

MSc. Business Information Technology

Master Thesis HS2021



Does Illegal Collusion Determine Stock Prices and Returns?

Matching of Refinitiv Financial Firm Data with Convicted Cartel Cases using Regular Expressions (RegEx)

Ordering Party: Zürcher Hochschule für Angewandte Wissenschaften, Departement
School of Management and Law, Institut für Wirtschaftsinformatik

Author: Fabian Kaewmulpet, matriculation number: 21568498

Supervisor: Dr. Andrea Günster

Co-supervisor: Nicole Bellert

Submission date: 26.05.2023

Management Summary

In the dynamic landscape of business and economics, understanding the factors that influence stock prices and returns is crucial for effective decision-making and strategic planning. This study delves into the intricate relationship between illegal collusion and stock prices, aiming to uncover whether such collusion has a significant impact on stock prices and returns, providing valuable insights and enabling informed decision-making in the face of potential market disruptions. It seeks to provide valuable insights into the interplay between illegal collusion and the financial outcomes of companies. The findings of this study have the potential to guide in understanding the risks associated with illegal collusion and their influence on stock prices and returns. The approach taken in this study was multi-faceted and involved the collection, preparation, and analysis of data from a range of sources. The artifact creation process involves analysing cartel information and selecting relevant data for further processing. Regular Expression has been utilized to prepare the data. It was then transferred to Orbis to obtain business identifiers and subsequently request a full financial export from Refinitiv. A range of statistical techniques have been used to analyse this data, including regression, correlation, and time series analysis. Regarding the impact of illegal collusion on stock price returns, two regression analyses were conducted. The findings indicated that the variables “pre_cartel”, “cartel”, and “post_cartel” did not have a statistically significant impact on stock price returns, since neither coefficient was statistically significant. These results suggest that illegal collusion does not significantly affect stock price returns based on the model employed. Based on the that, it can be concluded that illegal collusion does not yield an increase in stock price returns. This valuable insight suggests that participating in such activities is not worthwhile from this standpoint. While this analysis provides valuable information, it is crucial to acknowledge the limitations. The size and completeness of the available data were constrained, and the sample size was relatively small, limiting the statistical relevance and generalizability of the results. Further research is in general recommended to explore alternative approaches to determine the correlation between illegal collusion and stock prices, considering larger sample sizes and comprehensive databases. Despite the limitations, this study sheds light on the relationship between illegal collusion and stock price returns, emphasizing the need for further investigation and analysis. The study recommends further research to explore alternative approaches, larger sample sizes, and comprehensive databases to improve the statistical relevance and generalizability of the results. Due to non-stationarity in the used model for the stock price regression, another approach is recommended in further research. Based on the results, we can take some assurance that illegal collusion, as examined in this study, may not pose an immediate threat to stock price returns. However, it is still important to remain vigilant and periodically assess the risk of collusion in the industry or market segment relevant to the company.

Statement of truth

“I hereby declare that I have written this thesis independently, without the assistance of third parties and using only the sources indicated, and that I will not hand out copies of this thesis to third parties without the written consent of the course director.”

At the same time, all rights to the work are assigned to the “Zürcher Hochschule für Angewandte Wissenschaften” (ZHAW). The right to name the authorship remains unaffected.

Name/First Name Student (block capitals)

Kaewmulpet Fabian

Contents

| | |
|--|----|
| 1. Introduction | 1 |
| 1.1 Problem definition and field of application | 2 |
| 1.2 Objective(s)..... | 2 |
| 1.3 Research & procedural questions | 3 |
| 1.4 Delimitation | 4 |
| 2. Initial circumstances..... | 5 |
| 2.1 Preliminary work | 6 |
| 2.2 Cartel case information..... | 8 |
| 2.3 Bloomberg or Refinitiv | 10 |
| 2.4 Merging..... | 10 |
| 2.5 Relevance of this thesis..... | 11 |
| 3. Scientific Methods..... | 12 |
| 3.1 Design Science Research | 12 |
| 3.2 Design Theory..... | 14 |
| 3.3 Artifacts | 15 |
| 3.4 Regular Expression | 16 |
| 4. Artifact | 19 |
| 4.1 Artifact creation process | 19 |
| 4.2 Data filtration and preparation..... | 20 |
| 4.2.1 Regular Expression | 23 |
| 4.3 Receive Identifier..... | 31 |
| 4.3.1 Refinitiv Python API via OpenFIGL..... | 31 |
| 4.3.2 Orbis..... | 35 |
| 4.4 Receive RIC from ISIN | 41 |
| 4.5 Retrieve financial data and first interpretation | 45 |
| 5. Results..... | 55 |

| | | |
|-----|--|------|
| 5.1 | Fundamental data..... | 55 |
| 5.2 | Auto Regression Test for Residuals: Assessing Weak Stationarity..... | 56 |
| 5.3 | Unit Root Test for Stock Closing Price | 58 |
| 5.4 | Unit Root Test for Stock Closing Price Returns..... | 60 |
| 5.5 | Regression for Stock Price Returns including FixedEffects..... | 62 |
| 5.6 | Regression for Stock Price Returns including FixedEffects and trend..... | 63 |
| 6. | Conclusion..... | 66 |
| 6.1 | Impact of illegal collusion on stock price and stock price returns..... | 66 |
| 6.2 | Limitation..... | 67 |
| 6.3 | Discussion & recommendation | 68 |
| | Bibliography | I |
| | Appendix | III |
| | A. Illustration of the stock closing price returns for the stock “MAUP.PA” | III |
| | B. Panel Linear Regression for Stock price including Unit Root on residuals (Without fixed effects) | IV |
| | C. Regression for Stock price including fixed effects (Biased)..... | IV |
| | D. Regression for Stock price including fixed effects and the trend (Biased)..... | VI |
| | E. Panel Linear Regression for Stock Price Returns including Unit Root on residuals (Without fixed effects) (Biased) | VIII |
| | F. Regression with different parameter (Biased) | VIII |

Table of Figures

| | |
|---|----|
| Figure 1 Cartel data- example of the European Sugar Industry | 5 |
| Figure 2: Example of censored parties within the same cases- own illustration..... | 9 |
| Figure 3: Example of companies with similar name and address - own illustration..... | 9 |
| Figure 4: Relevance and rigor in design science research. Source Adapted from Hevner et al. (2004)..... | 13 |
| Figure 5: Layers of the artifact development process. Source The authors, based on Gill and Hevner (2011)..... | 15 |
| Figure 6: Potential structure of the artifact..... | 19 |
| Figure 7: Used data columns for the analysis..... | 22 |
| Figure 8: Example Orbis free text search | 36 |
| Figure 9: Short summary of Orbis for “James Budgett Sugars Limited” | 36 |
| Figure 10: Orbis mapping selection for bundled search..... | 37 |
| Figure 11: Orbis extract example | 39 |
| Figure 12: Refinitiv App Key Generator | 42 |
| Figure 13: development of stock closing price for MAUP.PA including case dates | 47 |
| Figure 14: Stock Closing Price and Returns development before, during and after the illegal collusion for MAERSKb.CO..... | 50 |
| Figure 15: Stock Closing Price and Returns development before, during and after the illegal collusion for 9107.T | 50 |
| Figure 16: Section of the constructed DataFrame | 56 |
| Figure 17: ADF-test results for the residuals | 57 |
| Figure 18: PanelOLS Estimation Summary for the alternative Unit Root Test for the stock closing price..... | 59 |
| Figure 19: PanelOLS Estimation Summary for the Unit Root Test for the stock closing price return..... | 61 |
| Figure 20: PanelOLS Estimation Summary for Stock Price Returns including FE | 63 |
| Figure 21: PanelOLS Estimation Summary for Stock Price Returns including EntityEffects and trend | 64 |

Table of Code-Snippets

| | |
|---|----|
| Code-Snippet 1: Example of RegEx matching beginning or end of a line..... | 17 |
| Code-Snippet 2: Example of RegEx replacement of bracket values..... | 18 |
| Code-Snippet 3: Cleansing of missing dates | 22 |
| Code-Snippet 4: import and preparation of data | 23 |
| Code-Snippet 5: Creation of function to check for special characters and whitespaces | 24 |
| Code-Snippet 6: Application of function and storage of results | 25 |
| Code-Snippet 7: Import of formatted cartel information | 26 |
| Code-Snippet 8: Category 1- Replacement of special characters using Regular Expression | 27 |
| Code-Snippet 9: Category 2- Regex to remove additional information..... | 28 |
| Code-Snippet 10: Category 3 and 4- Regex to remove country codes and company types at the beginning/ending | 29 |
| Code-Snippet 11: Category 5- Regular Expression to remove defined words..... | 30 |
| Code-Snippet 12: Category 6- Clearing of whitespaces..... | 30 |
| Code-Snippet 13: Batch export of the DataFrame..... | 30 |
| Code-Snippet 14: Creation of Variables for API script..... | 32 |
| Code-Snippet 15: OpenFIGI API request Loop | 33 |
| Code-Snippet 16: Import and merge of cartel data and Orbis-matched data | 40 |
| Code-Snippet 17: Creation of the fully merged DataFrame..... | 40 |
| Code-Snippet 18: Refinitiv Eikon API setup | 43 |
| Code-Snippet 19: For-Loop and export of RIC-DataFrame..... | 43 |
| Code-Snippet 20: Refinitiv gather financial data for MAUP.PA..... | 46 |
| Code-Snippet 21: Initial code for multi company search via the Refinitiv Eikon API.. | 48 |
| Code-Snippet 22: First part of the data gathering for the Smart Data Analytics..... | 51 |
| Code-Snippet 23: Second part of the data gathering for the Smart Data Analytics | 52 |
| Code-Snippet 24: Concatenate the gathered data into one DataFrame | 53 |

Table of Tables

| | |
|--|----|
| Table 1: Eight components of an Information Systems Design Theory. Source Adapted from Gregor et al. (2007)..... | 14 |
| Table 2: Example of SARL in Capital letters vs lower case | 18 |
| Table 3: Excerpt of “1. Sepcial_chars.xlsx” | 26 |
| Table 4: “all_companies.xlsx” table excerpt | 35 |
| Table 5: Matched companies based on country and city..... | 38 |
| Table 6 : Summary statistics for the used DataFrame | 55 |

1. Introduction

In a free market, where the price of goods and services are determined by supply and demand, the principal task of the government is to protect property rights. However, what should be left to the market and when and to what extent should governments act to ensure the welfare of their citizens are ongoing issues in economics. According to Graafland and Verbruggen (2022), a smaller and less interventionist government limits governmental decision making and preserves the freedom of economic agents to pursue their own trade-offs and preferences. This requires a strong foundation that includes property rights, contract security, legal certainty, and protection against aggression. If, however participating competitors fix prices, divide markets or rig tendering procedures, we define this as a business cartel that entails corporate and economic crime. Cartels give firms the advantage to minimize uncertainties and the risk of a competitive market, leading them to a potential benefit to their own company value as well as reduced expenses (Jaspers, 2017). According to Günster and van Dijk (2016), companies with European Commission convictions experience noticeably unfavourable stock market reactions. Their event study for a sample of 253 companies involved in 118 European antitrust cases throughout 1974 and 2004, displayed a negative stock price reaction for the companies. This result raises the question of whether the stock prices and returns of cartelized companies were in fact greater prior to the European Commission's conviction. The foundation of this Master Thesis is the provided information about cartels between 1964 and 2010. The information is displayed as an excel file with approximately 165 different cases of cartels including their date of decision, firm name, locations as well as details of the case. Further information will be gathered by the use of Bloomberg and Refinitiv which offer extremely broad and granular data and news on every stock, bond, and asset.

