantile(.10), \
            df_list['equity_win'].quantile(.10), \
df_list['net_profit_win'].quantile(.10)],
 'p25' : [df_list['output_win'].quantile(.25),\
     df_list['emp_comp_win'].quantile(.25),\
         df_list['capital_win'].quantile(.25),\
             df_list['equity_win'].quantile(.25)\
, df_list['net_profit_win'].\
    quantile(.25)],
 'p50' : [df_list['output_win'].quantile(.50),\
     df_list['emp_comp_win'].quantile(.50), \
        df_list['capital_win'].quantile(.50),\
             df_list['equity_win'].quantile(.50),\
 df_list['net_profit_win'].quantile(.50)],
 'p75' : [df_list['output_win'].quantile(.75),\
     df_list['emp_comp_win'].quantile(.75), \
        df_list['capital_win'].quantile(.75),\
             df_list['equity_win'].quantile(.75),\
 df_list['net_profit_win'].\
    quantile(.75)],
 'p90' : [df_list['output_win'].quantile(.90)\
    , df_list['emp_comp_win'].quantile(.90), \
        df_list['capital_win'].quantile(.90),\
             df_list['equity_win'].quantile(.90)\
, df_list['net_profit_win'].\
    quantile(.90)],
 }, index = ['Output', 'Labor', 'Capital', 'Equity', 'Net Profit_win'])



# Export to Latex
summary_public_nonlistedlat  = summary_public_nonlisted.to_latex(index=True)
title   = 'summary_nonlist_win'
extension               = '.tex'
path    = 'N:/durable/file-export/IssamSheikh/
S1_empirical_analysis/tables/win/' + title + extension

with open(path, 'w') as f:
    f.write(summary_public_nonlistedlat)

summary_public_listedlat   = summary_public_listed.to_latex(index=True)
title   = 'summary_list_win'
extension               = '.tex'
path    = 'N:/durable/file-export/IssamShei\
    kh/S1_empirical_analysis/tables/win/' + title + extension

with open(path, 'w') as f:
    f.write(summary_public_listedlat)
```

Rate of Returns ROE

In [ ]:
```
# Table 2: Rate of Returns ROE
# split into private and public, with bar π / a' for each profile
```

```python
ROE_by_pri = df_private.groupby('orgnr')['ROE_win'].mean()
ROE_by_pub = df_public.groupby('orgnr')['ROE_win'].mean()

# Creating Table 2 as a Dataframe
T1_profile = pd.DataFrame({
'mean':[df_private['ROE_win'].mean(),        ROE_by_pri.mean(),    \
           df_public['ROE_win'].mean(),          ROE_by_pub.mean()],
'std': [df_private['ROE_win'].std(),         ROE_by_pri.std(),          \
      df_public['ROE_win'].std(),            ROE_by_pub.std()],
'p10': [df_private['ROE_win'].quantile(.10), ROE_by_pri.quantile(.10), \
       df_public['ROE_win'].quantile(.10),   ROE_by_pub.quantile(.10)],
'p25': [df_private['ROE_win'].quantile(.25), ROE_by_pri.quantile(.25),  \
      df_public['ROE_win'].quantile(.25),   ROE_by_pub.quantile(.25)],
'p50': [df_private['ROE_win'].quantile(.50), ROE_by_pri.quantile(.50), \
       df_public['ROE_win'].quantile(.50),   ROE_by_pub.quantile(.50)],
'p75': [df_private['ROE_win'].quantile(.75), ROE_by_pri.quantile(.75), \
       df_public['ROE_win'].quantile(.75),   ROE_by_pub.quantile(.75)],
'p90': [df_private['ROE_win'].quantile(.90), ROE_by_pri.quantile(.90), \
       df_public['ROE_win'].quantile(.90),   ROE_by_pub.quantile(.90)],
'p95': [df_private['ROE_win'].quantile(.95), ROE_by_pri.quantile(.95),  \
      df_public['ROE_win'].quantile(.95),   ROE_by_pub.quantile(.95)]
}, index=['Private: ARR', 'Private: bar ARR', 'Public: ARR', 'Public: bar ARR'])

# Export to Latex
t1_lat  = T1_profile.to_latex(index=True)
title   = 'rateofreturns'
extension              = '.tex'
path    = 'N:/durable/file-export/IssamSheikh\
    /S1_empirical_analysis/tables/win/' + title + extension

with open(path, 'w') as f:
    f.write(t1_lat)

# Creating Table 2: Average Rates of Return:

# Creating Table 2 as a Dataframe
ROE_by_nlist  = df_nonlist.groupby('orgnr')['ROE_win'].mean()
ROE_by_list = df_list.groupby('orgnr')['ROE_win'].mean()

T1_listedvsnonlisted = pd.DataFrame({
'mean':[df_nonlist['ROE_win'].mean(),        ROE_by_nlist.mean(),  \
             df_list['ROE_win'].mean(),          ROE_by_list.mean()],
'std': [df_nonlist['ROE_win'].std(),         ROE_by_nlist.std(),   \
             df_list['ROE_win'].std(),           ROE_by_list.std()],
'p10': [df_nonlist['ROE_win'].quantile(.10), ROE_by_nlist.quantile(.10),\
        df_list['ROE_win'].quantile(.10),   ROE_by_list.quantile(.10)],
'p25': [df_nonlist['ROE_win'].quantile(.25), ROE_by_nlist.quantile(.25), \
       df_list['ROE_win'].quantile(.25),   ROE_by_list.quantile(.25)],
'p50': [df_nonlist['ROE_win'].quantile(.50), ROE_by_nlist.quantile(.50), \
       df_list['ROE_win'].quantile(.50),   ROE_by_list.quantile(.50)],
'p75': [df_nonlist['ROE_win'].quantile(.75), ROE_by_nlist.quantile(.75),\
        df_list['ROE_win'].quantile(.75),   ROE_by_list.quantile(.75)],
'p90': [df_nonlist['ROE_win'].quantile(.90), ROE_by_nlist.quantile(.90), \
       df_list['ROE_win'].quantile(.90),   ROE_by_list.quantile(.90)],
'p95': [df_nonlist['ROE_win'].quantile(.95), ROE_by_nlist.quantile(.95), \
       df_list['ROE_win'].quantile(.95),   ROE_by_list.quantile(.95)]
}, index=['public_nonlisted: ARR', 'public_nonlisted: bar ARR', \
    'public_listed: ARR', 'public_listed: bar ARR'])



# Export to Latex
t1_lat1  = T1_listedvsnon\
    listed.to_latex(index=True)
title   = 'T1_listedvsnonlisted'
extension               = '.tex'
path    = 'N:/durable/file-export/\
    IssamSheikh/S1_empirical_analysis/tables/win/' + title + extension


with open(path, 'w') as f:
    f.write(t1_lat1)
```

Rate of Returns ROA

```
In [ ]: # Table 2: Rate of Returns ROA
        # split into private and public, with bar π / a' for each profile

        ROE_by_pri = df_private.groupby(\
            'orgnr')['ROA_win'].mean()
        ROE_by_pub = df_public.groupby('\
            orgnr')['ROA_win'].mean()

        # Creating Table 2 as a Dataframe
        T1_profile = pd.DataFrame({
        'mean':[df_private['ROA_win'].mean(), \
                   ROE_by_pri.mean(),           \
                     df_public['ROA_win'].mean(), \
                 ROE_by_pub.mean()],
        'std': [df_private['ROA_win'].std(),    \
                  ROE_by_pri.std(),            \

                     df_public['ROA_win'].std(),    \
               ROE_by_pub.std()],
        'p10': [df_private['ROA_win'].quantile(.10), \
            ROE_by_pri.quantile(.10),     \
                df_public['ROA_win'].quantile(.10),\
                   ROE_by_pub.quantile(.10)],
        'p25': [df_private['ROA_win'].quantile(.25),\
             ROE_by_pri.quantile(.25),     \
                df_public['ROA_win'].quantile(.25),\
                       ROE_by_pub.quantile(.25)],
        'p50': [df_private['ROA_win'].quantile(.50), \
            ROE_by_pri.quantile(.50),     \
                df_public['ROA_win'].quantile(.50),   \
                    ROE_by_pub.quantile(.50)],
        'p75': [df_private['ROA_win'].quantile(.75),\
             ROE_by_pri.quantile(.75),     \
                df_public['ROA_win'].quantile(.75),  \
                     ROE_by_pub.quantile(.75)],
        'p90': [df_private['ROA_win'].quantile(.90),\
     ROE_by_pri.quantile(.90),    \\

        df_public['ROA_win'].quantile(.90),   \
            ROE_by_pub.quantile(.90)],
'p95': [df_private['ROA_win'].quantile(.95), \
    ROE_by_pri.quantile(.95),    \
        df_public['ROA_win'].quantile(.95),   ROE_by_pub.quantile(.95)]
}, index=['Private: ARR', 'Private: bar ARR', 'Public: ARR', 'Public: bar ARR'])

# Export to Latex
t1_lat  = T1_profile.to_latex(index=True)
title   = 'rateofreturnsroa'
extension               = '.tex'
path    = 'N:/durable/file-export\
    /IssamSheikh/S1_empirical_analysis/tables/win/' + title + extension

with open(path, 'w') as f:
    f.write(t1_lat)

# Creating Table 2: Average Rates of Return:

# Creating Table 2 as a Dataframe
ROE_by_nlist  = df_nonlist.groupby('orgnr')['ROA_win'].mean()
ROE_by_list = df_list.groupby('orgnr')['ROA_win'].mean()

T1_listedvsnonlisted = pd.DataFrame({
'mean':[df_nonlist['ROA_win'].mean(),\
            ROE_by_nlist.mean(),           df_list['ROA_win'].mean(),\
          ROE_by_list.mean()],
'std': [df_nonlist['ROA_win'].std(),\
             ROE_by_nlist.std(),            df_list['ROA_win'].std(), \
          ROE_by_list.std()],
'p10': [df_nonlist['ROA_win'].quantile(.10)\
    , ROE_by_nlist.quantile(.10),    df_list['ROA_win'].quantile(.10), \
          ROE_by_list.quantile(.10)],
'p25': [df_nonlist['ROA_win'].quantile(.25)\
    , ROE_by_nlist.quantile(.25),    df_list['ROA_win'].quantile(.25),\
           ROE_by_list.quantile(.25)],
'p50': [df_nonlist['ROA_win'].quantile(.50),\
     ROE_by_nlist.quantile(.50),    df_list['ROA_win'].quantile(.50), \
          ROE_by_list.quantile(.50)],
'p75': [df_nonlist['ROA_win'].quantile(.75),\
     ROE_by_nlist.quantile(.75),    df_list['ROA_win'].quantile(.75),\
           ROE_by_list.quantile(.75)],
'p90': [df_nonlist['ROA_win'].quantile(.90),\
    ROE_by_nlist.quantile(.90),    df_list['ROA_win'].quantile(.90),\
           ROE_by_list.quantile(.90)],
'p95': [df_nonlist['ROA_win'].quantile(.95),\
     ROE_by_nlist.quantile(.95),    df_list['ROA_win'].quantile(.95),\
           ROE_by_list.quantile(.95)]
}, index=['public_nonlisted: ARR', 'public_nonlisted: bar ARR', \
    'public_listed: ARR', 'public_listed: bar ARR'])



# Export to Latex
```

```
t1_lat1  = T1_listedvsnonlisted.to_\
    latex(index=True)
title   = 'T1_listedvsnonlistedroa'
extension               = '.tex'
path    = 'N:/durable/file-export/\
    IssamSheikh/S1_empirical_analysis/tables/win/' + title + extension


with open(path, 'w') as f:
    f.write(t1_lat1)
```

Output Growth Rates

In [ ]:
```
# Table 4: Output Growth Rates
table4_outputgrowth = df_permanent.dropna(subset=['lg_output_growth'])
# checking the permanent sample first --
#this is useful when choosing the std. for
# the Gausssian distribution. table4_outputgrowth and permanent are equal

##table4_outputgrowth:
outputgrowth_distribution    = pd.DataFrame({
 's.d'  : table4_outputgrowth['lg_output_growth']\
    .std(),
 'p10'  : table4_outputgrowth['lg_output_growth'].\
    quantile(.10),
 'p25'  : table4_outputgrowth['lg_output_growth']\
    .quantile(.25),
 'p50'  : table4_outputgrowth['lg_output_growth'].\
    quantile(.50),
 'p75'  : table4_outputgrowth['lg_output_growth'].\
    quantile(.75),
 'p90'  : table4_outputgrowth['lg_output_growth']\
    .quantile(.90),
 'p995' : table4_outputgrowth['lg_output_growth']\
    .quantile(.995),
 'p999' : table4_outputgrowth['lg_output_growth']\
    .quantile(.999)}, index = ['All Companies'])