The remaining of this chapter will offer a brief introduction into this Master Thesis' subject by defining the problem, the objectives, the research questions as well as this Thesis delimitation. The second chapter provides the reader with the initial circumstances. In chapter 3 the scientific methods used by the author are described. The fourth chapter describes the creation process and usage of the artifact, which results will be analysed in chapter 5. Chapter 6 will conclude the whole topic and offer a discussion and recommendation.

1.1 Problem definition and field of application

Illegal cartels upset the balance of a free market creating an unfair situation between consumers and distributors. Although some of these cartels will be exposed and result in legal action, there will also be some cases that will remain undetected. According to Combe, within the samples of all cartels convicted by the European Union from 1969 to 2007, the likelihood of being caught in a particular year was between 12.9% and 13.2%. This serves as the top limit of the probability of detection globally and the likelihood of detection in any given year has been therefore at most 13.2% (Combe et al., 2007). Although, the introduction of leniency programmes quenches market competitors from participating in cartels or even reporting them. Between 2002 and 2005 for example, the European Commission received 167 applications from undertakings including 87 for full leniency and 80 for the reduction of fines. Based on information provided by cartel members, the vast majority of EU cartel cases are initiated (Bruneckienė et al., 2015). Market collusion, bid-rigging, and other cartel activities will continue to happen despite the fact that progress is being made and legal observers are improving the fairness for market participants. It remains to be seen whether these will have a favourable effect on stock prices and returns. This Master Thesis aims to offer statistical support by matching the cartel information with stock prices and returns on equities relevant data for both the economic department as well as the information technology department.

1.2 Objective(s)

The main objective is to estimate and test the impact of a firm's involvement in a cartel process on its stock price and returns. To achieve this objective, numerous subgoals have to be accomplished. The first subgoal is to prepare the given information about the firm cartels in a way to make them useable for a data import in Jupyter notebook. The second subgoal will be achieved once a data source for the stock prices and returns has been chosen. Possible data sources to gather that information from are Bloomberg and Refinitiv. Soon after, the firm cartel data has to be compared and joined with the stock data from one of the proposed data sources using regular expressions. Once an algorithm has been created and the two data sets can be compared and joined, the third subgoal has been taken care of as well. After the combined information is available, the data will be analysed to display what impact the involvement in an illegal cartel on the stock price and

stock price returns has. Based on those objectives a neutral statement should conclude at the end of this Master Thesis.

1.3 Research & procedural questions

The purpose of this document is to answer the following research question:

Does Illegal Collusion Determine Stock Prices and Stock Price Returns?

The leading question concerns whether stock prices and returns of cartelized firms are indeed higher prior to and how they changed after the conviction by the European Commission.

In addition, the procedural questions will guide answering the leading research question.

Which data source should be used for the stock prices and stock price returns?

This question will be mandatory to answer, since this will provide the data which will be matched with the cartel information. The decision will be made by content and compatibility of the given data. In case multiple data sources contain valuable information, it is possible that various data sources will be chosen if joinable.

How can the cartel data be matched with the stock data?

To run an analysis over both information and check how the stock prices and returns have changed before, during and the after the firms have been convicted, both data sources have to be combined and matched, a technical challenge will be to match the firm names from the cartel data with the firm names of e.g., Bloomberg or Refinitiv.

How should the matched data be analysed?

By answering this question, the focus will be on what information is relevant in which form. Depending on the provided information the answer to this question will determine the final result.

How can the relevance of this document be displayed?

The relevance of this document is given if it contributes to an open topic or answers a research question that has not been asked yet. If the research question has been answered successfully it will add value to the professional relevance. By answering the procedural questions, the scientific relevance (rigor) will be contributed.

1.4 Delimitation

This Master Thesis will set the expectation horizon with its delimitation and will further expand or shrink it within the chapter 6.2 Limitation. It's very likely that the Regular Expression algorithm is not capable of finding every possible match. However, it aspires to create an algorithm that discover as many matches as possible and provide a reasonable number of statistical significances. The created artefact shall function to contribute its user with sufficient data to be relevant in a practical frame. The intention of this Master Thesis is mainly to create an artefact and analyse the resulting information. The created algorithm will be specifically built based on the given data, therefore it is not part of this Master Thesis to have a dynamic script that is applicable on all cartel information.

2. Initial circumstances

A sufficient number of data must be presented in order to make a meaningful statement on how illegal collusion affects stock prices and stock price returns. As briefly described in the previous chapter, a dataset of 165 different cartel cases is provided in an excel file. The data is structured and contains 78 columns as well as over 1,700 rows. The data needs to be reviewed and an analysis has to be done on which information is relevant for this Thesis. If the relevant information has been chosen, it is necessary to check if the information is also correctly formatted to be used for further procedures. The provided information contains knowledge about the nature of the parties involved in a case along with the contributing party names. As an example, we extract the cartel case 26918 concerning the “EUROPEAN SUGAR INDUSTRY” as the nature of the parties involved and the companies “Pfeifer & Langen”, “NV Centrale Suiker Maatschappij NV” and the “Générale Sucrière SA” among others as displayed in Figure 1.

| Case # | Date Decision | | | Nature of the Parties Involved | Party Name(s) |
|-----------|---------------|-------|------|--------------------------------|--|
| | Day | Month | Year | | |
| 154 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | NV Centrale Suiker Maatschappij NV |
| 155 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Coöperatieve Vereniging Suiker Unie UA |
| 156 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Westdeutsche Zuckervertriebsgesellschaft mbH & Co KG |
| 157 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Pfeifer & Langen |
| 158 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Südzucker Verkaufs-GmbH |
| 159 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Süddeutsche Zucker AG |
| 160 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Zuckerfabrik Franken GmbH |
| 161 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Sucre-Union SA |
| 162 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Société des Raffineries et Sucretries Say |
| 163 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Société F. Béghin SA |
| 164 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Générale Sucrière SA |
| 165 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Société Nouvelle de Raffinerie Lebaudy-Sommier SA |
| 166 26918 | 2 | 1 | 1973 | EUROPEAN SUGAR INDUSTRY | Société Nouvelle de Raffinerie Lebaudy-Sommier SA |

Figure 1 Cartel data- example of the European Sugar Industry

A further description of the cartel data which serves as the basis for this thesis, can be found in “2.2 Cartel case information”. To gather information about the stock prices and return on equities for the different firms that were convicted on participating in an illegal business cartel, Bloomberg or Refinitiv will be used. Both platforms can be used to receive market data such as; stock and bond prices for both live and historical events, fundamental company data including balance sheets, profit as well as loss statements but also environmental, social, and governance data. Both data sources will be considered and, based on an analysis one or even both will be used in this thesis.

2.1 Preliminary work

The foundation of this Master Thesis is the result and related data of an analysis of the European Antitrust Policy named “European Antitrust Policy 1957–2004: An Analysis of Commission Decisions”, which has been published on the 6th of February 2010. The paper covers a complete overview and statistical analysis of 538 formal commission decisions since the foundation of the European antitrust law enforcement in the treaty of Rome from 1957 up to and including 2004. The information used in this Master Thesis goes beyond the data provided in the paper and furthermore includes knowledge beyond 2004 and up to year of 2010. This chapter refers to the mentioned paper and will conduct a summarization of its content, since it is required to equip the reader with the necessary context.

According to the paper a visible and much debated part of the activities of the European Commission is the European competition policy. Therefore, the authors surveyed the implementation of the antitrust law in the European Community and characterized more than forty years of European competition policy enforcement. For each of these European Commission antitrust cases, data was gathered on variables such as the length of the decision, the duration of the investigation, the nationality of the firms, the nature of the alleged offence, the level of fines imposed, the Commissioner who signed the decision, and whether the case was appealed. The executed survey is based upon an extensive data set which has been consistently updated with publicly available information on the decisions of the Directorate General for Competition (called DG Comp in the paper and will be also called this way hereafter) in their publication in the Official Journal of the European Communities. The analyses do not include European merger and State aid decisions, which number in the hundreds, nor decisions made under the previous European Coal and Steel Community Treaty. The European Coal and Steel Community was established by the Treaty of Paris in 1951, which has been succeeded however by the Treaty of Rome in 1957 and laid the foundation of European competition policy. One general objective of the Treaty of Rome identifies the achievement of “a system ensuring that competition in the internal market is not distorted”. The DG Comp prepares decisions in the areas of antitrust, mergers, and state aid and investigates several hundred cases each year. The European Commission was given the authority to initiate investigations, make decisions, and impose remedies and sanctions with the so called “Regulation 17” in 1962. Based on the Regulation 17, investigations could translate into on of three possible

decisions. A negative clearance when the agreement or practice is in accordance with existing Community law. An exemption is only granted temporarily and under specific conditions and the last decision is an infringement decision. There are three types of report routes under Regulation 17 by which cases enter DG Comp. First, with the introduced notification requirement applying for firms to inform about existing or planned agreements and concerted practices (abolished with the introduction of Regulation 1/2003). Second, if an individual or business files a complaint about a specific act. Third if the EU commission decides to investigate on its own. In 1996, a fourth report route was introduced in the form of a leniency notice. The Paper furthermore analysed the hypothesis of independence between this decision type and the report route, which has been rejected at a one per cent significance level. Moreover, the same analyses have been conducted between the commission decisions and the economic conduct, between the commission decisions and the OECD sector and the between the commission decisions and the nationality of firms. The survey allowed insight in the average durations of a decision for each year as well. While early cases took on average approximately 110 months between 1963 and 1967 the number of months of investigation dropped to an average of about thirty months per decision opened after 1975. The backlog of cases at the Directorate-General for Competition (DG Comp) was reduced in the 1980s, and it continued to fall rapidly until 1992. The longest inquiry length for all Commission decisions was 289 months in the instance of the Association Belge des Banques in 1986, but this can be attributed to the backlog effect. It is also stated that decision documents have grown in length over time, with an increase in the number of recitals conveying information about the scope of investigation and complexity of the case. Due to the need for precision in factual description and economic and legal argumentation, as well as the implementation of fining guidelines in 1998 and the use of leniency applications for evidence, the length of decision documents may have risen. In general, the paper provides evidence of antitrust enforcement variations across sectors and countries. Non-European firms have fewer infringements, cheaper fines, and lower appeal rates, demonstrating that antitrust regulation does not favor non-European competitors. Factors such as the amount imposed, the firm's nationality, economic conduct, and whether other firms appealed in the same case all influence the likelihood of an appeal, raising challenges for future research. While enforcement has evolved throughout time, European competition law enforcement has developed, with rulings focused on infringements, being more detailed and made faster, and fines being greater. However, there are certain concerns, such as the

efficiency of the leniency program and the rising appeals rate, which point to probable regulatory failure (Carree et al., 2010). This Master Thesis endeavours to analyse the impact of stock prices and stock price returns of companies that have been convicted of certain offenses, utilizing data and information from the paper mentioned above. The analysis focuses on examining the financial data of these companies over the conviction period, with the aim of comprehending the consequences of the conviction on their financial performance. By scrutinizing various financial metrics and indicators, this analysis seeks to provide insights into the potential effects of the accusation/conviction on the stock prices and stock price returns of the impacted companies. The findings from this analysis may shed light on the implications of the conviction on the financial health and stability of these companies, and contribute to the existing knowledge on the relationship between legal convictions and financial performance.

2.2 Cartel case information

The foundation of this Master Thesis is an excel file that contains cartel data for the period from 1964 to 2010, the table is structured that each row displays a firm and:

- The cartel case number, the company has been involved in
- The decision date, when the case has been declared as a cartel
- The nature of the companies involved
- The name of the participating party
- The start and end date of the cartel
- The company address including street, number, PO-Box, postcode, city and country
- The information if the company had/has a history with the EC
- The information if the company is a repeat offender
- The OECD Sector (e.g., Agriculture, manufacturing, transport etc.)
- The traded/manufactured products
- The number of parties involved
- The information if the main country of the company is EU/non-EU
- The nature of the cartel case (complaint, commission own initiative, etc.)
- The information if a leniency occurred

Although the file does contain more information, they are not relevant to this thesis. Multiple rows are grouped with the same case-number. All parties involved in the given case have been accused or convicted of participating in illegal collusions. Figure 2 as an example displays the cases 25107, 324 and 595 as well as some of the participating parties. The party names have been made illegible due to confidentiality.

| Case # | Date Decision | | | Nature of the Parties Involved | | Party Name(s) | Start | | | Text | End | | | Text |
|--------|---------------|-------|------|--------------------------------|---|---------------|-------|------|------|---------------------------------------|-------|------|------|---------------------------------|
| | Day | Month | Year | Parties | Day | | Month | Year | Day | | Month | Year | | |
| 25 | 25107 | 29 | 12 | 1970 | Carreaux Ceramiques (Interest Grouping of Gem | [Redacted] | 19 | 12 | 1958 | Der Beschluß der Interessengemeinsch | 29 | 12 | 1970 | Die Voraussetzungen für die Anv |
| 26 | 25107 | 29 | 12 | 1970 | Carreaux Ceramiques (Interest Grouping of Gem | [Redacted] | 19 | 12 | 1958 | Der Beschluß der Interessengemeinsch | 29 | 12 | 1970 | Die Voraussetzungen für die Anv |
| 27 | 25107 | 29 | 12 | 1970 | Carreaux Ceramiques (Interest Grouping of Gem | [Redacted] | 19 | 12 | 1958 | Der Beschluß der Interessengemeinsch | 29 | 12 | 1970 | Die Voraussetzungen für die Anv |
| 28 | 25107 | 29 | 12 | 1970 | Carreaux Ceramiques (Interest Grouping of Gem | [Redacted] | 19 | 12 | 1958 | Der Beschluß der Interessengemeinsch | 29 | 12 | 1970 | Die Voraussetzungen für die Anv |
| 29 | 25107 | 29 | 12 | 1970 | Carreaux Ceramiques (Interest Grouping of Gem | [Redacted] | 19 | 12 | 1958 | Der Beschluß der Interessengemeinsch | 29 | 12 | 1970 | Die Voraussetzungen für die Anv |
| 30 | 25107 | 29 | 12 | 1970 | Carreaux Ceramiques (Interest Grouping of Gem | [Redacted] | 19 | 12 | 1958 | Der Beschluß der Interessengemeinsch | 29 | 12 | 1970 | Die Voraussetzungen für die Anv |
| 31 | 324 | 16 | 12 | 1971 | Vereeniging van Cementhandelaren (VCH) | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 32 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 33 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 34 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 35 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 36 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 37 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 38 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 39 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 40 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 41 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 42 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 43 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |
| 44 | 595 | 23 | 12 | 1971 | Nederlandse Cement-Handelsmaatschappij (NC | [Redacted] | | | 1955 | Die "Algemene Bepalingen en Prijsvoor | 16 | 12 | 1971 | Die "Algemene Bepalingen en Pri |

Figure 2: Example of censored parties within the same cases- own illustration

The start date and the end date of each given case will also mark specific milestones on a timeline in which the stock prices will be analyse. Those two dates combined with the decision date will provide enough information to describe how the stock price changed from before the cartel-case formed, during the illegal collusions as well as after the cartel ended and what effect the decision date had on the stock price. Within the same cases firms with similar names are often included. The names only differ slightly from each other and often they have the same address but there are multiple occasions as well where this is not the case. The address will support finding parties that had been registered with multiple company names under the same address, this will be relevant when matching the cartel data with Bloomberg or Refinitiv. A good example for this case is case 37857 displayed in Figure 3 which includes three different “Akzo Nobel” companies with different names and two of them having the same address.

| Case # | Date Decision | | | Nature of the Parties Involved | | Adresse | | | | | |
|--------|---------------|-------|------|--------------------------------|---------------------------------------|-------------------------|--------|--------|----------|------------|-------------|
| | Day | Month | Year | Parties | Party Name(s) | Street | Number | PO-Box | Postcode | City | Country |
| 37857 | 10 | 12 | 2003 | Organic peroxides | Akzo Nobel Polymer Chemicals BV | Stationsplein | 4 | | 3813 | Amersfoort | Niederlande |
| 37857 | 10 | 12 | 2003 | Organic peroxides | Akzo Nobel NV | Velperweg | 76 | | 6800 | Arnhem | Niederlande |
| 37857 | 10 | 12 | 2003 | Organic peroxides | Akzo Nobel Chemicals International BV | Stationsplein | 4 | | 3813 | Amersfoort | Niederlande |
| 37857 | 10 | 12 | 2003 | Organic peroxides | [Redacted] | Cours Michelet | 4 | | 92800 | Paris | Frankreich |
| 37857 | 10 | 12 | 2003 | Organic peroxides | [Redacted] | Dr. Gustav-Adolph-Stras | 3 | | 82049 | Pullach | Deutschland |
| 37857 | 10 | 12 | 2003 | Organic peroxides | [Redacted] | Brook Street | 42 | | W1K 5DB | London | UK |

Figure 3: Example of companies with similar name and address - own illustration

2.3 Bloomberg or Refinitiv

By connecting them to a dynamic network of information, people, and ideas, Bloomberg sees itself as the world's leading provider of business and financial news and information. The Company's strength is to deliver data, news and analytics through innovative technology in a quick and accurate way (*Bloomberg L.P. | About, Careers, Products, Contacts*, n.d.). Refinitiv is an LSEG (London Stock Exchange Group) business and one of the world's largest providers of financial market data and infrastructure. They provide information, insights, and technology to their customers for investing, trading, and risk decisions (*Refinitiv- About Us*, n.d.). Both companies provide a useable platform to access crucial information for this Master Thesis. To receive both stock prices and stock price returns from the cartel data by the use of the firm names, both Bloomberg and Refinitiv data has been compared. Both services have been tested on different criteria like accessibility, data-quality, API's and implementation possibilities. The outcome considers that Refinitiv offers more flexibility and useability with the easily accessible Eikon library, which can be directly integrated in a python Jupyter Notebook. It is therefore chosen as the used tool for the reception of business identifier.

2.4 Merging

A crucial part for the data analysis is the already mentioned merging between the firms from the cartel data with the financial information from Refinitiv. Since each company has a distinct spelling on Refinitiv, where the name of the cartel information might differ, the data has to be either changed before the match with Refinitiv happens or the use of tools from the financial data platforms have to come in play to retrieve the correct data for the right companies. Refinitiv only supports a search over the unique RIC (Reuters Instrument Code), which is a ticker-like code used by Thomson Reuters to identify financial instruments and indices (*MIFID II MASTER RIC USER GUIDE*, 2018). As an example, the RIC for the company Apple Inc. is "AAPL".

Based on the given data from the cartel information file there will be three main challenges. The first one concerns the companies that are or were not listed at the stock market. For those firms there won't be any information about how the stock price or the stock price returns changed since those values do not exist for the unlisted companies. The international diversification of the companies leads to the fact that the types of companies are wide. While the business type "PLC (public limited company)" is

indicating that the company was listed at the stock market so does the “NV (Naamloze vennootschap)” for the Netherlands and the “AG (Aktiengesellschaft)” for the German speaking areas. Moreover, limited company types like the “Ltd. (limited)” or the “CV (Commanditaire Vennootschap)” from the Netherlands can indicate that the firm was most likely not listed at the stock market. Because of this first challenge the companies will be categorized in “stock market listed, unlisted and unknown” distributed to better estimate the chances of matching a company from the cartel information with the associated Refinitiv information. The second challenge lies within the companies that should be listed but no RIC nor any other unique identifier is present due to the age and possibly non-existence of the company or other circumstances. Those companies have to be extracted and most likely manually checked. The third and last challenge concerns the firm names in the provided cartel information file. The names have to be automatically cleaned by removing special characters and translating (e.g.) Cyrillic or Hanzi alphabets into Latin. Once those challenges have been solved, the data can be merged with the financial information from Refinitiv.