#print(outputgrowth_distribution.to_markdown())


# Export to Latex
outputgrowthlat1        = outputgrowth_distribution.\
    to_latex(index=True)
title   = 'outputgrowt'
extension               = '.tex'
path    = 'N:/durable/file-export/IssamSheikh\
    /S1_empirical_analysis/tables/win/' + title + extension
with open(path, 'w') as f:
    f.write(outputgrowthlat1)

# Gaussian:
std_dev      = table4_outputgrowth['lg_output_growth'].std()
sample       = table4_outputgrowth.shape[0]
mean         = 0
```

```
gaussian    = np.random.normal(mean, std_dev, sample)

# Distribution Table for Output Growth:
outputgrowth_profiles_distributionnan    = pd.DataFrame({
'St.d':   [table4_outputgrowth['lg_output_growth'].std(),  \
             table4_outputgrowth.loc[table4_outputgrowth['private'] == 1\
]['lg_output_growth'].std(),\
             table4_outputgrowth.loc[table4_outputgrowth['public'] == 1\
]['lg_output_growth'].std(),\
              gaussian.std()],
'P01' :   [table4_outputgrowth['lg_output_growth'].quantile(0.01), \
     table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['lg_output_growth'].quantile(0.01)\
        ,    table4_outputgrowth.loc[table4_outputgrowth['public'] == 1]\
            ['lg_output_growth'].quantile(0.01),    \
 np.percentile(gaussian, 0.1)],
'P05' :   [table4_outputgrowth['lg_output_growth'].quantile(0.05),\
      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['lg_output_growth'].quantile(0.05),\
            table4_outputgrowth.loc[table4_outputgrowth['public'] == 1]\
['lg_output_growth'].quantile(0.05),     np.percentile(gaussian, 0.5)],
'P25' :   [table4_outputgrowth['lg_output_growth'].quantile(.25), \
      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['lg_output_growth'].quantile(.25), \
            table4_outputgrowth.loc[table4_outputgrowth['public'] == 1]\
['lg_output_growth'].quantile(.25),      \
    np.percentile(gaussian, 25)],
'P75' :   [table4_outputgrowth['lg_output_growth'].quantile(.75), \\

      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['lg_output_growth'].quantile(.75), \\

            table4_outputgrowth.loc[table4_outputgrowth['public'] == 1]\
['lg_output_growth'].quantile(.75),     \
     np.percentile(gaussian, 75)],
'P90' :   [table4_outputgrowth['lg_output_growth'].quantile(.90), \
      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['lg_output_growth'].quantile(.90), \
            table4_outputgrowth.loc[table4_outputgrowth['public'] == 1]\
['lg_output_growth'].quantile(.90),      \
    np.percentile(gaussian, 90)],
'P99.5' : [table4_outputgrowth['lg_output_growth'].quantile(.995),\
      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['lg_output_growth'].quantile(.995), \
           table4_outputgrowth.loc[table4_outputgrowth['public'] == 1]\
            ['lg_output_growth'].quantile(.995),     \
np.percentile(gaussian, 99.5)],
'P99.9' : [table4_outputgrowth['lg_output_growth'].quantile(.999),\
    \
      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['lg_output_growth'].quantile(.999),\
            table4_outputgrowth.loc[table4_outputgrowth['public'] == 1]\
['lg_output_growth'].quantile(.999),     np.percentile(gaussian, 99.9)],

}, index = ['Full Sample', "Private Companies", \
    "Public Companies", "Gaussian"])
```

```
#print(outputgrowth_profiles_distributionnan.to_markdown())


# Export to Stata
outputgrowth_profiles_distributionnanlat      = \
    outputgrowth_profiles_distributionnan.to_latex(index=True)
title   = 'outputgrowthall'
extension                = '.tex'
path    = 'N:/durable/file-export/IssamSheikh/S1_\
    empirical_analysis/tables/win/' + title + extension
with open(path, 'w') as f:
    f.write(outputgrowth_profiles_distributionnanlat)


# Kurtosis:
#print(table4_outputgrowth\
# ['lg_output_growth'].skew())
#print(table4_outputgrowth\
# ['lg_output_growth'].kurtosis())
#print(table4_outputgrowth\
# [table4_outputgrowth['private']==1]['lg_output_growth'].kurtosis())
#print(table4_outputgrowth\
# [table4_outputgrowth['public_nonlisted']==1]['lg_output_growth'].kurtosis())
#print(table4_outputgrowth\
# [table4_outputgrowth['public_listed']==1]['lg_output_growth'].kurtosis())
#print(table4_outputgrowth\
# [table4_outputgrowth['public']==1]['lg_output_growth'].kurtosis())
print(df_permanent.shape[0] - table4_outputgrowth.shape[0]) # 222392
table4_outputgrowth.shape[0]
```

In [ ]:

```
table4_outputgrowth.loc[table4_outputgrowth['private']==1].shape[0]
table4_outputgrowth.loc[table4_outputgrowth['public']==1].shape[0]
```

Labor and Profit Deviations

In [ ]:

```
# Distribution Table for Output Growth:
labor_deviations_tablenan  = pd.DataFrame({
'Mean':   [table4_outputgrowth['labor_deviation'].mean(),\
              table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
['labor_deviation'].mean(),        \
 table4_outputgrowth.loc[table4_outputgrowth['public'] == 1]\
['labor_deviation'].mean()],
'P01' :   [table4_outputgrowth['labor_deviation'].quantile(0.01),\
      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
['labor_deviation'].quantile(0.01), table4_outputgrowth.loc[\
    table4_outputgrowth['public'] == 1]['labor_deviation'].q\
        uantile(0.01)],
'P10' :   [table4_outputgrowth['labor_deviation'].quantile(.10),\
       table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['labor_deviation'].quantile(.10),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['labor_deviation']\
                .quantile(.10)],
'P25' :   [table4_outputgrowth['labor_deviation'].quantile(.25),\
       table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['labor_deviation'].quantile(.25),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['labor_deviation'].\
                quantile(.25)],
'P50' :   [table4_outputgrowth['labor_deviation'].quantile(.50),\
       table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['labor_deviation'].quantile(.50),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['labor_deviation'].\
                quantile(.50)],
'P75' :   [table4_outputgrowth['labor_deviation'].quantile(.75), \
      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['labor_deviation'].quantile(.75),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['labor_deviation'].\
                quantile(.75)],
'P90' :   [table4_outputgrowth['labor_deviation'].quantile(.90),\
       table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['labor_deviation'].quantile(.90),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['labor_deviation'].\
                quantile(.90)],
'P99' :   [table4_outputgrowth['labor_deviation'].quantile(.99),
\   \
    table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
    ['labor_deviation'].quantile(.99),  table4_outputgrowth.loc\
        [table4_outputgrowth['public'] == 1]['labor_deviation'].quantile(.99)]\
            }, index=['Permanent Sample', 'Private Companies', \
                'Public Companies'])

# Export to Latex
labor_deviations_tablelat       = labor_deviations_t\
           ablenan.to_latex(index=True)
title  = 'labor_deviations_tablenan'
extension       = '.tex'
path   = 'N:/durable/file-export/IssamSheikh/S1_\
    empirical_analysis/tables/win/' + title + extension
with open(path, 'w') as f:
    f.write(labor_deviations_tablelat)

profit_deviations_tablenan  = pd.DataFrame({
'Mean':   [table4_outputgrowth['profit_deviation'].mean(),\
              table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
['profit_deviation'].mean(),        \
 table4_outputgrowth.loc[table4_outputgrowth['public'] == 1]\
['profit_deviation'].mean()],
'P01' :   [table4_outputgrowth['profit_deviation'].quantile(0.01),\
      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
['profit_deviation'].quantile(0.01), table4_outputgrowth.loc[\
    table4_outputgrowth['public'] == 1]['profit_deviation'].q\
        uantile(0.01)],
'P10' :   [table4_outputgrowth['profit_deviation'].quantile(.10),\
       table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['profit_deviation'].quantile(.10),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['profit_deviation']\
                .quantile(.10)],
'P25' :   [table4_outputgrowth['profit_deviation'].quantile(.25),\
       table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['profit_deviation'].quantile(.25),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['profit_deviation'].\
                quantile(.25)],
'P50' :   [table4_outputgrowth['profit_deviation'].quantile(.50),\
       table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['profit_deviation'].quantile(.50),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['profit_deviation'].\
                quantile(.50)],
'P75' :   [table4_outputgrowth['profit_deviation'].quantile(.75), \
      table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['profit_deviation'].quantile(.75),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['profit_deviation'].\
                quantile(.75)],
'P90' :   [table4_outputgrowth['profit_deviation'].quantile(.90),\
       table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
        ['profit_deviation'].quantile(.90),  table4_outputgrowth.loc\
            [table4_outputgrowth['public'] == 1]['profit_deviation'].\
                quantile(.90)],
'P99' :   [table4_outputgrowth['profit_deviation'].quantile(.99),
\   \
    table4_outputgrowth.loc[table4_outputgrowth['private'] == 1]\
    ['profit_deviation'].quantile(.99),  table4_outputgrowth.loc\
        [table4_outputgrowth['public'] == 1]['profit_deviation'].quantile(.99)]\
            }, index=['Permanent Sample', 'Private Companies', \
                'Public Companies'])
# Export to Latex
profit_deviations_tablelat       = profit_deviations_\
    tablenan.to_latex(index=True)
title  = 'profit_deviations_tablenan'
extension        = '.tex'
path   = 'N:/durable/file-export/IssamSheikh/S1_\
    empirical_analysis/tables/win/' + title + extension
with open(path, 'w') as f:
    f.write(profit_deviations_tablelat)
```

In [ ]:
```
x = table4_outputgrowth[table4_outputgrowth['private'] == 1]
y = table4_outputgrowth[table4_outputgrowth['public'] == 1]
print(x['labor_deviation'].corr(x['profit_deviation']))
print(y['labor_deviation'].corr(y['profit_deviation']))
```

In [ ]:
```
df_private['labor_share'] = df_private['emp_comp_win'] / df_private['output_win']
```

In [ ]:
```
df_public['labor_share'] = df_public['emp_comp_win'] / df_public['output_win']
```

In [ ]:
```
df_private['labor_share'].mean()
```

Convert to .DTA

In [ ]:
```
# Keep only variables used for the regressions:
# ROA, profit_share, capital_win, pi_hat, emp_comp, cap

df_reg =\
    table4_outputgrowth\
      [['orgnr', 'year', 'ROA_win', \
         'profit_share_win', 'lg_capital_win', \
            'capital_win', \
   'emp_comp_win', 'lg_emp_comp_win', \
      'net_profit_win', 'pi_hat_win', \
         'lg_output_growth', 'output_win', \
   'private', 'public', 'public_listed', \
      'public_nonlisted']]
```

In [ ]:
```
df_reg.to_stata('N:/durable/BIStudents/\
IssamSheikh/codes/Duplicates/5_analyses -\
 Copy/5_empirical_analysis/comovement.dta')
```

Plots

In [ ]:
```
# Example of A Firm
# Labor and Output, Labor Share and Profit Share
p1 = df_permanent[df_permanent['orgnr'] == 'F012378621']

fig, axs = plt.subplots(1, 3)
axs[0].plot(p1['year'], p1['norm_emp_comp'],         \
    color='darkred',  linestyle='--',      label = \
'log l')
axs[0].plot(p1['year'], p1['norm_output_growth'],   \
     color='midnightblue',  label = 'log y')
axs[0].legend()
axs[0].set_title('Output and Labor', loc='center')

axs[1].plot(p1['year'], p1['l_output_win'],        \
     color='midnightblue',    label = 'l/y')
axs[1].axhline(y=p1['l_output_win'].mean(),        \
     color='darkred',      linestyle='--',    \
        label = 'l/y')
axs[1].legend(loc='upper right')\

axs[1].set_title('Labor share', loc='center')

axs[2].plot(p1['year'], p1['profit_share_win'],     \
        color='midnightblue',    label = 'π / a')
axs[2].axhline(y=p1['profit_share_win'].mean(),    \
         color='darkred',      linestyle='--',   \
             label = 'π / a')
axs[2].legend(loc='lower right')
axs[2].set_title('Profit share', loc='center')

plt.tight_layout()

save_path     = 'N:/durable/file-export/\
    IssamSheikh/S1_empirical_analysis/figures/ex1.jpg'
plt.savefig(save_path, format = 'jpeg')
```

In [ ]:
```
#  Rates of Return and Equity