2.5 Relevance of this thesis

The goal of this Master Thesis is to develop an artefact, which allows the user to gain more knowledge and insight on the impact of convicted cartels. Although the cartel convictions have a direct and practical impact on the companies, a statistical and number proved approach for the data between 1964 and 2010 is yet to be done. By analysing the gathered data for firms that have been convicted or accused of participating in illegal collisional activities, this thesis will provide empirical information on how the stock-prices and returns changed throughout the different phases of EU cartel convictions starting by the investigations, over to the proclamation and the changes after the convictions.

3. Scientific Methods

One can define research work as a systematic investigation whose central goal is typically the development or refinement of theories and, in some cases the solution to problems. The need for research work arises from the realization that adequate and systemized information to answer some given problem is lacking. To fulfil this central goal the researcher gathers knowledge in the area of the given topic and assembles this logically. The way in which this knowledge is constructed is a perspective displayed by the “Scientific Method” (Dresch et al., 2014). Studies that are conducted using the paradigm of traditional sciences, such as natural and social sciences, concentrate on describing, explaining, exploring, or forecasting events and how they relate to one another. The traditional sciences can have constraints to examine the design, creation, or construction of a new artifact, something that doesn't yet exist, or to carry out research that focuses on problem solutions. Therefore traditional sciences are critiqued based on these presumptions, and the usage of the Design Science is suggested as a new epistemological paradigm for doing research (Dresch et al., 2014). Since this Master Thesis aims to create an artifact to solve the described research question, the scientific method can be classified as Design Science.

3.1 Design Science Research

According to Dresch, Design Science is the epistemological basis for the study of what is artificial, and Design Science Research is therefore a method that establishes and operationalizes research when the desired goal is an artifact or a recommendation. It is possible to conduct design science-based research in both an academic setting (Rigor) and an organizational setting (Relevance). Figure 4 models both contexts and describes how they relate to Design Science Research. The “Environment” in this figure refers to the setting in which the issue is being observed, or where the researcher obtains the occurrence of interest. In this setting, the item is operational. There are people, the organization, and its technology in this setting. The “Environment” for this thesis is the market and its participants. The “Relevance” displays how stock prices changed before and after a conviction for different companies, which in our setting displays the organizational part. Design Science Research can help with the creation and production of artifacts and improve the body of already existing information based on the challenges and “Organizational Needs” that the researcher is interested in observing. These artifacts

subsequently undergo evaluations, refinements, and justifications of their importance. The existing “Knowledge Base” needs to be consulted and used to assist these developments, construction, justification, and assessment activities. The academic community accepts the foundations and procedures that make up this body of knowledge. This knowledge base is composed of well-established foundations and methods that are recognized by the academic community (Hevner et al., 2004). The Knowledge base will be used to gather existing methods, information, and algorithms to already solved similar problems. This Master Thesis will therefore make use of already existing methodologies for data analysis and validation criteria. The “Rigor” related “Foundations” will provide existing frameworks, constructs and instruments that help building the final artifact. By the creation of an artifact to solve organizational needs while using applicable knowledge including foundations and methods from the traditional research component, this Master Thesis is going to make use of Design Science Research. The final artifact and hence the product of our Design Science Research Process will be a construct which can be used to evaluate stock price data and company-based information for different firms before and after a conviction.

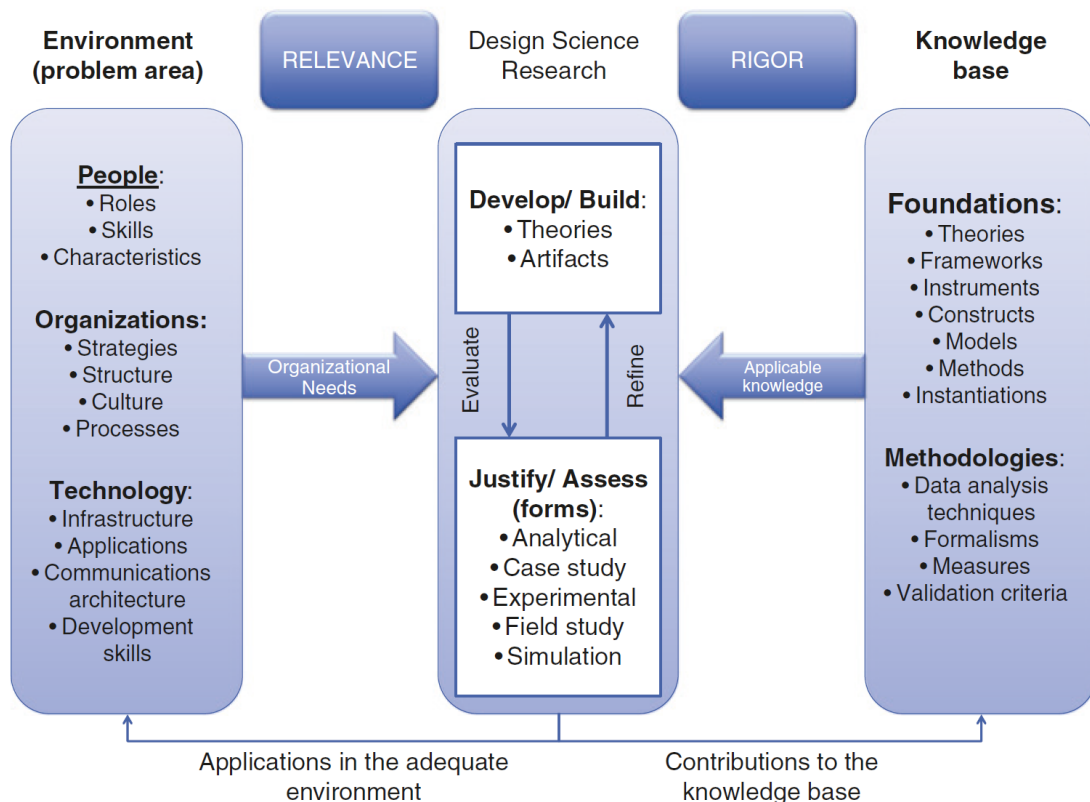


Figure 4: Relevance and rigor in design science research. Source Adapted from Hevner et al. (2004)

3.2 Design Theory

In addition to producing an artifact, Design Science Research also aims to create knowledge about it. When this knowledge is complete and developed, it can be formalized into a design theory to make it explicit and organized. It can be applied to new situations and expanded in a process of cumulative knowledge development (Johannesson & Perjons, 2014). Gregor and Jones proposed an anatomy for design theories in the information system area separating the design theories in the eight components depicted in Table 1.

Table 1: Eight components of an Information Systems Design Theory. Source Adapted from Gregor et al. (2007)

| Component | Description | Thesis Application |
|---|---|---|
| Core components | | |
| 1) Purpose and scope (The causa finalis) | “What the system is for”, the set of meta-requirements or goals that specifies the type of artifact to which the theory applies and in conjunction also defines the scope, or boundaries, of the theory | The artifact will merge and analyse data based on the existing cartel conclusions together with Refinitiv information |
| 2) Constructs (The causa materialis) | Representations of the entities of interest in the theory | All main and subcategories of each entity will be displayed and described |
| 3) Principle of form and function (The causa formalis) | The abstract “blueprint” or architecture that describes an IS artifact, either product or method/intervention | By modelling the gross structure of the artifact in none-specific categories a blueprint will be created |
| 4) Artifact mutability | The changes in state of the artifact anticipated in the theory, that is, what degree of artifact change is encompassed by the theory | During the whole artifact process creation, the artifact will undergo different states of information transformation and transportation |
| 5) Testable propositions | Truth statements about the design theory | Once the artifact has been constructed it will be tested to provide a truth statement |
| 6) Justificatory knowledge | The underlying knowledge or theory from the natural or social or design sciences that gives a basis and explanation for the design (kernel theories) | Used knowledge and theory in any form will be displayed and explained in the thesis |
| Additional components | | |
| 7) Principles of implementation (The causa efficiens) | A description of processes for implementing the theory (either product or method) in specific contexts. | By describing the process of implementation, we provide readers with enough knowledge to rebuild the artifact and use it |
| 8) Expository instantiation | A physical implementation of the artifact that can assist in representing the theory both as an expository device and for purposes of testing. | The artifact will be used in the given context to represent the underlying theory |

3.3 Artifacts

It is possible to think of artifacts as objects that are man-made, or artificial. Despite the fact that they are manufactured and therefore constructed according to the principles of design science, artifacts are nevertheless subject to natural laws which are governed by conventional science. In light of this, it may be claimed that artifacts are man-made objects that can be classified according to their purposes, ambitions, and adaptability (Dresch et al., 2014). Gill and Hevner however define artifacts as “a symbolic representation or a physical instantiation of design concepts”. They describe the design process as a construction of several layers which are strongly related to the characteristics and properties of the artifacts that are being developed as shown in Figure 5.

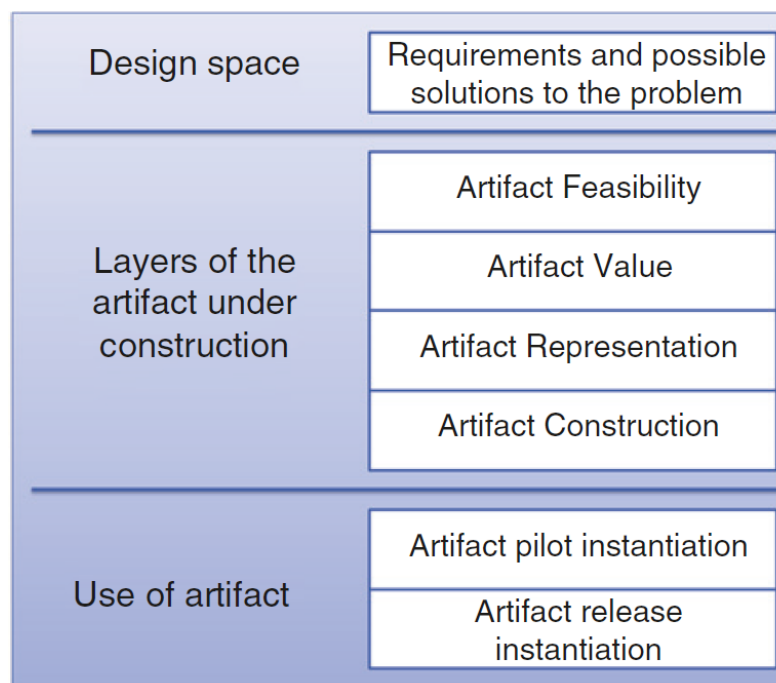


Figure 5: Layers of the artifact development process. Source The authors, based on Gill and Hevner (2011)

The first layer called the design space represents the set of possible solutions to the problem in which the possible artifacts to be developed exist, as well as the requirements for their proper functioning. In this layer, the researcher can examine what exists and what does not yet exist relative to the problem that is being studied, as well as regarding the desired artifact to be developed. After a possible artifact has been chosen as a solution, the development of the artifact itself comes in play in the second layer. The “Artifact feasibility” aims to ensure that what is being proposed can indeed be implemented, while

the “Artifact Value” represents the benefits of this artifact to its users and why it will be developed instead of any other. The “Artifact Representation” determines the most appropriate format to communicate the artifacts concepts. The last point in this layer is the “Artifact Construction” itself. It guides its users in its further implementation in the real context. The last layer is the “Use of artifact” layer which prepares the artifact for its implementation and use in real environment. This happens by piloting instantiations and afterwards releasing those instantiations. This thesis will make use of this concept by trying to create the artifact based on the described layers. They are going to be used as guidance to support the artifact creation process.

3.4 Regular Expression

Since Regular Expression will be a major part for the data cleaning and preparation, this chapter dedicates to introduce the reader to the topic, provide a definition, and explain basic Regular Expression skills, rules, and appliances. A regular expression, according to Goyvaerts and Levithan, is a type of text pattern that may be used with many modern applications and programming languages. The phrase regular expression is derived from mathematics and computer science theory, and it refers to a property of mathematical expressions known as regularity. A deterministic finite automaton can be used to implement such an expression in software (DFA). A DFA is a finite state machine with no backtracking. The text patterns utilized by the first grep tools were mathematically defined as regular expressions. Despite the term, regular expressions are not regular expressions in the mathematical sense. They are realized using a nondeterministic finite automaton (NFA). Regular Expressions simplify many programming and text processing tasks that allows the user to fulfil them in a way, which wouldn't be feasible at all without the regular expressions, since it would require dozens if not hundreds of lines of procedural code to extract the necessary information (e.g., extract all email addresses from a document). With the usage of regular expressions the same or even better results can be achieved with only a few lines of code, or even only one line (Goyvaerts & Levithan, 2009). This functionality allows the artifact, created in this Master Thesis, to remove unnecessary information from the company names as well as identifying special characters from firm names for instance “è, ñ, ë” or “â”. Those characters might be not recognized by the database we are retrieving the company information from. However, it also permits us to erase certain details about the company's ownership structure changing

the example from “Société Artésienne de Vinyl SA” to “Societe artesienne de vinyl”. Finding a portion of text that is characterized and matched by a regular expression is what pattern matching is all about. The regular expression engine is the underlying programming that searches the text. Stubblebine describes two rules as follows: First, the earliest or leftmost match wins, which means that regular expressions are applied to the input beginning with the first character and progressing to the last, and the engine delivers a match as soon as it finds one. The second rule states that the standard quantifiers are greedy and they attempt to match as many times as possible. The quantifiers settle for less than the maximum only if this is necessary for the success of the match. This process of giving up characters and trying less-greedy matches is the before mentioned “backtracking” (Stubblebine, 2007). The search-and-replace function, which accepts a subject string, a regular expression, and a replacement string as input, is a frequent task for regular expressions. The result is the subject string with all regular expression matches replaced with the replacement text. This replacement text is not a regular expression, but it is possible to create dynamic replacement texts using specific syntax (Goyvaerts & Levithan, 2009). The search-and-replace function will be used to exchange special characters with common characters but also to replace company structure from abbreviations or notations with an empty string to delete them.

Further in this chapter, only those Regular Expression skills that are used in this Master Thesis will be further described to have the reader understand how the algorithms work. To find a match at the start and/or at the end of a line, Regular Expression is using different expression tokens such as `<^>`, `<$>`, `<\A>`, `<\Z>`, and `<\z>`. Those tokens are called “anchors” and they are not matching the characters inside but instead they match at certain positions, effectively anchoring the regular expression match at those positions. The `<^>` anchor will be used to check if a certain string matches at the beginning of a company name e.g., “NV Société Rateau”. On the contrary the `<$>` anchor is used to match only at the end of a line indicated by line breaks for instance in “Heintz van Landewyck SARL”. The appliance of both rules is demonstrated in Code-Snippet 1.

```
#RegEx to remove different company types at the beginning or Ending of a line
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'^[nN][vV] ', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'[sS][aA][rR][lL]$', '', regex=True)
```

Code-Snippet 1: Example of RegEx matching beginning or end of a line

Additionally, Code-Snippet 1 is also displaying the application of another rule that covers the usage of upper- and lower-case letters. Inside the cartel information file, it is visible that all kinds of cases should be covered. An example of the application of this method is shown for index 336 in Table 2. In this case all letters of the company type are written in capital letters. However, for the Index 1580, displayed in the same table the company type starts with a capital “S” and the rest of the string is written in lower case.

Table 2: Example of SARL in Capital letters vs lower case

| | | Date Decision | | | Nature of the Parties Involved | |
|------|--------|---------------|-------|------|--------------------------------|---------------------------|
| | Case # | Day | Month | Year | Parties | Party Name(s) |
| 336 | 28852 | 20 | 7 | 1978 | FEDETAB | Heintz van Landewyck SARL |
| 1578 | 39125 | 12 | 11 | 2008 | Carglass | Splintex France Sarl |

Another Regular Expression function that is used during the data preparation is displayed in Code-Snippet 2. In regular expressions, the logical “OR” operation is denoted using the vertical bar (|). The vertical bar acts as a logical OR operator, allowing the user to specify alternative patterns that can match different substrings. For example, consider the following regular expression pattern: pattern1|pattern2. This pattern would correspond to either pattern1 or pattern2. The regular expression will be regarded a match if either pattern1 or pattern2 is present in the input text (Stubblebine, 2007).

```
#Remove country codes in brackets from names
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'\[.*?\]|\(.*?\)', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'^\w\s', '', regex=True)
```

Code-Snippet 2: Example of RegEx replacement of bracket values

This is used in the first expression “[.*?]|(.*?)” to divide two matchings and match both patterns separately. The first regular expression matches a sequence beginning with an opening square bracket “[”, followed by zero or more characters of any type (represented by .*?), in a non-greedy or lazy manner (marked by “?”), until it encounters a closing square bracket “]”. To put it another way, it catches any text included within square brackets, including the brackets themselves. The second part of the expression acts in a similar way. This section of the regular expression matches a sequence that begins with an opening parenthesis “(”, followed by zero or more characters of any type (expressed by “.*?”), in a non-greedy or lazy manner (marked by “?”), until it reaches a closing parenthesis “)”. There the function captures any text enclosed within the parentheses including the parentheses themselves and replace them with the empty string “”.

4. Artifact

This chapter will describe the artifact creation process including different approaches as well as the whole artifact creation process itself. This chapter will therefore include the planned artifact creation process, the data preparation and filtration, the used Regular expressions, and the step of receiving the Identifier while using two different approaches. The first approach is describing a python script that automatically calls an API delivering the company names and retrieving the company identifier if available in return. The second approach uses Orbis, which is a database that stores information for companies (such as the company identifier).

4.1 Artifact creation process

Figure 6 will guide the explanation of how the artifact will be potentially set up. The first step of the artefact creation process is to analyse the cartel information. As described at the beginning the cartel information is present in an excel file containing 23 columns, each describing another detail for a company that was accused of participating in illegal activities (for instance formation or participating in/of a cartel or price fixing).

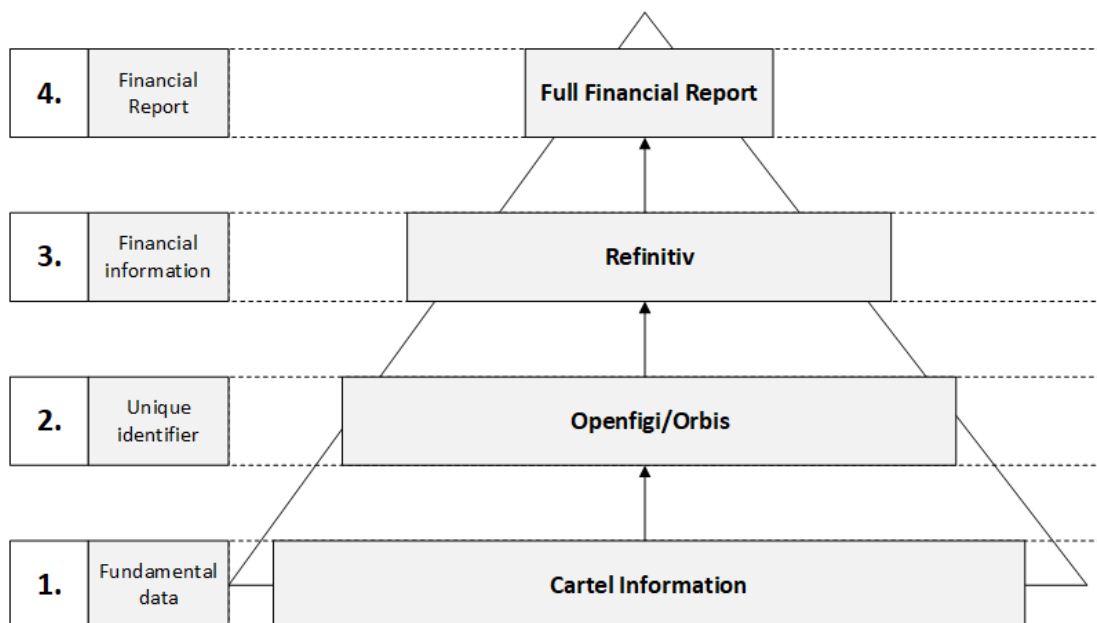


Figure 6: Potential structure of the artifact

Within the information it has to be decided which information is relevant for this thesis and therefore can be used for the whole process or even parts of it. Once the data has been reduced to the crucial information only, we will be using Regular Expression to format

the company names. By applying various pattern-finding expressions we will replace special character letters (e.g., ñ, ý, ë and ô) with common letters and remove company structures from the name to enable more freedom for the step of receiving the firm identifier for the formatted company names. For this second step we will consider 2 processes, where one displays the usage of an API called “openfigi” which is called within our python script, sending the formatted company names, and receiving the existing business identifiers from the API. The second approach requires a more manual way. This approach uses the “Bureau van Dijk” Database called “Orbis”. Orbis has the option to upload datasets and retrieve certain company information from the database. Based on which approach result in better data, the selected step will be established in the whole process. With the help of the company identifier the third step is then to feed them to Refinitiv with the additional available data and retrieve a full financial export for the requested companies. Refinitiv will however only provide information about companies that are stock market listed. Once this information has been prepared and combined with the existing information, the financial report is available and the artifact creation process is done, the next step is then to analyse and evaluate the given data.