# Create bins for the total assets
num_bins = 13
```

```
bins      = np.linspace(min(df_permanent.loc\
    [df_permanent['private']==1]['total_assets_win'\
        ]), max(df_permanent.loc[df_permanent['private']\
            ==1]['total_assets_win']), num_bins+1)
x_bins    = np.digitize(df_permanent.loc[df_permanent\
    ['private']==1]['total_assets_win'], bins)

# Calculate the mean y value for each bin
y_means = [np.mean(df_permanent.loc[df_permanent\
    ['private']==1]['ROA_win'][x_bins == i]) \
        for i in range(1, num_bins+1)]

# Scatter plot with binned variables
plt.plot(bins[:-1], y_means, marker='o', \
    linestyle='--', color='darkblue')
plt.xlabel('Total assets')
plt.ylabel('ROA')
plt.show()

save_path     = 'N:/durable/file-export/\
    IssamSheikh/S1_empirical_analysis/figures/\
        roa_private.jpg'
plt.savefig(save_path, format = 'jpeg')
```

```
In [ ]: # Create bins for the total assets
num_bins = 13
bins1     = np.linspace(min(df_permanent.\
    loc[df_permanent['public']==1]['total_assets_win']\
        ), max(df_permanent.loc[df_permanent['public']\
            ==1]['total_assets_win']), num_bins+1)
x_bins1   = np.digitize(df_permanent.loc[df_permanent[\
    'public']==1]['total_assets_win'], bins)

# Calculate the mean y value for each bin
y_means1 = [np.mean(df_permanent.loc[df_permanent\
    'public']==1]['ROA_win'][x_bins1 == i]) for i\
         in range(1, num_bins+1)]

# Scatter plot with binned variables
plt.plot(bins1[:-1], y_means, marker='o', \
    linestyle='--', color='darkred')
plt.xlabel('Total assets')
plt.ylabel('ROA')
plt.show()
save_path     = 'N:/durable/file-export/\
    IssamSheikh/S1_empirical_analysis/figures/\
        roa_public.jpg'
plt.savefig(save_path, format = 'jpeg')
```

```
In [ ]: # Create bins for the total assets  - PRIVATE
num_bins = 13
bins      = np.linspace(min(df_permanent.\
    loc[df_permanent['private']==1]['total_assets_win'])\
        , max(df_permanent.loc[df_permanent['private']==1]\
            ['total_assets_win']), num_bins+1)
```

```
x_bins    = np.digitize(df_permanent.loc[df_permanent\
    ['private']==1]['total_assets_win'], bins)
# Calculate the mean y value for each bin
y_means = [np.mean(df_permanent.loc[df_permanent['private']\
    ==1]['ROA_win'][x_bins == i]) for i in range(1, num_bins+1)]

# Create bins for the total assets  - PUBLIC
bins_public     = np.linspace(min(df_permanent.loc\
    [df_permanent['public']==1]['total_assets_win']),\
         max(df_permanent.loc[df_permanent['public']==1]\
            ['total_assets_win']), num_bins+1)
x_bins_public   = np.digitize(df_permanent.loc[df_permanent\
    ['public']==1]['total_assets_win'], bins_public)
# Calculate the mean y value for each bin
y_means_public  = [np.mean(df_permanent.loc[df_permanent\
    ['public']==1]['ROA_win'][x_bins_public == i]) for i in\
         range(1, num_bins+1)]

# Plot

fig,  axs = plt.subplots(1, 2)
axs[0].plot(bins[:-1], y_means, marker='o', linestyle='--',\
    color='darkblue')
axs[0].set_xlabel('Total Assets')
axs[0].set_ylabel('ROA')
axs[0].set_title('Private', loc='center')
axs[1].plot(bins_public[:-1], y_means_public, \
    marker='o', linestyle='--', color='darkred')
axs[1].set_xlabel('Total Assets')
#axs[1].set_ylabel('Total Assets')
axs[1].set_title('Public', loc='center')


plt.tight_layout()
save_path     = 'N:/durable/file-export/IssamSheikh\
    /S1_empirical_analysis/figures/roapripub.jpg'
plt.savefig(save_path, format = 'jpeg')
```

In [ ]:

```
pip install nbconvert
```

In [ ]:

```
# Import needed libraries
import pandas as pd
import numpy as np
import pyarrow.parquet as pq
pd.set_option('mode.use_inf_as_na', True)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline
from tabulate import tabulate
from scipy.stats.mstats import winsorize
import pyarrow as pa
import re

# Display 6 columns for viewing pupuses
pd.set_option('display.max_columns', 10)
# Reduce decimals points to 2
pd.options.display.float_format = '{:,.2f}'.format
```

In [ ]:

```
# Read the data
df_full      = pd.read_parquet('N:/durable/BIStudents/\
    IssamSheikh/data/clean/full_sample.parquet')
df_private   = pd.read_parquet('N:/durable/BIStudents/\
    IssamSheikh/data/clean/final/permanent_sample/df_priv\
ate.parquet')
df_nonlist   = pd.read_parquet('N:/durable/BIStudents/\
    IssamSheikh/data/clean/final/permanent_sample/df_nonli\
        st.parquet')
df_list      = pd.read_parquet('N:/durable/BIStudents/\
    IssamSheikh/data/clean/final/permanent_sample/df_list.\
        parquet')
df_public    = pd.read_parquet('N:/durable/BIStudents\
    /IssamSheikh/data/clean/final/permanent_sample/df_publ\
        ic.parquet')
```

Winsorising

Level variables

Returns

Full sample

In [ ]:

```
## ROE, ROA
# Winsorize ROE
df_full['ROE_win'] = winsorize(df_full['ROE'], (0.01, 0.01))
#df_full['ROE'].hist()
```

```
# Winsorize ROA
df_full['ROA_win'] = winsorize(df_full['ROA'], (0.01, 0.01))
#df_full['ROA_win'].hist()
```

Private firms

```
In [ ]: ## ROE, ROA
# Winsorize ROE
df_private['ROE_win'] = winsorize(df_private['ROE'], (0.01, 0.01))
#df_private['ROE_win'].hist()

# Winsorize ROA
df_private['ROA_win'] = winsorize(df_private['ROA'], (0.01, 0.01))
#df_private['ROA'].hist()
```

Public firms

```
In [ ]: ## ROE, ROA
# Winsorize ROE
df_public['ROE_win'] = winsorize(df_public['ROE'], (0.01, 0.01))
#df_public['ROE_win'].hist()

# Winsorize ROA
df_public['ROA_win'] = winsorize(df_public['ROA'], (0.01, 0.01))
#df_public['ROA_win'].hist()
```

Listed firms

```
In [ ]: ## ROE, ROA
# Winsorize ROE
df_list['ROE_win'] = winsorize(df_list['ROE'], (0.01, 0.01))
#df_list['ROE'].hist()

# Winsorize ROA
df_list['ROA_win'] = winsorize(df_list['ROA'], (0.01, 0.01))
#df_list['ROA_win'].hist()
```

Nonlisted firms

```
In [ ]: ## ROE, ROA
# Winsorize ROE
df_nonlist['ROE_win'] = winsorize(df_nonlist['ROE'], (0.01, 0.01))
#df_nonlist['ROE'].hist()

# Winsorize ROA
df_nonlist['ROA_win'] = winsorize(df_nonlist['ROA'], (0.01, 0.01))
#df_nonlist['ROA_win'].hist()
```

Size

Full sample

In [ ]:

```python
## total assets, operating income, output
# Winsorize total assets
df_full['total_assets_win'] = winsorize(df_full['total_assets'], (0.01, 0.01))
df_full['lg_total_assets'] = np.log(df_full['total_assets'])
#df_full['lg_total_assets'].hist()

# Winsorize operating income
df_full['op_inc_win'] = winsorize(df_full['op_inc'], (0.01, 0.01))
df_full['lg_op_inc'] = np.log(df_full['op_inc'])
#df_full['lg_op_inc'].hist()

# Winsorize operating revenue
df_full['rev_win'] = winsorize(df_full['rev'], (0.01, 0.01))
df_full['lg_rev'] = np.log(df_full['rev'])
#df_full['lg_rev'].hist()

# Winsorize output
df_full['output_win'] = winsorize(df_full['output'], (0.01, 0.01))
df_full['lg_output'] = np.log(df_full['output'])
#df_full['lg_output'].hist()

# Winsorize capital
df_full['capital_win'] = winsorize(df_full['capital'], (0.01, 0.01))
df_full['lg_capital'] = np.log(df_full['capital'])
#df_full['lg_capital'].hist()
```

Private firms

In [ ]:

```python
## total assets, operating income, output
# Winsorize total assets
df_private['total_assets_win'] = winsorize(df_private['total_assets'], (0.01, 0.01)
df_private['lg_total_assets'] = np.log(df_private['total_assets'])
#df_private['lg_total_assets_win'].hist()

# Winsorize operating income
df_private['op_inc_win'] = winsorize(df_private['op_inc'], (0.01, 0.01))
df_private['lg_op_inc'] = np.log(df_private['op_inc'])
#df_private['op_inc'].hist()

# Winsorize output
df_private['output_win'] = winsorize(df_private['output'], (0.01, 0.01))
df_private['lg_output'] = np.log(df_private['output'])
#df_private['output'].hist()
```

Public firms

In [ ]:

```python
## total assets, operating income, output
# Winsorize total assets
df_public['total_assets_win'] = winsorize(df_public['total_assets'], (0.01, 0.01))
df_public['lg_total_assets'] = np.log(df_public['total_assets'])
#df_public['lg_total_assets_win'].hist()

# Winsorize operating income
df_public['op_inc_win'] = winsorize(df_public['op_inc'], (0.01, 0.01))
```

```
df_public['lg_op_inc'] = np.log(df_public['op_inc'])
#df_public['op_inc'].hist()

# Winsorize output
df_public['output_win'] = winsorize(df_public['output'], (0.01, 0.01))
df_public['lg_output'] = np.log(df_public['output'])
#df_public['output'].hist()
```

Listed firms

```
In [ ]: ## total assets, operating income, output
        # Winsorize total assets
        df_list['total_assets_win'] = winsorize(df_list['total_assets'], (0.01, 0.01))
        df_list['lg_total_assets'] = np.log(df_list['total_assets'])
        #df_list['lg_total_assets_win'].hist()

        # Winsorize operating income
        df_list['op_inc_win'] = winsorize(df_list['op_inc'], (0.01, 0.01))
        df_list['lg_op_inc'] = np.log(df_list['op_inc'])
        #df_list['op_inc'].hist()

        # Winsorize output
        df_list['output_win'] = winsorize(df_list['output'], (0.01, 0.01))
        df_list['lg_output'] = np.log(df_list['output'])
        #df_list['output'].hist()
```

Non-listed firms

```
In [ ]: ## total assets, operating income, output
        # Winsorize total assets
        df_nonlist['total_assets_win'] = winsorize(df_nonlist['total_assets'], (0.01, 0.01)
        df_nonlist['lg_total_assets'] = np.log(df_nonlist['total_assets'])
        #df_nonlist['lg_total_assets_win'].hist()

        # Winsorize operating income
        df_nonlist['op_inc_win'] = winsorize(df_nonlist['op_inc'], (0.01, 0.01))
        df_nonlist['lg_op_inc'] = np.log(df_nonlist['op_inc'])
        #df_nonlist['op_inc'].hist()

        # Winsorize output
        df_nonlist['output_win'] = winsorize(df_nonlist['output'], (0.01, 0.01))
        df_nonlist['lg_output'] = np.log(df_nonlist['output'])
        #df_nonlist['output'].hist()
```

Ratio variable

Productivity

Full sample

```
In [ ]: ## output_l, output_k, output_a
        # Winsorize output_l
```

```
df_full['output_l_win'] = winsorize(df_full['output_l'], (0.01, 0.01))
df_full['lg_output_l'] = np.log(df_full['output_l'])
#df_full['output_l'].hist()

# Winsorize output_k
df_full['output_k_win'] = winsorize(df_full['output_k'], (0.01, 0.01))
df_full['lg_output_k'] = np.log(df_full['output_k'])
#df_full['lg_output_k'].hist()

# Winsorize output_a
df_full['output_a_win'] = winsorize(df_full['output_a'], (0.01, 0.01))
df_full['lg_output_a'] = np.log(df_full['output_a'])
#df_full['lg_output_a'].hist()
```

Private firms

In [ ]:
```
## output_l, output_k, output_a
# Winsorize output_l
df_private['output_l_win'] = winsorize(df_private['output_l'], (0.01, 0.01))
#df_private['output_l_win'].hist()