4.2 Data filtration and preparation

Data filtration and preparation are critical procedures in scientific research because they ensure the accuracy, dependability, and validity of data used for analysis and interpretation. One of the primary reasons for data preparation and filtration is to reduce biases and errors that can occur during data collection. Data preparation and filtration techniques, such as data cleaning, normalization, and imputation, can help mitigate these biases and ensure that the data used for analysis is as accurate and representative as possible. Furthermore, data preparation and filtration are critical steps in enhancing data quality. Data cleaning strategies including deleting duplicates, fixing errors, and filling in missing values can assist in improving data quality by ensuring that it is complete, consistent, and dependable. It is essential for obtaining solid findings and precise judgments from data. They provide a strong foundation for scientific research by guaranteeing that the data used for analysis is accurate, reliable, and representative (Provost & Fawcett, 2013). Inside the given data, we analyse each column or group of columns for those that belong together and label them as relevant for this thesis, along with a justification for why it will be valuable. At the end of the chapter an overview will

be provided that includes all selected columns and serves as the filtered Data. Each line in the dataset is marked with an index displayed as a running number starting at 1. This column will be kept and used to refer to certain values in the dataset. The next column is called “Case#” and marks multiple lines/companies that have been accused or convicted for working in the same party. This column will be certainly of interest for this thesis to see if the financial data differs between the participants in the same case. The adjoining three columns display the day, month, and year of the investigation starting. This information will be necessary to divide our financial analysis into “before investigation”, “during investigation”, and “after investigation”. Therefore, included alongside the three columns is the “decision date” separated by day, month, and year again. The column “Duration” indicates the number of months it took from the start of investigation till the decision date of a certain case. Since we can easily access this data given in the previous columns, we won’t include it in our filtered dataset. The “Nature of Parties Involved” column specifies the market for a case covering multiple companies. Since the scope of this thesis doesn’t cover checking if the nature of parties does impact the stock prices or returns, this column won’t be included. The next column “Party Name(s)” is the most crucial one for us. This column will be the fundamental resource for receiving the unique identification to run the check-up with the financial data and must be included in our dataset. The next 4 columns are referencing to the start date/information of the particular case, followed by 4 columns indicating the case end date/information. This will be considered as useful information to mark the data and see if there is a change in stock prices and returns over the given time frame. The column “ParentCompany” will also be included to state if the company is owned by a different firm or is owning sub-companies. In the datasheet there are dedicated Columns for the firm addresses including the street, the house number, PO-Box, postal code, the city, and the country. Out of that information the country column will be the most important one to specify the data before sending them to the Identifier-Database. However, to make a comparison with the received data, we also include the postal code, city, and the street name. Some of the columns also include question to the specific lines, cases or comments- these won’t be required in our dataset. Furthermore, the columns “History with EC” and “Repeat Offender” will also be included to mark the entries. The columns related to the manufacturing/product industry as well as OECD (The Organisation for Economic Co-operation and Development) information are as well not relevant for this Thesis. Multiple columns are related to the cases as well as their natures. This information might be important for the analysis and

will be therefore included with the data for “EU/non-EU, Points, Report Route, Indicator Report Route, Leniency, Formal Decision, indicator Formal Decision, nature of conduct and the start and the end of the cartel”. All other available fields are not considered important for this thesis. This filtration leaves us with the data fields displayed in Figure 7. A new file will be created including that information named “0. Cartels_Formatted”.

Data columns used for the analysis

| | | | | Adresse | Day Month Year |
|---------------|-----------------|-------------------|-------------------------------|----------|----------------------|
| index | History with EC | Report Route | Indicator Formal Decision | Country | Start Investigations |
| Case # | Repeat Offender | Formal Decision | Indicator Report Route | Postcode | Date Decision |
| Party Name(s) | EU/Non-EU | Leniency | Nature of the Case | City | Start Cartel |
| ParentCompany | Points | Nature of Conduct | Nature of the Formal Decision | Street | End Cartel |

Figure 7: Used data columns for the analysis

Since lot of the dates are not or only partially available, the empty cells have to be taken care of. The procedure is for all dates similar, and the goal is to end up with valid dates to use for further tasks.

```
# Convert "Investigation Day", "Investigation Month", and "Investigation Year" to integers or strings
data['Investigation Day'] = data['Investigation Day'].fillna(1).astype(int)
data['Investigation Month'] = data['Investigation Month'].fillna(1).astype(int)
data['Investigation Year'] = data['Investigation Year'].fillna(0).astype(int)
# Convert "Investigation Year" to string
data['Investigation Year'] = data['Investigation Year'].astype(str)
# Fill leading zeros to "Investigation Day" and "Investigation Month" columns
data['Investigation Day'] = data['Investigation Day'].apply(lambda x: str(x).zfill(2))
data['Investigation Month'] = data['Investigation Month'].apply(lambda x: str(x).zfill(2))
# Merge "Investigation Day", "Investigation Month", and "Investigation Year" columns into "Investigation date" column
data['Investigation date'] = data['Investigation Year'] + '-' + data['Investigation Month'] + '-' + data['Investigation Day']
# Drop original "Investigation Day", "Investigation Month", and "Investigation Year" columns
data.drop(['Investigation Day', 'Investigation Month', 'Investigation Year'], axis=1, inplace=True)
```

Code-Snippet 3: Cleansing of missing dates

The DataFrame named “data” is subjected to data cleaning and modification in Code-Snippet 3. In more detail, it merges the three DataFrame columns titled “Investigation Day,” “Investigation Month,” and “Investigation Year” into a single column titled “Investigation date.” Prior to converting them into integers, it first replaces any missing

values in the “Investigation Day” and “Investigation Month” columns with 1. Additionally, it transforms any missing values in the “Investigation Year” column into an integer and replaces them with 0. The “Investigation Day” and “Investigation Month” columns are then given leading zeros to ensure that they have two digits, and the “Investigation Year” column is changed from an integer to a string. Following that, it combines the “Investigation Day”, “Investigation Month”, and “Investigation Year” columns into a new column called “Investigation Date.” It uses hyphens to join the three columns together to create a date with the format “YYYY-MM-DD”. In order to make room for the newly generated “Investigation date” column, the DataFrame's original “Investigation Day”, “Investigation Month”, and “Investigation Year” columns are finally removed. As stated before a similar process is applied for the “Decision Date”, the “Cartel Start Date” as well as the “Cartel End Date”.

4.2.1 Regular Expression

Before diving into the Regular Expression, it was necessary to define what special characters or symbols are included in the company names. By collecting the specialities of the string, it allows the usage of the correct Regular Expressions to match the right patterns. To do so, Code-Snippet 4 displays the import of the formatted cartel information based on our filtration from the previous chapter with the pandas “read_excel()” method. This allows us to store the data in the “data” DataFrame. For this step only the “Party Name(s)” and the countries are relevant, therefore the data DataFrame is cropped and stored in an alternative DataFrame named “df”. To have a better overview of the data a new column called “Index” is added to the DataFrame and an index for all entries is applied. The “special_df” DataFrame is used to store the identified special characters, that were included in the party names.

```
#Import cartel data
data = pd.read_excel('0. Cartels_Formatted.xlsx', sheet_name='CartelsFirmLevel')

#Gather the "Party Name(s)" only
df = pd.DataFrame(data, columns=["Party Name(s)", "Country"])
df["Index"] = range(len(df))

#Create an empty special dataframe
special_df = pd.DataFrame(columns=df.columns.tolist() + ['Special Char'])
```

Code-Snippet 4: import and preparation of data

Next we define a new method to check for non-alphanumeric or whitespace characters in the party names and return their index and the special character itself. This method is depicted in Code-Snippet 5 and is named “has_special_chars”. This functions input value is a string, and the “if” statement in the supplied function checks whether the input parameter “string” is an instance of the “str” (string) data type. If “string” is not a member of the “str” class, the method returns a tuple with three values: False, -1, and None. The “return” statement accomplishes this by returning the mentioned values (False, -1, None).

```
#Define a function to check for non-alphanumeric or whitespace characters and return their
index and the special character
def has_special_chars(string):
    if not isinstance(string, str):
        return (False, -1, None)
    else:
        match = re.search(r'([a-zA-Z0-9s.,;:/()$+])', string)
        if match and not match.group() in [",", "%", "&", "-"]:
            return (True, match.start(), match.group())
        else:
            return (False, -1, None)
```

Code-Snippet 5: Creation of function to check for special characters and whitespaces

If the “if” condition before evaluates to False, then the “else”-block in the provided function will be executed. In other words, the code within the “else” block is performed if “string” is an instance of “str” and the “if” condition within the “else” block is not met. The first else-statement in this procedure searches for a certain pattern in the “string” variable using the “re.search()” function from the “re” module (Regular Expression module). “r’([a-zA-Z0-9s.,;:/()\$+])’” denotes any character other than an uppercase letter, lowercase letter, digit, whitespace, period, comma, semicolon, colon, forward slash, parentheses, dollar sign, plus sign, or double quote. The “re.search()” function scans the “string” from left to right and looks for the first occurrence of the specified pattern. If a match is discovered, it is stored in the variable (“match”), which contains information about the matched text, such as the matched characters and their starting index in the “string”. The “if” statement following the “re.search()” function checks two conditions: 1. If “match” is not none, it indicates that a match was found. 2. If the matched character (retrieved by using “match.group()”) is not in the list [“,”, “%”, “&”, “-“]. If both conditions are met, the function returns a tuple with three values: True, the starting index of the match (by using “match.start()”), and the matched character

(`match.group()`). If none of the conditions are met, the function executes the “else” block and produces a tuple with three values: False, -1, and None. This means that there was no match or that the matching character was on the exclusion list [`“”`, `“%”`, `“&”`, `“-”`]. After the function has been created, the next step is to loop through the rows of the formatted DataFrame and append to the special DataFrame if party names include special characters.

```
#Loop through rows of formatted DataFrame and append to special DataFrame if Party Name(s)
has special characters
for index, row in df.iterrows():
    has_special, special_index, special_char = has_special_chars(row['Party Name(s)'])
    if has_special:
        row['Special Char'] = special_char
        special_df = special_df.append(row, ignore_index=True)

#Save the special DataFrame to an Excel file
special_df.to_excel("1. Special_chars.xlsx", index=False)
```

Code-Snippet 6: Application of function and storage of results

Code-Snippet 6 displays the application of the constructed method. The given code iterates through the rows of the DataFrame “df” using the “iterrows()” function, which provides an iterator with index and row data pairs. The code uses the previously constructed function “has_special_chars()” for each row, passing the value of the “Party Name(s)” column as an input. The “has_special_chars()” function returns then a tuple with three values, either False, -1, and None or True, the starting index of the match, and the matched character. Those are handed over to “has_special” (a boolean indicating whether the row’s “Party Name(s)” column contains special characters), “special_index” (the index of the special character in the “Party Name(s)” column), and “special_char” (the actual special character). If “has_special” is True, indicating that the “Party Name(s)” column contains a special character, the code updates the “Special Char” column of the current row with the “special_char” value using the “row[‘Special Char’] = special_char” assignment statement. Then, the current row is appended to the DataFrame “special_df” using the “append()” method with “ignore_index=True” to maintain a sequential index for the appended rows. In summary, the code loops through the rows of a DataFrame, checks if the “Party Name(s)” column of each row has special characters using the “has_special_chars()” function, and if so, updates the row’s “Special Char” column and

appends it to a new DataFrame called “special_df”. It is then saving the DataFrame “special_df” to an Excel file named “1. Special_chars.xlsx” using the “to_excel()” method provided by Pandas library. The “to_excel()” method allows saving a DataFrame to an Excel file. The first argument of the “to_excel()” method specifies the file name and the file format. In this case, the file name is “1. Special_chars.xlsx”, which indicates that the DataFrame will be saved to an Excel file with the extension “.xlsx”. An excerpt of the resulting file is displayed in Table 3. Based on this list we can remove duplicates from the column “Special Char” and replace them in the next part.

Table 3: Excerpt of “1. Sepcial_chars.xlsx”

| Party Name(s) | Country | Index | Special Char |
|--|-------------|-------|--------------|
| Société chimique Pointet-Girard SA | Frankreich | 3 | é |
| Société nogentaise de produits chimiques | Frankreich | 4 | é |
| Française des matières colorantes SA | Frankreich | 10 | ç |
| AGROB AG für Grob- und Feinkeramik | Deutschland | 19 | ü |
| Hoesch AG Uüttenwerke | Deutschland | 48 | ü |
| Klöckner-Werke AG Hütte Bremen | Deutschland | 51 | ö |
| Klöckner-Werke AG Georgsmarienwerke | Deutschland | 52 | ö |
| Friedr. Krupp Hüttenwerke AG | Deutschland | 53 | ü |
| Hüttenwerk Oberhausen AG | Deutschland | 59 | ü |
| Rhein Stahl Hüttenwerke AG | Deutschland | 61 | ü |

The Regular Expressions used in the Artifact divides into 6 different categories, each one fulfilling its own purpose. The first one makes use of the list that has been created previously. It provides the information on which special characters have to be replaced. To do so however, the data from the formatted cartel information will be imported first. This step is displayed in Code-Snippet 7, it reads the Excel file named “0. Cartels_Formatted.xls” using the “pd.read_excel()” function from the Pandas library. The data from the sheet named “CartelsFirmLevel” is then stored into the DataFrame called “data”. This data is then used to create a new DataFrame “df”.

```
#Read the data from the formatted cartel information
data = pd.read_excel('0. Cartels_Formatted.xls', sheet_name='CartelsFirmLevel')
```

[...]

```
#Creation of new dataframe
df = pd.DataFrame(data)
```

Code-Snippet 7: Import of formatted cartel information

During the next step (first category), the given code in Code-Snippet 8 is used to replace occurrences of the specified special characters and with common letters in the “Party Name(s)” column of the DataFrame named “df” using the “str.replace()” method provided by Pandas. It then uses the “str.strip()” method to remove any leading or trailing whitespace from the generated strings before assigning the revised column back to the “Party Name(s)” column of the DataFrame “df”. This will replace special characters with common letters and remove leading/trailing whitespace from the generated strings in the “Party Name(s)” column. This step has to be done since both Orbis and openfigi do not recognize those special characters.

```
#Category 1- replace special characters with certain "usual" letters
df['Party Name(s)'] = df['Party Name(s)'].str.replace("É", "E").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("é", "e").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("è", "e").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ë", "e").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("Á", "A").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("à", "a").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ã", "a").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ç", "c").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ñ", "n").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ï", "i").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ó", "o").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ô", "o").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("õ", "o").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ø", "o").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ü", "u").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ý", "y").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("ž", "z").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("E", "E").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("-", " ")
```

Code-Snippet 8: Category 1- Replacement of special characters using Regular Expression

The second step uses the code displayed in Code-Snippet 9 to select the column “Party Name(s)” from the DataFrame “df” using DataFrame indexing notation “df[‘Party Name(s)’]” and applies the “str.replace()” method to it. This is a string method provided by Pandas for performing string replacement operations on a column of a DataFrame. The first argument of “str.replace()” in each line is the regular expression pattern to be searched for in the column “Party Name(s)”. in the first line e.g., the pattern is “[sS][pP][aA]”, which is a regular expression that matches a space followed by the characters “s” or “S”, followed by “p” or “P”, followed by “a” or “A”. This pattern is used to search for occurrences of the word “spa” regardless of the spelling in the column.

The second argument of “str.replace()” for every line is the replacement string to be used for replacing the matched pattern. In those cases, an empty string (“”) is used as the replacement, which means that any occurrences of the matched pattern will be removed from the column. Since the “regex” parameter is set to “True” it indicates that the first argument provided is a regular expression pattern. This allows for using regular expressions for pattern matching in the “str.replace()” method. The result of the “str.replace()” operation is assigned back to the same column “Party Name(s)” in the DataFrame “df”, effectively replacing any occurrences of the word “spa” in any spelling with an empty string in the column.

```
#Category 2- regex to remove different company types, abbreviations and redundant information
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [sS][pP][aA]', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'[gG][mM][bB][hH]', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'mbH$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [aA][gG]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [sS][aA]', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'^[sS][aA] ', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [aA][sS]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [bB][vV]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [kK][gG]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [aA][bB]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [pP][lL][cC]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [cC][vV]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [rR][lL]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [eE][vV]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [nN][vV]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [bB][vV]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [sS][lL]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [tT][dD]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [pP][lL][cC]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [oO][yY][jJ]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [oO][yY]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [lL][tT][dD]', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [lL][lL][cC]', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [sS][rR][lL]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [iI][nN][cC]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' [cC][oO]$', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'Zuivelfabriek ', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'Zuivelindustrie ', '', regex=True)
```

Code-Snippet 9: Category 2- Regex to remove additional information

Both the third as well as the fourth category are displayed in Code-Snippet 10. Both categories select the column “Party Name(s)”, and apply the “str.replace()” method used

previously. The first line uses the regular expression pattern “[.*?\\]|\\(.*?\\)” to search for occurrences of text enclosed in square brackets or parentheses in the column “Party Name(s)”. The matched pattern is then replaced with an empty string (“”), effectively removing any text enclosed in square brackets or parentheses from the column. The second “str.replace()” operation uses the regular expression pattern “r[^\\w\\s]”, which matches any character that is not a word character (i.e., alphabetic, numeric, or underscore) or a whitespace character. The matched pattern is replaced with an empty string (“”), effectively removing any non-word and non-whitespace characters from the column “Party Name(s)” as well.

```
#Category 3- remove country codes in brackets from names
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'[.*?\\]|\\(.*?\\)', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'[^\\w\\s]', '', regex=True)

#Category 4- RegEx to remove different company types at the beginning or Ending of a line
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'^[nN][vV] ', '', regex=True)
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r'[sS][aA][rR][lL]$', '', regex=True)
```

Code-Snippet 10: Category 3 and 4- Regex to remove country codes and company types at the beginning/ending

The first “str.replace()” method for category 4 in Code-Snippet 10 uses the regular expression pattern “^[nN][vV] “ to search for occurrences of text that starts with “nv” regardless of upper or lower case followed by a space at the beginning of the column “Party Name(s)”. The matched pattern is replaced with an empty string (“”), effectively removing the “nv “ prefix from the text in the column. The second “str.replace()” operation uses the regular expression pattern “r [sS][aA][rR][lL]\$” to search for occurrences of text that ends with “ sarl” regardless of upper or lower case at the end of the line of an entry in the column Party Name(s). The matched pattern is replaced with an empty string (“”), effectively removing the “ sarl” suffix from the text in the column. For all 4 “str.replace()” methods the “regex” parameter is set to “True”, indicating that the provided patterns are regular expressions. The results of these operations are assigned back to the same column “Party Name(s)” in the DataFrame “df”. In the 5th Category of Regular Expression application, displayed in Code-Snippet 11, a series of string replacement operations on the column “Party Name(s)” of the DataFrame “df” is performed in order to remove specific substrings from the text in that column. The first “str.replace()” method for instance replaces occurrences of the substring “ Co.” with an

empty string ("") using the "str.replace()" method, and then removes any leading or trailing white spaces from the resulting text using the "str.strip()" method. This method is used for often occurring company types that are consistent in their appearance.

```
#Category 5- remove defined words from company names
df['Party Name(s)'] = df['Party Name(s)'].str.replace(" Co.", "").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("SA de CV", "").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("& Co. ", "").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("Corporation", "").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("Coöperatieve", "").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("Limited", "").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace(" LTD ", "").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("Ltd.", "").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("Inc.", "").str.strip()
df['Party Name(s)'] = df['Party Name(s)'].str.replace("Company", "").str.strip()
```

Code-Snippet 11: Category 5- Regular Expression to remove defined words

The 6th and last category serves the purpose to remove any multiple whitespaces that are left after the clearing processes. The application of this category is displayed in Code-Snippet 12.

```
#Category 6- remove multiple spaces that are left over
df['Party Name(s)'] = df['Party Name(s)'].str.replace(r' +', ' ', regex=True)
```

Code-Snippet 12: Category 6- Clearing of whitespaces

Once all 6 categories have been applied to the dataset, it will be exported in batches of 999 lines at maximum, since this is the limit for the Orbis import. The given code in Code-Snippet 13 is used to export data from a DataFrame "df" into two separate Excel files, each containing a subset of rows from the original DataFrame.

```
#Export the first 999 rows
df.iloc[:999].to_excel("2. Companies clean.xlsx")

#Export the next 999 rows starting from row 1000
df.iloc[999:1998].to_excel("3. Companies clean2.xlsx")

# Assign the df to the new DataFrame "companies"
companies = df["Party Name(s)"]
```

Code-Snippet 13: Batch export of the DataFrame

The first line of code exports the first 999 rows from the DataFrame “df” into an Excel file named “2. Companies clean.xlsx”. The `iloc` attribute is used to select rows from the DataFrame based on their positional index, and `:999` specifies the range of rows to be selected, which includes all rows from the beginning (index 0) up to but not including row 999. The “`to_excel()`” method is then called on the selected subset of rows to export them into an Excel file with the specified name. The second line of code uses the same rule to export line 999 to 1998 to create the file “3. Companies clean2.xlsx”. In summary, these two lines of code are used to export data from the DataFrame “df” into two different Excel files, the first comprising the first 999 rows of the original DataFrame and the second including the next 999 rows beginning with row 1000. The last line then creates a new DataFrame to store all the data in a new independent DataFrame called “companies”.