# Winsorize output_k
df_private['output_k_win'] = winsorize(df_private['output_k'], (0.01, 0.01))
#df_private['output_k_win'].hist()

# Winsorize output_a
df_private['output_a_win'] = winsorize(df_private['output_a'], (0.01, 0.01))
#df_private['output_a_win'].hist()
```

Public firms

In [ ]:
```
## output_l, output_k, output_a
# Winsorize output_l
df_public['output_l_win'] = winsorize(df_public['output_l'], (0.01, 0.01))
#df_public['output_l_win'].hist()

# Winsorize output_k
df_public['output_k_win'] = winsorize(df_public['output_k'], (0.01, 0.01))
#df_public['output_k_win'].hist()

# Winsorize output_a
df_public['output_a_win'] = winsorize(df_public['output_a'], (0.01, 0.01))
#df_public['output_a_win'].hist()
```

Listed firms

In [ ]:
```
## output_l, output_k, output_a
# Winsorize output_l
df_list['output_l_win'] = winsorize(df_list['output_l'], (0.01, 0.01))
#df_list['output_l_win'].hist()

# Winsorize output_k
df_list['output_k_win'] = winsorize(df_list['output_k'], (0.01, 0.01))
#df_list['output_k_win'].hist()
```

```
# Winsorize output_a
df_list['output_a_win'] = winsorize(df_list['output_a'], (0.01, 0.01))
#df_list['output_a_win'].hist()
```

Non-listed firms

```
In [ ]: ## output_l, output_k, output_a
# Winsorize output_l
df_nonlist['output_l_win'] = winsorize(df_nonlist['output_l'], (0.01, 0.01))
#df_nonlist['output_l_win'].hist()

# Winsorize output_k
df_nonlist['output_k_win'] = winsorize(df_nonlist['output_k'], (0.01, 0.01))
#df_nonlist['output_k_win'].hist()

# Winsorize output_a
df_nonlist['output_a_win'] = winsorize(df_nonlist['output_a'], (0.01, 0.01))
#df_nonlist['output_a_win'].hist()
```

Others

Full sample

```
In [ ]: ## Working capital, Operating leverage, Financial leverage, Liquidity ratio
# Winsorize working_capital
df_full['working_capital_win'] = winsorize(df_full\
    ['working_capital'], (0.01, 0.01))
#df_full['working_capital'].hist()

# Winsorize operating_leverage
df_full['operating_leverage_win'] = winsorize(df_full\
    ['operating_leverage'], (0.01, 0.01))
#df_full['operating_leverage_win'].hist()

# Winsorize financial_leverage
df_full['financial_leverage_win'] = winsorize(df_full[\
    'financial_leverage'], (0.01, 0.01))
#df_full['financial_leverage_win'].hist()

# Winsorize liquidity_ratio
df_full['liquidity_ratio_win'] = winsorize(df_full[\
    'liquidity_ratio'], (0.01, 0.01))
#df_full['liquidity_ratio_win'].hist()
```

Private firms

```
In [ ]: ## Working capital, Operating leverage, Financial leverage, Liquidity ratio
# Winsorize working_capital
df_private['working_capital_win'] = winsorize(\
    df_private['working_capital'], (0.01, 0.01))
#df_private['working_capital_win'].hist()
```

```
# Winsorize operating_leverage
df_private['operating_leverage_win'] = winsorize(\
    df_private['operating_leverage'], (0.01, 0.01))
#df_private['operating_leverage'].hist()

# Winsorize financial_leverage
df_private['financial_leverage_win'] = winsorize(\
    df_private['financial_leverage'], (0.01, 0.01))
#df_private['financial_leverage_win'].hist()

# Winsorize liquidity_ratio
df_private['liquidity_ratio_win'] = winsorize(\
    df_private['liquidity_ratio'], (0.01, 0.01))
#df_private['liquidity_ratio_win'].hist()
```

Public firms

```
In [ ]: ## Working capital, Operating leverage, Financial leverage, Liquidity ratio
        # Winsorize working_capital
        df_public['working_capital_win'] = winsorize(\
            df_public['working_capital'], (0.01, 0.01))
        #df_public['working_capital'].hist()

        # Winsorize operating_leverage
        df_public['operating_leverage_win'] = winsorize(\
            df_public['operating_leverage'], (0.01, 0.01))
        #df_public['operating_leverage'].hist()

        # Winsorize financial_leverage
        df_public['financial_leverage_win'] = winsorize(\
            df_public['financial_leverage'], (0.01, 0.01))
        #df_public['financial_leverage'].hist()

        # Winsorize liquidity_ratio
        df_public['liquidity_ratio_win'] = winsorize(\
            df_public['liquidity_ratio'], (0.01, 0.01))
        #df_public['liquidity_ratio_win'].hist()
```

Listed firms

```
In [ ]: ## Working capital, Operating leverage, Financial leverage, Liquidity ratio
        # Winsorize working_capital
        df_list['working_capital_win'] = winsorize(\
            df_list['working_capital'], (0.01, 0.01))
        #df_list['working_capital'].hist()

        # Winsorize operating_leverage
        df_list['operating_leverage_win'] = winsorize(d\
            f_list['operating_leverage'], (0.01, 0.01))
        #df_list['operating_leverage'].hist()

        # Winsorize financial_leverage
        df_list['financial_leverage_win'] = winsorize(d\
            f_list['financial_leverage'], (0.01, 0.01))
        #df_list['financial_leverage'].hist()
```

```
# Winsorize liquidity_ratio
df_list['liquidity_ratio_win'] = winsorize(\
    df_list['liquidity_ratio'], (0.01, 0.01))
#df_list['liquidity_ratio'].hist()
```

Non_listed firms

In [ ]:
```
## Working capital, Operating leverage, Financial leverage, Liquidity ratio
# Winsorize working_capital
df_nonlist['working_capital_win'] = winsorize(\
    df_nonlist['working_capital'], (0.01, 0.01))
#df_nonlist['working_capital'].hist()

# Winsorize operating_leverage
df_nonlist['operating_leverage_win'] = winsorize(\
    df_nonlist['operating_leverage'], (0.01, 0.01))
#df_nonlist['operating_leverage_win'].hist()

# Winsorize financial_leverage
df_nonlist['financial_leverage_win'] = winsorize(\
    f_nonlist['financial_leverage'], (0.01, 0.01))
#df_nonlist['financial_leverage'].hist()

# Winsorize liquidity_ratio
df_nonlist['liquidity_ratio_win'] = winsorize(\
    df_nonlist['liquidity_ratio'], (0.01, 0.01))
#df_nonlist['liquidity_ratio'].hist()
```

Regression df

In [ ]:
```
tst = df_full[df_full['total_assets'] > 3000000000]
```

In [ ]:
```
# Create bins for the x variable
num_bins = 20
bins = np.linspace(min(df_full['total_assets_win']), \
    max(df_full['total_assets_win']), num_bins+1)
x_bins = np.digitize(df_full['total_assets_win'], bins)

# Calculate the mean y value for each bin
y_means = [np.mean(df_full['ROA'][x_bins == i]) for i in range(1, num_bins+1)]

# Scatter plot with binned variables
plt.scatter(df_full['total_assets_win'], df_full['ROA'], \
    alpha=0.5, label='Data')
plt.scatter(bins[:-1], y_means, marker='o', \
    label='Binned Means')
plt.xlabel('X')
plt.ylabel('Y')
#plt.xlim(0.5,)
plt.title('Scatter Plot with Binned Variable')
plt.legend()
plt.show()
```

In [ ]:
```
# Create bins for the x variable
num_bins = 20
bins = np.linspace(min(df_full['total_assets_win']),\
     max(df_full['total_assets_win']), num_bins+1)
x_bins = np.digitize(df_full['total_assets_win'], bins)

# Calculate the mean y value for each bin
y_means = [np.mean(df_full['ROA_win'][x_bins == i])\
     for i in range(1, num_bins+1)]

# Scatter plot with binned variables
#plt.scatter(df_full['total_assets_win'], \
# df_full['ROA_win'], alpha=0.5, label='Data')
plt.scatter(bins[:-1], y_means, marker='o',\
     label='Binned Means')
plt.xlabel('X')
plt.ylabel('Y')
#plt.xlim(0.5,)
plt.title('Scatter Plot with Binned Variable')
plt.legend()
plt.show()
```

In [ ]:
```
df1 = df_full[['orgnr', 'year', 'age', 'industry', 'type',\
'ROE_win', 'ROA_win', 'total_assets_win', 'op_inc_win',\
'lg_total_assets', 'lg_op_inc', 'lg_output', 'lg_capital', \
'mrkt_share', 'operating_leverage_win', 'working_capital_win', \
    'financial_leverage_win', 'liquidity_ratio_win',\
'lg_output_a', 'lg_output_k', 'lg_output_l', 'output_a_win', \
    'output_k_win', 'output_l_win', \
'private', 'public','public_nonlisted']]

df2 = df_private[['orgnr', 'year', 'age', 'industry', 'type',\
'ROE_win', 'ROA_win','total_assets_win',  'op_inc_win', \
'lg_total_assets', 'lg_op_inc', 'lg_output',\
'mrkt_share', 'operating_leverage_win', 'working_capital_win', \
    'financial_leverage_win', 'liquidity_ratio_win',\
'output_a_win', 'output_k_win', 'output_l_win',\
'private', 'public','public_nonlisted']]

df3 = df_public[['orgnr', 'year', 'age', 'industry', 'type',\
'ROE_win', 'ROA_win', 'total_assets_win', 'op_inc_win', \
'lg_total_assets', 'lg_op_inc', 'lg_output',\
'mrkt_share', 'operating_leverage_win', 'working_capital_win',\
     'financial_leverage_win', 'liquidity_ratio_win',\
'output_a_win', 'output_k_win', 'output_l_win',\
'private', 'public','public_nonlisted']]

df4 = df_list[['orgnr', 'year', 'age', 'industry', 'type',\
'ROE_win', 'ROA_win', 'total_assets_win', 'op_inc_win', \
'lg_total_assets', 'lg_op_inc', 'lg_output',\
'mrkt_share', 'operating_leverage_win', 'working_capital_win',\
     'financial_leverage_win', 'liquidity_ratio_win',\
'output_a_win', 'output_k_win', 'output_l_win',\
'private', 'public','public_nonlisted']]

df5 = df_nonlist[['orgnr', 'year', 'age', 'industry', 'type',\
'ROE_win', 'ROA_win', 'total_assets_win', 'op_inc_win', \
'lg_total_assets', 'lg_op_inc', 'lg_output',\
'mrkt_share', 'operating_leverage_win', 'working_capital_win',\
     'financial_leverage_win', 'liquidity_ratio_win',\
'output_a_win', 'output_k_win', 'output_l_win',\
'private', 'public','public_nonlisted']]
```

In [ ]:
```
# Save pandas dataframe as parquet file
pq.write_table(pa.Table.from_pandas(df1),\
     'N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_full.parquet')
pq.write_table(pa.Table.from_pandas(df2), '\
    N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_private.parquet')
pq.write_table(pa.Table.from_pandas(df3), '\
    N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_public.parquet')
pq.write_table(pa.Table.from_pandas(df4), \
    'N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_listed.parquet')
pq.write_table(pa.Table.from_pandas(df5), \
    'N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_nonlisted.parquet')
```

Summary Statistics

Full Sample

In [ ]:
```
#Summary Statistic
summary_all = pd.DataFrame({    'mean': [df1['ROA_win'].mean(),\
        df1['total_assets_win'].mean(),\
        df1['op_inc_win'].mean(),\
          df1['mrkt_share'].mean(), \
     df1['output_l_win'].mean(),\
df1['output_k_win'].mean(),\
            df1['output_a_win'].mean(), \
       df1['operating_leverage_win'].mean(),\
df1['working_capital_win'].mean(),\
            df1['liquidity_ratio_win'].mean(),\
        df1['financial_leverage_win'].mean()], \

'p10' : [df1['ROA_win'].quantile(.10), \
    df1['total_assets_win'].quantile(.10)\
        , df1['op_inc_win'].quantile(.10),\
             df1['mrkt_share'].quantile(.10),\
   df1['output_l_win'].quantile(.10)\
    , df1['output_k_win'].quantile(.10),\
         df1['output_a_win'].quantile(.10),\
             df1['operating_leverage_win'].quantile(.10), \
df1['working_capital_win'].quantile(.10), \
    df1['liquidity_ratio_win'].quantile(.10)\
        , df1['financial_leverage_win'].\
            quantile(.10)],
'p25' : [df1['ROA_win'].quantile(.25), \
    df1['total_assets_win'].quantile(.25),\
         df1['op_inc_win'].quantile(.25), \
              df1['mrkt_share'].quantile(.25)\
,   df1['output_l_win']\
    .quantile(.25), df1['output_k_win']\
        .quantile(.25), df1['output_a_win']\
            .quantile(.25), \
df1['operating_leverage_win']\
    .quantile(.25), \
        df1['working_capital_win']\
            .quantile(.25), df1['liquidity_ratio_win'].\
quantile(.25), df1['financial_leverage_win']\
    .quantile(.25)],
'p50' : [df1['ROA_win'].quantile(.50),\
     df1['total_assets_win'].quantile(.50),\
         df1['op_inc_win'].quantile(.50),  \
             df1['mrkt_share'].quantile(.50),\
   df1['output_l_win'].quantile(.50),\
     df1['output_k_win'].quantile(.50),\
         df1['output_a_win'].quantile(.50),\
        df1['operating_leverage_win']\
         .quantile(.50), df1['working_capital_win'\
            ].quantile(.50), df1['liquidity_\
ratio_win'].quantile(.50),\
     df1['financial_leverage_win'\
        ].quantile(.50)],
'p75' : [df1['ROA_win'].quantile(.75),\
     df1['total_assets_win'].quantile(.75)\
        , df1['op_inc_win'].quantile(.75), \
              df1['mrkt_share'].quantile(.75),\
   df1['output_l_win'].quantile(.75)\
    , df1['output_k_win'].quantile(.75),\
         df1['output_a_win'].quantile(.75), \
            df1['operating_leverage_win'].quantile(.75),\
 df1['working_capital_win'].quantile(.75),\
     df1['liquidity_ratio_win'].quantile(.75),\
         df1['financial_leverage_win'].quantile(.75)],
'p90' : [df1['ROA_win'].quantile(.90), df1['total_assets_win'].\
    quantile(.90), df1['op_inc_win'].quantile(.90), \
          df1['mrkt_share'].quantile(.90),  \
             df1['output_l_win'].quantile(.90), \
df1['output_k_win'].quantile(.90),\
     df1['output_a_win'].quantile(.90)\
        , df1['operating_leverage_win'].\
            quantile(.90), df1['working_capital_win']\
.quantile(.90), df1['liquidity_ratio_win']\
    .quantile(.90), \
        df1['financial_leverage_win'].\
            quantile(.90)],
}, index = ['ROA', 'Total Assets', '\
    Operating Income', 'Market Share', '\
        Output Share of Labor', 'Output Share of Capital', \
            'Output Share of Assets', 'Operating Leverage', \
'Working Capital', 'Liquidity Ratio', \
    'Financial Leverage'])
```

```
# Export to Latex
summary_all_lat         = summary_all.to_latex(index=True)
title    = 'summary_all'
extension               = '.tex'
path     = 'N:/durable/file-export/\
    IssamSheikh/S2_profitability/Tables/' + title + extension

with open(path, 'w') as f:
    f.write(summary_all_lat)
```

Private

In [ ]:
```
#Summary Statistic
summary_private = pd.DataFrame({    'mean': [df2['ROA_win'].mean(),\
        df2['total_assets_win'].mean(),\
        df2['op_inc_win'].mean(),\
          df2['mrkt_share'].mean(), \
     df2['output_l_win'].mean(),\
df2['output_k_win'].mean(),\
            df2['output_a_win'].mean(), \
       df2['operating_leverage_win'].mean(),\
df2['working_capital_win'].mean(),\
            df2['liquidity_ratio_win'].mean(),\
        df2['financial_leverage_win'].mean()], \

'p10' : [df2['ROA_win'].quantile(.10), \
    df2['total_assets_win'].quantile(.10)\
        , df2['op_inc_win'].quantile(.10),\
             df2['mrkt_share'].quantile(.10),\
   df2['output_l_win'].quantile(.10)\
    , df2['output_k_win'].quantile(.10),\
         df2['output_a_win'].quantile(.10),\
             df2['operating_leverage_win'].quantile(.10), \
df2['working_capital_win'].quantile(.10), \
    df2['liquidity_ratio_win'].quantile(.10)\
        , df2['financial_leverage_win'].\
            quantile(.10)],
'p25' : [df2['ROA_win'].quantile(.25), \
    df2['total_assets_win'].quantile(.25),\
         df2['op_inc_win'].quantile(.25), \
            df2['mrkt_share'].quantile(.25)\
,   df2['output_l_win']\
    .quantile(.25), df2['output_k_win']\
        .quantile(.25), df2['output_a_win']\
            .quantile(.25), \
df2['operating_leverage_win']\
    .quantile(.25), \
        df2['working_capital_win']\
            .quantile(.25), df2['liquidity_ratio_win'].\
quantile(.25), df2['financial_leverage_win']\
    .quantile(.25)],
'p50' : [df2['ROA_win'].quantile(.50),\
     df2['total_assets_win'].quantile(.50),\
         df2['op_inc_win'].quantile(.50),  \
             df2['mrkt_share'].quantile(.50),\
   df2['output_l_win'].quantile(.50),\
     df2['output_k_win'].quantile(.50),\
         df2['output_a_win'].quantile(.50),\
        df2['operating_leverage_win']\
         .quantile(.50), df2['working_capital_win'\
            ].quantile(.50), df2['liquidity_\
ratio_win'].quantile(.50),\
     df2['financial_leverage_win'\
        ].quantile(.50)],
'p75' : [df2['ROA_win'].quantile(.75),\
     df2['total_assets_win'].quantile(.75)\
        , df2['op_inc_win'].quantile(.75), \
              df2['mrkt_share'].quantile(.75),\
   df2['output_l_win'].quantile(.75)\
    , df2['output_k_win'].quantile(.75),\
         df2['output_a_win'].quantile(.75), \
            df2['operating_leverage_win'].quantile(.75),\
 df2['working_capital_win'].quantile(.75),\
     df2['liquidity_ratio_win'].quantile(.75),\
         df2['financial_leverage_win'].quantile(.75)],
'p90' : [df2['ROA_win'].quantile(.90), df2['total_assets_win'].\
    quantile(.90), df2['op_inc_win'].quantile(.90), \
          df2['mrkt_share'].quantile(.90),  \
             df2['output_l_win'].quantile(.90), \
df2['output_k_win'].quantile(.90),\
     df2['output_a_win'].quantile(.90)\
        , df2['operating_leverage_win'].\
            quantile(.90), df2['working_capital_win']\
.quantile(.90), df2['liquidity_ratio_win']\
    .quantile(.90), \
        df2['financial_leverage_win'].\
            quantile(.90)],
}, index = ['ROA', 'Total Assets', '\
    Operating Income', 'Market Share', '\
        Output Share of Labor', 'Output Share of Capital', \
            'Output Share of Assets', 'Operating Leverage', \
'Working Capital', 'Liquidity Ratio', \
    'Financial Leverage'])
# Export to Latex
summary_private_lat         = summary_private\
    .to_latex(index=True)
title   = 'summary_private'
extension                   = '.tex'
path    = 'N:/durable/file-export/IssamSheikh\
    /S2_profitability/Tables/' + title + extension

with open(path, 'w') as f:
    f.write(summary_private_lat)
```

Public

```
In [ ]: #Summary Statistic
        summary_public = pd.DataFrame({     'mean': [df3['ROA_win'].mean(),\
                df3['total_assets_win'].mean(),\
                df3['op_inc_win'].mean(),\
                  df3['mrkt_share'].mean(), \
             df3['output_l_win'].mean(),\
        df3['output_k_win'].mean(),\
                    df3['output_a_win'].mean(), \
               df3['operating_leverage_win'].mean(),\
        df3['working_capital_win'].mean(),\
                    df3['liquidity_ratio_win'].mean(),\
                df3['financial_leverage_win'].mean()], \

        'p10' : [df3['ROA_win'].quantile(.10), \
            df3['total_assets_win'].quantile(.10)\
                , df3['op_inc_win'].quantile(.10),\
                     df3['mrkt_share'].quantile(.10),\
           df3['output_l_win'].quantile(.10)\
            , df3['output_k_win'].quantile(.10),\
                 df3['output_a_win'].quantile(.10),\
                     df3['operating_leverage_win'].quantile(.10), \
        df3['working_capital_win'].quantile(.10), \
            df3['liquidity_ratio_win'].quantile(.10)\
                , df3['financial_leverage_win'].\
                    quantile(.10)],
        'p25' : [df3['ROA_win'].quantile(.25), \
            df3['total_assets_win'].quantile(.25),\
                 df3['op_inc_win'].quantile(.25), \
                      df3['mrkt_share'].quantile(.25)\
        ,   df3['output_l_win']\
            .quantile(.25), df3['output_k_win']\
                .quantile(.25), df3['output_a_win']\
                    .quantile(.25), \
        df3['operating_leverage_win']\
            .quantile(.25), \
                df3['working_capital_win']\
                    .quantile(.25), df3['liquidity_ratio_win'].\
        quantile(.25), df3['financial_leverage_win']\
            .quantile(.25)],
        'p50' : [df3['ROA_win'].quantile(.50),\
             df3['total_assets_win'].quantile(.50),\
                 df3['op_inc_win'].quantile(.50),  \
                     df3['mrkt_share'].quantile(.50),\
           df3['output_l_win'].quantile(.50),\
             df3['output_k_win'].quantile(.50),\
                 df3['output_a_win'].quantile(.50),\
                df3['operating_leverage_win']\
                 .quantile(.50), df3['working_capital_win'\
                    ].quantile(.50), df3['liquidity_\
        ratio_win'].quantile(.50),\
             df3['financial_leverage_win'\
                ].quantile(.50)],
        'p75' : [df3['ROA_win'].quantile(.75),\
             df3['total_assets_win'].quantile(.75)\
                , df3['op_inc_win'].quantile(.75), \
              df3['mrkt_share'].quantile(.75),\
   df3['output_l_win'].quantile(.75)\
    , df3['output_k_win'].quantile(.75),\
         df3['output_a_win'].quantile(.75), \
            df3['operating_leverage_win'].quantile(.75),\
 df3['working_capital_win'].quantile(.75),\
     df3['liquidity_ratio_win'].quantile(.75),\
         df3['financial_leverage_win'].quantile(.75)],
'p90' : [df3['ROA_win'].quantile(.90), df3['total_assets_win'].\
    quantile(.90), df3['op_inc_win'].quantile(.90), \
          df3['mrkt_share'].quantile(.90),  \
             df3['output_l_win'].quantile(.90), \
df3['output_k_win'].quantile(.90),\
     df3['output_a_win'].quantile(.90)\
        , df3['operating_leverage_win'].\
            quantile(.90), df3['working_capital_win']\
.quantile(.90), df3['liquidity_ratio_win']\
    .quantile(.90), \
        df3['financial_leverage_win'].\
            quantile(.90)],
}, index = ['ROA', 'Total Assets', '\
    Operating Income', 'Market Share', '\
        Output Share of Labor', 'Output Share of Capital', \
            'Output Share of Assets', 'Operating Leverage', \
'Working Capital', 'Liquidity Ratio', \
    'Financial Leverage'])
# Export to Latex
summary_public_lat         = summary_publi\
    c.to_latex(index=True)
title   = 'summary_public'
extension              = '.tex'
path    = 'N:/durable/file-export/IssamSheik\
    h/S2_profitability/Tables/' + title + extension

with open(path, 'w') as f:
    f.write(summary_public_lat)
```

Listed

In [ ]:

```
#Summary Statistic
summary_list = pd.DataFrame({    'mean': [df4['ROA_win'].mean(),\
        df4['total_assets_win'].mean(),\
        df4['op_inc_win'].mean(),\
          df4['mrkt_share'].mean(), \
     df4['output_l_win'].mean(),\
df4['output_k_win'].mean(),\
            df4['output_a_win'].mean(), \
       df4['operating_leverage_win'].mean(),\
df4['working_capital_win'].mean(),\
            df4['liquidity_ratio_win'].mean(),\
        df4['financial_leverage_win'].mean()], \

'p10' : [df4['ROA_win'].quantile(.10), \
    df4['total_assets_win'].quantile(.10)\
         , df4['op_inc_win'].quantile(.10),\
                df4['mrkt_share'].quantile(.10),\
   df4['output_l_win'].quantile(.10)\
    , df4['output_k_win'].quantile(.10),\
         df4['output_a_win'].quantile(.10),\
             df4['operating_leverage_win'].quantile(.10), \
df4['working_capital_win'].quantile(.10), \
    df4['liquidity_ratio_win'].quantile(.10)\
        , df4['financial_leverage_win'].\
            quantile(.10)],
'p25' : [df4['ROA_win'].quantile(.25), \
    df4['total_assets_win'].quantile(.25),\
         df4['op_inc_win'].quantile(.25), \
              df4['mrkt_share'].quantile(.25)\
,   df4['output_l_win']\
    .quantile(.25), df4['output_k_win']\
        .quantile(.25), df4['output_a_win']\
            .quantile(.25), \
df4['operating_leverage_win']\
    .quantile(.25), \
        df4['working_capital_win']\
            .quantile(.25), df4['liquidity_ratio_win'].\
quantile(.25), df4['financial_leverage_win']\
    .quantile(.25)],
'p50' : [df4['ROA_win'].quantile(.50),\
     df4['total_assets_win'].quantile(.50),\
         df4['op_inc_win'].quantile(.50),  \
             df4['mrkt_share'].quantile(.50),\
   df4['output_l_win'].quantile(.50),\
     df4['output_k_win'].quantile(.50),\
         df4['output_a_win'].quantile(.50),\
        df4['operating_leverage_win']\
         .quantile(.50), df4['working_capital_win'\
            ].quantile(.50), df4['liquidity_\
ratio_win'].quantile(.50),\
     df4['financial_leverage_win'\
        ].quantile(.50)],
'p75' : [df4['ROA_win'].quantile(.75),\
     df4['total_assets_win'].quantile(.75)\
        , df4['op_inc_win'].quantile(.75), \
              df4['mrkt_share'].quantile(.75),\
   df4['output_l_win'].quantile(.75)\
    , df4['output_k_win'].quantile(.75),\
         df4['output_a_win'].quantile(.75), \
            df4['operating_leverage_win'].quantile(.75),\
 df4['working_capital_win'].quantile(.75),\
     df4['liquidity_ratio_win'].quantile(.75),\
         df4['financial_leverage_win'].quantile(.75)],
'p90' : [df4['ROA_win'].quantile(.90), df4['total_assets_win'].\
    quantile(.90), df4['op_inc_win'].quantile(.90), \
          df4['mrkt_share'].quantile(.90),  \
             df4['output_l_win'].quantile(.90), \
df4['output_k_win'].quantile(.90),\
     df4['output_a_win'].quantile(.90)\
        , df4['operating_leverage_win'].\
            quantile(.90), df4['working_capital_win']\
.quantile(.90), df4['liquidity_ratio_win']\
    .quantile(.90), \
        df4['financial_leverage_win'].\
            quantile(.90)],
}, index = ['ROA', 'Total Assets', '\
    Operating Income', 'Market Share', '\
        Output Share of Labor', 'Output Share of Capital', \
            'Output Share of Assets', 'Operating Leverage', \
'Working Capital', 'Liquidity Ratio', \
    'Financial Leverage'])
# Export to Latex
summary_list_lat          = summary_list.to_l\
    atex(index=True)
title   = 'summary_list'
extension               = '.tex'
path    = 'N:/durable/file-export/IssamSheikh/\
    S2_profitability/Tables/' + title + extension

with open(path, 'w') as f:
    f.write(summary_list_lat)
```

Nonlist

In [ ]:

```
#Summary Statistic
summary_nonlist = pd.DataFrame({     'mean': [df5['ROA_win'].mean(),\
        df5['total_assets_win'].mean(),\
        df5['op_inc_win'].mean(),\
          df5['mrkt_share'].mean(), \
     df5['output_l_win'].mean(),\
df5['output_k_win'].mean(),\
            df5['output_a_win'].mean(), \
       df5['operating_leverage_win'].mean(),\
df5['working_capital_win'].mean(),\
            df5['liquidity_ratio_win'].mean(),\
        df5['financial_leverage_win'].mean()], \

'p10' : [df5['ROA_win'].quantile(.10), \
    df5['total_assets_win'].quantile(.10)\
        , df5['op_inc_win'].quantile(.10),\
               df5['mrkt_share'].quantile(.10),\
   df5['output_l_win'].quantile(.10)\
    , df5['output_k_win'].quantile(.10),\
         df5['output_a_win'].quantile(.10),\
             df5['operating_leverage_win'].quantile(.10), \
df5['working_capital_win'].quantile(.10), \
    df5['liquidity_ratio_win'].quantile(.10)\
        , df5['financial_leverage_win'].\
            quantile(.10)],
'p25' : [df5['ROA_win'].quantile(.25), \
    df5['total_assets_win'].quantile(.25),\
         df5['op_inc_win'].quantile(.25), \
              df5['mrkt_share'].quantile(.25)\
,   df5['output_l_win']\
    .quantile(.25), df5['output_k_win']\
         .quantile(.25), df5['output_a_win']\
             .quantile(.25), \
df5['operating_leverage_win']\
     .quantile(.25), \
         df5['working_capital_win']\
             .quantile(.25), df5['liquidity_ratio_win'].\
quantile(.25), df5['financial_leverage_win']\
     .quantile(.25)],
'p50' : [df5['ROA_win'].quantile(.50),\
      df5['total_assets_win'].quantile(.50),\
          df5['op_inc_win'].quantile(.50),  \
              df5['mrkt_share'].quantile(.50),\
   df5['output_l_win'].quantile(.50),\
     df5['output_k_win'].quantile(.50),\
         df5['output_a_win'].quantile(.50),\
         df5['operating_leverage_win']\
          .quantile(.50), df5['working_capital_win'\
             ].quantile(.50), df5['liquidity_\
ratio_win'].quantile(.50),\
     df5['financial_leverage_win'\
         ].quantile(.50)],
'p75' : [df5['ROA_win'].quantile(.75),\
     df5['total_assets_win'].quantile(.75)\
         , df5['op_inc_win'].quantile(.75), \
              df5['mrkt_share'].quantile(.75),\
   df5['output_l_win'].quantile(.75)\
    , df5['output_k_win'].quantile(.75),\
         df5['output_a_win'].quantile(.75), \
            df5['operating_leverage_win'].quantile(.75),\
 df5['working_capital_win'].quantile(.75),\
     df5['liquidity_ratio_win'].quantile(.75),\
         df5['financial_leverage_win'].quantile(.75)],
'p90' : [df5['ROA_win'].quantile(.90), df5['total_assets_win'].\
    quantile(.90), df5['op_inc_win'].quantile(.90), \
          df5['mrkt_share'].quantile(.90),  \
             df5['output_l_win'].quantile(.90), \
df5['output_k_win'].quantile(.90),\
     df5['output_a_win'].quantile(.90)\
         , df5['operating_leverage_win'].\
             quantile(.90), df5['working_capital_win']\
.quantile(.90), df5['liquidity_ratio_win']\
    .quantile(.90), \
        df5['financial_leverage_win'].\
            quantile(.90)],
}, index = ['ROA', 'Total Assets', '\
    Operating Income', 'Market Share', '\
        Output Share of Labor', 'Output Share of Capital', \
            'Output Share of Assets', 'Operating Leverage', \
'Working Capital', 'Liquidity Ratio', \
    'Financial Leverage'])
# Export to Latex\

summary_nonlist_lat         = summary_nonlist.\
    to_latex(index=True)
title    = 'summary_nonlist'
extension                = '.tex'
```

```
path    = 'N:/durable/file-export/IssamSheikh\
    /S2_profitability/Tables/' + title + extension

with open(path, 'w') as f:
    f.write(summary_nonlist_lat)
```

Export to STATA

In [ ]:
```
# Save pandas dataframe as dta file
df1.to_stata('N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_full.dta')
df2.to_stata('N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_private.dta')
df3.to_stata('N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_public.dta')
df4.to_stata('N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_listed.dta')
df5.to_stata('N:/durable/BIStudents/IssamSheikh/data/clean/reg1/reg1_\
    nonlisted.dta')
```

PLOTS

In [ ]:
```
# Plotting all in one graph, 4 rows and 3 columns
import matplotlib.ticker as ticker
fig, axs     = plt.subplots(2, 3, figsize=(6.3, 9))

## First row
#ROA
axs[0, 0].plot(df_private.groupby('year')['ROA_win'].mean(),  \
    olor='darkblue',         label='Private Companies')
axs[0, 0].plot(df_nonlist.groupby('year')['ROA_win'].mean(),  \
    color='darkred',         label='Nonlisted Companies')
axs[0, 0].plot(df_list.groupby('year')['ROA_win'].mean(),    \
     color='darkgreen',         label='Listed Companies')
# Adjust to the appropriate labels and title
axs[0, 0].set_xlim(2007,2019)
#axs[0, 0].set_ylim(500000, 2000000)
#axs[0, 0].set_xlabel('Year')
axs[0, 0].set_ylabel('ROA')
axs[0, 0].set_title('Return On Assets')

# Total Assets
axs[0, 1].plot(df_private.groupby('year')['total_assets_win'].mean(), \
     color='darkblue',         label='Private Companies')
axs[0, 1].plot(df_nonlist.groupby('year')['total_assets_win'].mean(),\
      color='darkred',         label='Nonlisted Companies')
axs[0, 1].plot(df_list.groupby('year')['total_assets_win'].mean(), \
        color='darkgreen',        label='Nonlisted Companies')
# Formatting:
#formatter1    = ticker.FuncFormatter(lambda x, pos: f'{x/0:.1f}M')
#axs[0,1].yaxis.set_major_formatter(formatter1)
# Adjust to the appropriate labels and title
axs[0, 1].set_xlim(2007,2019)
#axs[0, 1].set_ylim(1000000, 3900000)
```

```python
#axs[0, 1].set_xlabel('Year')
axs[0, 1].set_ylabel('KR')
axs[0, 1].set_title('Total Assets')


# Operating Income
axs[0, 2].plot(df_private.groupby('year')['op_inc_win'].mean(),  \
    color='darkblue',         label='Private Companies')
axs[0, 2].plot(df_nonlist.groupby('year')['op_inc_win'].mean(), \
     color='darkred',         label='Nonlisted Companies')
axs[0, 2].plot(df_list.groupby('year')['op_inc_win'].mean(),   \
      color='darkgreen',        label='Nonlisted Companies')
# Formatting:
#formatter2   = ticker.FuncFormatter(lambda x, pos: f'{x/1e6:.2f}M')
#axs[0, 2].yaxis.set_major_formatter(formatter2)
# Adjust to the appropriate labels and title
#axs[0, 2].set_xlim(2007,2019)
#axs[0, 2].set_ylim(0.025, None)
#axs[0, 2].set_xlabel('Year')
axs[0, 2].set_ylabel('KR')
axs[0, 2].set_title('Operating Income')

## Second row
# Market Share
axs[1, 0].plot(df_private.groupby('year')['mrkt_share'].mean(),  \
    color='darkblue',         label='Private Companies')
axs[1, 0].plot(df_nonlist.groupby('year')['mrkt_share'].mean(),  \
    color='darkred',         label='Nonlisted Companies')\
        color='darkgreen',         label='Listed Companies')
# Adjust to the appropriate labels and title
#axs[1, 0].set_xlim(2007,2019)
#axs[0, 0].set_ylim(500000, 2000000)
#axs[0, 0].set_xlabel('Year')
axs[1, 0].set_ylabel('%')
axs[1, 0].set_title('Market Share')

# Operating Leverage
axs[1, 1].plot(df_private.groupby('year')['operating_leverage_win'].mean(), \
     color='darkblue',         label='Private Companies')
axs[1, 1].plot(df_nonlist.groupby('year')['operating_leverage_win'].mean(), \
     color='darkred',         label='Nonlisted Companies')
axs[1, 1].plot(df_list.groupby('year')['operating_leverage_win'].mean(),   \
      color='darkgreen',         label='Listed Companies')
# Adjust to the appropriate labels and title
#axs[1, 1].set_xlim(2007,2019)
#axs[0, 0].set_ylim(500000, 2000000)
#axs[0, 0].set_xlabel('Year')
axs[1, 1].set_title('Operating Leverage')

# Working Capital
axs[1, 2].plot(df_private.groupby('year')['working_capital_win'].mean(),\
    color='darkblue',         label='Private Companies')\

axs[1, 2].plot(df_nonlist.groupby('year')['working_capital_win'].mean(),\
      color='darkred',         label='Nonlisted Companies')
axs[1, 2].plot(df_list.groupby('year')['working_capital_win'].mean(),   \
      color='darkgreen',         label='Listed Companies')
# Adjust to the appropriate labels and title
#axs[1, 2].set_xlim(2007,2019)
#axs[0, 0].set_ylim(500000, 2000000)
#axs[0, 0].set_xlabel('Year')
axs[1, 2].set_title('Working Capital')

plt.tight_layout()
```

In [ ]:
```
## Third row
# Output Share for Labor
axs[2, 0].plot(df_private.groupby('year')['output_l_win'].mean(),  \
    color='darkblue',         label='Private Companies')
axs[2, 0].plot(df_nonlist.groupby('year')['output_l_win'].mean(),  \
    color='darkred',         label='Nonlisted Companies')
axs[2, 0].plot(df_list.groupby('year')['output_l_win'].mean(),     \
    color='darkgreen',         label='Listed Companies')
# Adjust to the appropriate labels and title
axs[2, 0].set_xlim(2007,2019)
#axs[0, 0].set_ylim(500000, 2000000)
#axs[0, 0].set_xlabel('Year')
#axs[2, 0].set_ylabel('%')
axs[2, 0].set_title('Output Share of Labor')

# Output Share of capital
axs[2, 1].plot(df_private.groupby('year')['output_k_win'].mean(),  \
    color='darkblue',         label='Private Companies')
axs[2, 1].plot(df_nonlist.groupby('year')['output_k_win'].mean(),  \
    color='darkred',         label='Nonlisted Companies')
axs[2, 1].plot(df_list.groupby('year')['output_k_win'].mean(),    \
     color='darkgreen',         label='Listed Companies')
# Adjust to the appropriate labels and title
axs[2, 1].set_xlim(2007,2019)
#axs[0, 0].set_ylim(500000, 2000000)
#axs[0, 0].set_xlabel('Year')
#axs[2, 0].set_ylabel('%')
axs[2, 1].set_title('Output Share of Capital')


# Output Share of Assets
axs[2, 2].plot(df_private.groupby('year')['output_a_win'].mean(),  \
    color='darkblue',         label='Private Companies')
axs[2, 2].plot(df_nonlist.groupby('year')['output_a_win'].mean(),  \
    color='darkred',         label='Nonlisted Companies')
axs[2, 2].plot(df_list.groupby('year')['output_a_win'].mean(),     \
    color='darkgreen',         label='Listed Companies')
# Adjust to the appropriate labels and title
axs[2, 2].set_xlim(2007,2019)
#axs[0, 0].set_ylim(500000, 2000000)
#axs[0, 0].set_xlabel('Year')
#axs[2, 0].set_ylabel('%')
axs[2, 2].set_title('Output Share of Assets')

## Fourth row
# Liquidity Ratio
axs[3,0].plot(df_private.groupby('year')['liquidity_ratio_win'].mean(), \
      color='darkblue',         label='Private Companies')
axs[3,0].plot(df_nonlist.groupby('year')['liquidity_ratio_win'].mean(),\
       color='darkred',         label='Nonlisted Companies')
axs[3,0].plot(df_list.groupby('year')['liquidity_ratio_win'].mean(),    \
       color='darkgreen',         label='Listed Companies')
# Adjust to the appropriate labels and title
axs[3,0].set_xlim(2007,2019)
#axs[0, 0].set_ylim(500000, 2000000)
#axs[0, 0].set_xlabel('Year')
#axs[2, 0].set_ylabel('%')
axs[3,0].set_title('Liquidity Ratio')

# Financial Leverage
axs[3,1].plot(df_private.groupby('year')['financial_leverage_win'].mean(),\
       color='darkblue',         label='Private Companies')
axs[3,1].plot(df_nonlist.groupby('year')['financial_leverage_win'].mean(),\
       color='darkred',         label='Nonlisted Companies')
axs[3,1].plot(df_list.groupby('year')['financial_leverage_win'].mean(),  \
        color='darkgreen',         label='Listed Companies')
# Adjust to the appropriate labels and title
axs[3,1].set_xlim(2007,2019)
#axs[0, 0].set_ylim(500000, 2000000)
#axs[0, 0].set_xlabel('Year')
#axs[2, 0].set_ylabel('%')
axs[3,1].set_title('Financial Leverage')

handles, labels = axs[0, 0].get_legend_handles_labels()
fig.legend(handles, labels, loc='lower right')

#plt.subplots_adjust(left=0.05, right=0.95, bottom=0.05,\
#  top=0.9, wspace=0.2, hspace=0.4)
#fig.text(0.5, 0.95, 'Private vs. Non-listed vs Listed\
#  Companies', ha='center', va='center', fontsize=14)

plt.tight_layout()


save_path     = 'N:/durable/file-export/IssamSheikh/\
    S2_profitability/all_in_one.jpg'
plt.savefig(save_path, format = 'jpeg')

plt.show()
```

In [ ]:
```python
# Import needed libraries
import pandas as pd
import numpy as np
pd.set_option('mode.use_inf_as_na', True)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline
from scipy.stats.mstats import winsorize
import pyarrow as pa
```

In [ ]:
```python
# Read Dataset
df_full = pd.read_parquet('N:/durable/BIStudents/\
    IssamSheikh/data/clean/full_sample.parquet')
```

Winsorising

Level variable

In [ ]:
```python
# Winsorising total assets
df_full['total_assets_win'] = winsorize(df_full\
    ['total_assets'], (0.01, 0.01))
```

Ratios

In [ ]:
```python
# Winsorising ROA
df_full['ROA_win'] = winsorize(df_full['ROA'], (0.01, 0.01))
```

STATA

Figure 1

In [ ]:
```python
743950627/50
```

In [ ]:
```python
df_fig1 = df_full[['orgnr', 'year', 'ROA', 'ROA_win', \
    'total_assets', 'total_assets_win']]

df_fig1.to_stata('N:/durable/BIStudents/IssamSheikh/\
    data/clean/STATA/df_figures/fig1.dta')
```

Figure 2

In [ ]:
```python
df_fig2 =
```

Figures

```
In [ ]: # Create bins for the x variable
num_bins = 100
bins = np.linspace(df_full['total_assets_win'].min(),\
     df_full['total_assets_win'].max(), num_bins+1)
df_full['bin'] = pd.cut(df_full['total_assets_win'],\
     bins, labels=False)

# Calculate the mean and percentiles for each bin
f1 = pd.DataFrame()
f1['bin_means'] = df_full.groupby('bin')['ROA'].mean()
f1['bin_q01'] = df_full.groupby('bin')['ROA'].quantile(0.01)
f1['bin_q99'] = df_full.groupby('bin')['ROA'].quantile(0.99)
f1['bin'] = f1.index

# Get the left endpoints of the bins
bin_left = bins[:-1]

# Scatter plot with binned means and percentiles
#plt.scatter(df_full['total_assets_win'], \
# df_full['ROA'], alpha=0.5, label='Data')
plt.scatter(bins[:-1], f1['bin_means'],\
     marker='o', label='Binned Means')
plt.fill_between(f1['bin'], f1['bin_q01'],\
    ['bin_q99'], color='red')
#plt.errorbar(bin_left, bin_means, \
# yerr=[bin_means - bottom_percentile, \
# top_percentile - bin_means], fmt='o', color='red',\
#  label='5th-95th Percentiles', capsize=3)
plt.xlabel('Total Assets Win')
plt.ylabel('ROA_win')
plt.title('Scatter Plot with Binned Means and 5th-95th Percentiles')
plt.legend()
plt.show()

#plt.scatter(df_full['mrkt_share'], df_full['ROA_win'], alpha=0.5, label='Data')
#plt.scatter(bins[:-1], y_means, marker='o', label='Binned Means')
```

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

# Create bins for the x variable
num_bins = 100
bins = np.linspace(df_full['total_assets_win'].min(), \
    df_full['total_assets_win'].max(), num_bins+1)
df_full['bin'] = pd.cut(df_full['total_assets_win'], bins, labels=False)

# Calculate the mean and percentiles for each bin
f1 = pd.DataFrame()
f1['bin_means'] = df_full.groupby('bin')['ROA'].mean()
f1['bin_q01'] = df_full.groupby('bin')['ROA'].quantile(0.01)
```

```
f1['bin_q99'] = df_full.groupby('bin')['ROA'].quantile(0.99)
f1['bin'] = f1.index

# Scatter plot with binned means and percentiles
plt.scatter(f1['bin'], f1['bin_means'], marker='o', label='Binned Means')

for i in range(len(f1)):
    plt.fill_between([f1['bin'].iloc[i]], f1['bin_q01'].iloc[i], \
        f1['bin_q99'].iloc[i], color='red', alpha=0.2)

plt.xlabel('Bin')
plt.ylabel('ROA_win')
plt.title('Scatter Plot with Binned Means and 1st-99th Percentiles')
plt.legend()
plt.show()
```

In [ ]:
```
# Create bins for the x variable
num_bins = 20
bins = np.linspace(min(df_full['mrkt_share']), \
    max(df_full['mrkt_share']), num_bins+1)
x_bins = np.digitize(df_full['mrkt_share'], bins)

# Calculate the mean y value for each bin
y_means = [np.mean(df_full['ROA_win'][x_bins == i]) for i \
    in range(1, num_bins+1)]

# Scatter plot with binned variables
plt.scatter(df_full['mrkt_share'], df_full['ROA_win'], \
    alpha=0.5, label='Data')
plt.scatter(bins[:-1], y_means, marker='o', \
    label='Binned Means')
plt.xlabel('X')
plt.ylabel('Y')
#plt.xlim(0.5,)
plt.title('Scatter Plot with Binned Variable')
plt.legend()
plt.show()
```

Figure 3

In [ ]:
```
# Variables of interest
f3 = df_full[['year', 'orgnr', 'ROA_win', 'total_assets',\
    'to_listed', 'public_listed']]
f3 = f3[f3['total_assets'] > 10000000]

# Calculate the mean ROA for each group
go_listed = f3[f3['to_listed'] == 1]['ROA_win']
not_go_listed = f3[(f3['to_listed'] == 0) & (f3['public_listed']\
     != 1)]['ROA_win']
data = [go_listed, not_go_listed]

# Create a box plot with labels
boxprops = dict(facecolor='lightgrey', edgecolor='grey', \
    linewidth=2)
plt.boxplot(data, labels=['Go Listed', "Don't Go Listed"],\
     patch_artist=True, boxprops=boxprops, widths=0.9)
plt.title('Firms Got Listed - ROA Comparison')
plt.xlabel('Firm Status')
plt.ylabel('ROA')
```

Figure 4

In [ ]:
```
# Variables of interest
f4 = df_full[['year', 'orgnr', 'ROA_win', \
    'total_assets_win', 'to_listed']]

# Keep only companies that got listed
org_listed = f4[f4['to_listed'] == 1]['orgnr']
f4 = f4.merge(org_listed, on='orgnr', how='right')

# Transforming listing year
listing_year = f4[f4['to_listed'] == 1][['orgnr', 'year']]
f4 = f4.merge(listing_year, on='orgnr', how='inner')

# Rename years
f4.rename(columns={'year_x': 'year', 'year_y': \
    'listing_year'}, inplace=True)

# Computing year difference
f4['year_difference'] = f4['year'] - f4['listing_year']

# ROA means by year difference
mean = pd.DataFrame()
mean['ROA_win'] = f4.groupby('year_difference')\
    ['ROA_win'].mean()
mean['year_difference'] = mean.index

# Keep only +/- 10 years
mean = mean[(mean['year_difference'] >= -7) & \
    (mean['year_difference'] <= 7)]
```

In [ ]:
```
# Linear Regressing
# Fit a line using polyfit mean['year_difference'] <= 0
mean_low = mean[mean['year_difference'] <= 0]
coefficients = np.polyfit(mean_low['year_difference'],\
     mean_low['ROA_win'], 1)
m, b = coefficients
# Calculate the y values for the fitted line
fit_line_low = m * mean_low['year_difference'] + b
# Calculate the residuals
residuals_low = mean_low['ROA_win'] - fit_line_low
# Calculate the standard error
std_error_low = np.std(residuals_low)
# Calculate the 95% confidence interval
confidence_interval_low = 1.645 * std_error_low #1.96 for 95%CI
# Calculate the upper and lower bounds of the confidence interval
upper_bound_low = fit_line_low + confidence_interval_low
lower_bound_low = fit_line_low - confidence_interval_low
```

```
# Fit a line using polyfit mean['year_difference'] >= 0
mean_high = mean[mean['year_difference'] >= 0]
coefficients = np.polyfit(mean_high['year_difference'], mean_high['ROA_win'], 1)
m, b = coefficients
# Calculate the y values for the fitted line
fit_line_high = m * mean_high['year_difference'] + b
# Calculate the residuals
residuals_high = mean_high['ROA_win'] - fit_line_high
# Calculate the standard error
std_error_high = np.std(residuals_high)
# Calculate the 95% confidence interval
confidence_interval_high = 1.645 * std_error_high
# Calculate the upper and higher bounds of the confidence interval
upper_bound_high = fit_line_high + confidence_interval_high
lower_bound_high = fit_line_high - confidence_interval_high
```

In [ ]:
```
# Scatter plot with bin means
plt.scatter(mean['year_difference'], mean['ROA_win'], color='grey')

# Plot the fitted lines
plt.plot(mean_low['year_difference'], fit_line_low, \
    color='darkred', label='Fitted Line')
plt.plot(mean_high['year_difference'], fit_line_high, \
    olor='darkblue', label='Fitted Line')

# Plot the confidence interval
plt.fill_between(mean_low['year_difference'], lower_bound_low, \
    upper_bound_low, color='gray', alpha=0.3,\
         label='90% Confidence Interval')
plt.fill_between(mean_high['year_difference'], lower_bound_high,\
     upper_bound_high, color='gray', alpha=0.3, \
        label='90% Confidence Interval')
plt.xlabel('Year after listing')
plt.ylabel('ROA')
#plt.ylim(-0.075,0.075)
#plt.title('Scatter Plot with Bin Means')
#plt.legend()
#plt.show()
```

In [ ]:
```
# Scatter plot with bin median
plt.scatter(mean['year_difference'], mean['ROA_win'], color='grey')

# Plot the fitted lines
plt.plot(mean_low['year_difference'], fit_line_low, \
    color='darkred', label='Fitted Line')
plt.plot(mean_high['year_difference'], fit_line_high, \
    color='darkblue', label='Fitted Line')

# Plot the confidence interval
plt.fill_between(mean_low['year_difference'], lower_bound_low,\
     upper_bound_low, color='gray', alpha=0.3, \
        abel='90% Confidence Interval')
plt.fill_between(mean_high['year_difference'], lower_bound_high,\
     upper_bound_high, color='gray', alpha=0.3, \
        label='90% Confidence Interval')
```

```
plt.xlabel('Year after listing')
plt.ylabel('ROA')
plt.title('Scatter Plot with Bin Median')
#plt.legend()
#plt.show()
```

In [ ]:
```
# Quadratic Regressing
# Fit a line using polyfit mean['year_difference'] <= 0
mean_low = mean[mean['year_difference'] <= 0]
coefficients_low = np.polyfit(mean_low['year_difference'], \
    mean_low['ROA_win'], 2)
a_low, b_low, c_low = coefficients_low
# Calculate the y values for the fitted curve
fit_curve_low = a_low * mean_low['year_difference']**2 +\
     b_low * mean_low['year_difference'] + c_low
# Calculate the residuals
residuals_low = mean_low['ROA_win'] - fit_curve_low
# Calculate the standard error
std_error_low = np.std(residuals_low)
# Calculate the 95% confidence interval
confidence_interval_low = 1.645 * std_error_low #1.96 for 95%CI
# Calculate the upper and lower bounds of the confidence interval
upper_bound_low = fit_curve_low + confidence_interval_low
lower_bound_low = fit_curve_low - confidence_interval_low


# Fit a line using polyfit mean['year_difference'] >= 0
mean_high = mean[mean['year_difference'] >= 0]
coefficients_high = np.polyfit(mean_high['year_difference'], \
    mean_high['ROA_win'], 2)
a_high, b_high, c_high = coefficients_high
# Calculate the y values for the fitted curve
fit_curve_high = a_high * mean_high['year_difference']**2 + b_high *\
     mean_high['year_difference'] + c_high
# Calculate the residuals
residuals_high = mean_high['ROA_win'] - fit_curve_high
# Calculate the standard error
std_error_high = np.std(residuals_high)
# Calculate the 95% confidence interval
confidence_interval_high = 1.645 * std_error_high
# Calculate the upper and higher bounds of the confidence interval
upper_bound_high = fit_curve_high + confidence_interval_high
lower_bound_high = fit_curve_high - confidence_interval_high
```

In [ ]:
```
# Scatter plot with bin means
plt.scatter(mean['year_difference'], mean['ROA_win'], color='grey')

# Plot the fitted lines
plt.plot(mean_low['year_difference'], fit_curve_low, \
    color='darkred', label='Fitted Line')
plt.plot(mean_high['year_difference'], fit_curve_high, \
    color='darkblue', label='Fitted Line')

# Plot the confidence interval
plt.fill_between(mean_low['year_difference'], lower_bound_low, \
    upper_bound_low, color='gray', alpha=0.3, label='90% Confidence Interval')
plt.fill_between(mean_high['year_difference'], lower_bound_high,\
     upper_bound_high, color='gray', alpha=0.3, label='90% Confidence Interval')
plt.xlabel('Year after listing')
plt.ylabel('ROA')
#plt.title('Scatter Plot with Bin Means')
#plt.legend()
#plt.show()
```

In [ ]:
```
# Scatter plot with bin median
plt.scatter(mean['year_difference'], mean['ROA_win'], color='grey')

# Plot the fitted lines
plt.plot(mean_low['year_difference'], fit_curve_low, \
    color='darkred', label='Fitted Line')
plt.plot(mean_high['year_difference'], fit_curve_high, \
    color='darkblue', label='Fitted Line')

# Plot the confidence interval
plt.fill_between(mean_low['year_difference'], lower_bound_low,\
     upper_bound_low, color='gray', alpha=0.3,\
         label='90% Confidence Interval')
plt.fill_between(mean_high['year_difference'], lower_bound_high,\
     upper_bound_high, color='gray', alpha=0.3, \
        label='90% Confidence Interval')
plt.xlabel('Year after listing')
plt.ylabel('ROA')
plt.title('Scatter Plot with Bin Median')
#plt.legend()
#plt.show()
```

Figure 5

In [ ]:
```
# By Industry
f5 = df_full[['orgnr', 'year', 'to_listed', 'industry']]

f51 = pd.DataFrame()
f51['to_listed'] = f5.groupby('industry')['to_listed'].sum()
f51['industry'] = f51.index
f51 = f51[f51['to_listed'] != 0]
f51['industry'] = f51['industry'].astype(str)
```

In [ ]:
```
# Create the figure and axis
fig, ax = plt.subplots(figsize=(10, 6))

# Add grid lines
ax.grid(True, axis='y', linestyle='--', color='gray', alpha=0.5)

# Plot the bars
ax.bar(f51['industry'], f51['to_listed'], color='darkblue')

# Customize other plot settings (e.g., labels, title)
ax.set_xlabel('Industry')
ax.set_ylabel('Number of Firms Listed')
ax.set_title('Number of Firms Listed by Industry')
```

```
plt.xticks(rotation=45)
plt.show()
```

In [ ]:
```
# By Year
f5 = df_full[['orgnr', 'year', 'to_listed', 'industry']]

f52 = pd.DataFrame()
f52['to_listed'] = f5.groupby('year')['to_listed'].sum()
f52['year'] = f52.index
```

In [ ]:
```
# Create the figure and axis - Industry
fig, ax = plt.subplots(figsize=(10, 6))

# Add grid lines
ax.grid(True, axis='y', linestyle='--', color='gray', alpha=0.5)

# Plot the bars
ax.bar(f52['year'], f52['to_listed'], color='darkblue')

# Customize other plot settings (e.g., labels, title)
ax.set_xlabel('Year')
ax.set_ylabel('Number of Firms Listed')
ax.set_title('Number of Firms Listed by Industry')
plt.xticks(rotation=45)
plt.show()
```

In [ ]:
```
pip install nbconvert
```

In [ ]:
```
# Import needed libraries
import pandas as pd
import numpy as np
import pyarrow.parquet as pq
pd.set_option('mode.use_inf_as_na', True)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
%matplotlib inline
from tabulate import tabulate
from scipy.stats.mstats import winsorize
import pyarrow as pa
import re

# Display 6 columns for viewing pupuses
pd.set_option('display.max_columns', 10)
# Reduce decimals points to 2
pd.options.display.float_format = '{:,.2f}'.format
```

In [ ]:
```
# Read Dataset
df_full = pd.read_parquet('N:/durable/BIStudents/IssamSheikh\
    /data/clean/full_sample.parquet')
```

In [ ]:
```
# Variables of interest
df_probit = df_full[df_full['total_assets'] > 10000000][['orgnr', \
             'year', 'to_listed', 'public', 'industry', 'total_assets',\
  'rev', 'capex', \
  'ROA', 'growth', 'financial_leverage', \
  'output_l', 'output_k', 'output_a']]
```

Winsorizing

Level Variables

In [ ]:
```
# Revenue
# Logged Revenue
df_probit['lg_rev'] = np.log(df_probit['rev'])
#df_probit['lg_rev'].hist()
```

In [ ]:
```
# CAPEX
#df_probit['capex_win'] = winsorize(df_probit['capex'], (0.01, 0.01))
#df_probit['capex_win'].hist()
#df_probit['capex'].describe()
```

In [ ]:
```
# Growth
#df_probit['growth_win'] = winsorize(df_probit['growth'], (0.01, 0.01))
```

```
#df_probit['growth'].hist()
#df_probit['growth'].describe()
```

Ratio variables

In [ ]:
```
# ROA
# Winsorize ROA
df_probit['ROA_win'] = winsorize(df_probit['ROA'], (0.01, 0.01))
#df_probit['ROA_win'].hist()
#df_probit['ROA_win'].isna().sum()
```

In [ ]:
```
# Financial Leverage
#df_probit['financial_leverage_win'] = \
#winsorize(df_probit['financial_leverage'], (0.01, 0.01))
#df_probit['financial_leverage'].hist()
#df_probit['fin_lev_win'].isna().sum()
```

In [ ]:
```
# Gowth
# Set a maximum value for 'growth' column )99th percentile)
df_probit['growth_win'] = np.where(df_probit['growth'] >\
              df_probit['growth'].quantile(0.99), \
                df_probit['growth'].quantile(0.99), \
df_probit['growth'])
#df_probit['growth_win'].describe()
```

In [ ]:
```
# Bank rate
#bank_rate = df_probit[['industry', 'int_rate']]
```

Productivity

In [ ]:
```
## output_l, output_k, output_a
# Winsorize output_l
df_probit['output_l_win'] = winsorize(df_probit['output_l'], (0.01, 0.01))
df_probit['lg_output_l'] = np.log(df_probit['output_l'])
#df_probit['output_l'].hist()

# Winsorize output_k
df_probit['output_k_win'] = winsorize(df_probit['output_k'], (0.01, 0.01))
df_probit['lg_output_k'] = np.log(df_probit['output_k'])
#df_probit['lg_output_k'].hist()

# Winsorize output_a
df_probit['output_a_win'] = winsorize(df_probit['output_a'], (0.01, 0.01))
df_probit['lg_output_a'] = np.log(df_probit['output_a'])
#df_probit['lg_output_a'].hist()
```

Probit regression data

```
In [ ]: # Check for missing values
        #print(df_probit.isnull().sum())
```

```
In [ ]: # Drop public firms if not listing year
        df_probit = df_probit[((df_probit['public'] == 1) & (df_probit['to_listed'] == 1))
```

```
In [ ]: df = df_probit[['orgnr', 'year', 'to_listed', 'industry',\
                        'lg_rev', 'capex', 'growth_win',\
                        'ROA_win', 'financial_leverage', \
                        'lg_output_l', 'lg_output_k', 'lg_output_a']]
```

```
In [ ]: df_full[df_full['public'] == 1].groupby('orgnr')['age'].max().describe()
```

```
In [ ]: p0 = df_probit[df_probit['to_listed'] == 0]
        p1 = df_probit[df_probit['to_listed'] == 1]

        failure = len(p0)
        success = len(p1)
        success / (failure + success)
```

```
In [ ]: tst2 =df_full.groupby('industry')['tot_fixe_assets'].describe()
        tst2 = tst2.sort_values(by='min')
```

```
In [ ]: tst = df_probit[(df_probit['industry'] == 71) | (df_probit['industry'] == 72) | (df
        tst1 =tst.groupby('industry')['total_assets'].describe()
```

```
In [ ]: # Save pandas dataframe as dta file
        df_probit.to_stata('N:/durable/BIStudents/\
        IssamSheikh/data/clean/reg2/reg2_full.dta')
```