4.3 Receive Identifier

Following the creation of a cleansed list of all company names, the next step is to send this information to a service or database that provides a unique identifier for each company. This step is required in order to acquire correct financial information from Refinitiv in the following stage. Refinitiv can only receive a unique identity via an interface, but not a search by company name. Two distinct ways will be employed for the stage of acquiring the identifier. The first option, which is detailed in 4.3.1 Refinitiv Python API via Open will employ a Jupyter Notebook script to transmit the prepared firm names to `openfigi` via an API call and receive the unique identifier. The second method detailed in Chapter 4.3.2 Orbis is a more manual method that uploads the exported excel file directly to the Orbis database. The user can revalidate the uploaded entries and compare the results with alternative findings on Orbis. The list can be retrieved and imported again once the verification and validation processes are done.

4.3.1 Refinitiv Python API via OpenFIGI

OpenFIGI is a worldwide identifier system for financial instruments such shares, options, futures, and fixed income instruments. It provides a unique, machine-readable, and standardized identity for financial instruments, which aids in the financial industries efforts to expedite and automate procedures related to trade execution, clearing, settlement, reporting, and data management. Bloomberg LP created OpenFIGI, an open, community-driven initiative, in 2009 in conjunction with other financial firms and

industry participants. Its goal is to overcome the issues of identifying and maintaining financial instrument data, which can be complex and fragmented due to the many naming conventions, codes, and identifiers used by exchanges, data suppliers, and financial institutions. Each financial instrument is assigned a unique 12-character alphanumeric code known as a FIGI (Financial Instrument Global Identifier) by OpenFIGI. FIGIs are designed to be persistent, which means they do not change over time, and portable, which means they may be utilized across different systems and platforms. OpenFIGI provides a RESTful API (Application Programming Interface) through which developers can access the FIGI database and receive information on financial instruments such as their security type, exchange code, ticker symbol, market segment, and other pertinent data. Financial institutions, data providers, software developers, and other stakeholders rely on the OpenFIGI API to incorporate FIGI-based identity and data management capabilities into their applications and workflows (*Bloomberg Launches Online Request Utility and New Mapping Tools for the Financial Instrument Global Identifier (FIGI) | Press | Bloomberg LP*, n.d.). The OpenFIGI API therefore allows the mapping from third-party identifiers to FIGI based on the API call. The API is free and open to the public, though unauthenticated traffic will be subject to a lower rate-limit. The Rate limits for the Search/Filter API is restricting the use without an API key by 5 requests per minute, 15.000 maximum results and 100 results per page on a maximum amount of 150 pages. In case of a successful transmission the API will return the response code 200, which results in the response message “OK”. If the response code is greater then 400, the request and the transmission of a result has failed (*OpenFIGI API | OpenFIGI*, n.d.). The API script begins at the spot where the last script ended with the export of the excel files “2. Companies clean.xlsx” and “3. Companies clean2.xlsx”. the same DataFrame used for the export is now used to create the new DataFrame “companies”. The code shown in Code-Snippet 14 involves the creation and initialization of three variables related to the previously created DataFrame “df”.

```
#Create three variables consisting of 2 DataFrames
companies = df["Party Name(s)"]
request_counter = 0
all_rows = pd.DataFrame()
```

Code-Snippet 14: Creation of Variables for API script

The first line of code creates a new variable called “companies” and assigns it the data from the DataFrame “df”’s “Party Name(s)” column. The column “Party Name(s)” is extracted from the DataFrame using square bracket notation with the column name as the key, and the resulting sequence of values is saved in the “companies” variable. The subsequent code line declares a new variable called “request_counter” and assigns it the value 0. This variable is used in the code to keep track of the processing count due to the API-Limitation by OpenFIGI. The third line generates a new DataFrame named “all_rows” by invoking the pd.DataFrame() function with no arguments, resulting in an empty DataFrame with no rows or columns. This empty DataFrame is then saved in the variable “all_rows” to be used for the next coding step.

```
for company in companies:
    url = "https://api.openfigi.com/v3/search"
    data = { "query": company }
    headers = { "Content-Type": "application/json" }
    response = requests.post(url, json=data, headers=headers)
    if response.status_code == 200:
        result = response.json()
        try:
            df = pd.DataFrame(result)
            # Convert the JSON strings in the 'data' column into a table-like structure
            new_df = json_normalize(df['data'])
            # Take only the first row of the resulting DataFrame
            row = new_df.iloc[0, :]
        except IndexError:
            # Handle the case where the index does not exist
            print("The index ", company, " does not exist, skipping")
        else:
            first_row = new_df.iloc[0, :]
            # Append the first row to the all_rows DataFrame
            all_rows = all_rows.append(first_row, ignore_index=True)
    else:
        print("Request failed for", company, "with status code:", response.status_code)

    request_counter += 1
    if request_counter % 5 == 0:
        time.sleep(61)

#Save the all_rows DataFrame as an Excel file
all_rows.to_excel('all_companies.xlsx', index=False)
```

Code-Snippet 15: OpenFIGI API request Loop

Code-Snippet 15 begins with a for-loop that iterates through the DataFrame “companies” that was previously constructed (in Code-Snippet 14). It signifies that the following lines of code will be executed once for each element in the list of companies, with the current element referred to as company. Then in the next line the openfigi-search API URL gets assigned to the variable “url”, which is the endpoint for the API (Application Programming Interface) provided by OpenFIGI for searching company information. Next is the preparation of the data payload, which is going to be sent as JSON data in the request body when making an API call. It contains a single key-value pair where the key is “query” and the value is the current company name. The request headers with the content type is assigned to be “application/json”. By using the “request.post()” function a HTTP POST request to the URL specified before is sent, passing the “data” dictionary as JSON data in the request body and setting the previously defined header for the request. The response from the server is then stored in the “response” variable. The next step opens an If-Statement checking if the HTTP status code of the response is equal to “200”, which indicates a successful response. If this is the case, the JSON data then gets extracted from the response and assigned to the “result” variable. By opening a Try-block, error handling gets enabled, which is responsible for catching expected index errors, in case the company name is not know or recognized. It then prints out the error message indicating that the index does not exist for the current “company” element, and the loop iteration is skipped. The “Try-block” itself, reuses the “df” Dataframe and assigns it the DataFrame from the received result, converting it into a tabular structure. Afterwards a new DataFrame “new_df” is created by normalizing the JSON data in the “data” column of the “df” DataFrame, which converts nested JSON data into a flat tabular structure. However only the first line of the newly created DataFrame is important and therefore gets selected and assigned to the variable “row”, as well as assigned to the variable “first_row” if no exception has been thrown. The “first_row” gets then appended to the existing DataFrame “all_rows” ignoring the index of the “first_row” DataFrame and creating a new index for the combined DataFrame. In case the previous If-Statement is not true, the else-block gets entered, the code then prints the error message indicating that the request for the current “company” element has failed, along with the HTTP status code returned by the server. Independent of the HTTP status code, it then increments the value of the variable “request_counter” followed up by the check if the value of this variable is a multiple of 5, which indicates that a certain number of requests have been made, in this case every 5th request. If this is the case, it pauses the execution of the code for 61 seconds using the

“time.sleep()” function. This has to be done due to the API usage limitation set to a maximum of 5 requests per minute. The DataFrame “all_rows” is then saved as an Excel file with the filename “all_companies.xlsx” by using the “to_excel()” method provided by the “pandas” library. This export includes the column figi as the unique company identifier and therefore the important information for this thesis, as well as the company name, the company ticker displayed in Table 4 and various additional information, that is not relevant for this thesis.

Table 4: “all_companies.xlsx” table excerpt

| Figi | Name | Ticker | ... |
|--------------|--------------------------|----------------------|------------|
| BBG0003HKQL5 | BOEHRINGER MANNHEIM GMBH | BMAN 5.125 09/19/96 | ... |
| BBG00007TWD7 | CIBA SPECIAL CHEM FIN LX | BASGR 4.875 06/20/18 | ... |
| BBG0013S2T15 | SANDOZ CORPORATION | SANDOZ | ... |
| BBG0013T34S8 | IMPERIAL CHEM INDUST PLC | ICI | ... |
| BBG00012F5B3 | WESSEL-WERK GMBH | WESEDE L 12/12/10 A | ... |
| BBG000158PH4 | HEURTEY PETROCHEM SA | ALHPC L 12/10/12 FA | ... |
| BBG00PQ29LC2 | TUBES HAREN NIMY | TUBVPEUR | ... |
| BBG00B9SKPQ8 | ENCI Wind Farm | WD24A028 | ... |
| BBG00016XRN3 | PFEIFER LANGEN KG | PFLADE L 08/10/14 | ... |
| BBG0008F2QQ9 | TALLINNA SADAM | TALSAD F 03/17/09 | ... |
| ... | ... | ... | ... |

With this approach the figi-company identifiers for 223 different companies (12,85%) have been received with the extensive data preparation and 352 with a basic data preparation, which displays only a small part (20,2999%) of the given 1736 companies and is therefore not feasible for this thesis.

4.3.2 Orbis

Bureau van Dijk, a company that specializes in business intelligence, data analytics, and corporate data solutions, provides the Orbis database as a commercial database. Orbis is a global company database that provides extensive information on private and public firms worldwide, including financials, ownership, M&A deals, news, and other pertinent business data. Businesses, financial institutions, researchers, and other professionals utilize Orbis for a variety of objectives, including due diligence, risk assessment, market research, competitive analysis, and compliance. The database has information on millions of businesses from many industries and countries, and it is constantly updated to give

current and reliable data for corporate decision-making. Bureau van Dijk, a subsidiary of Moody's Analytics, is a major provider of business information and corporate data solutions (*Company Structures & Ownership Information | Bureau van Dijk, n.d.*). Orbis allows the search with free text. As displayed in Figure 8 the user is allowed to type in any company name and Orbis suggests matching companies. The best match with the given string is atop of the suggested entries. The given entries can then be selected, and its information is displayed to the user.

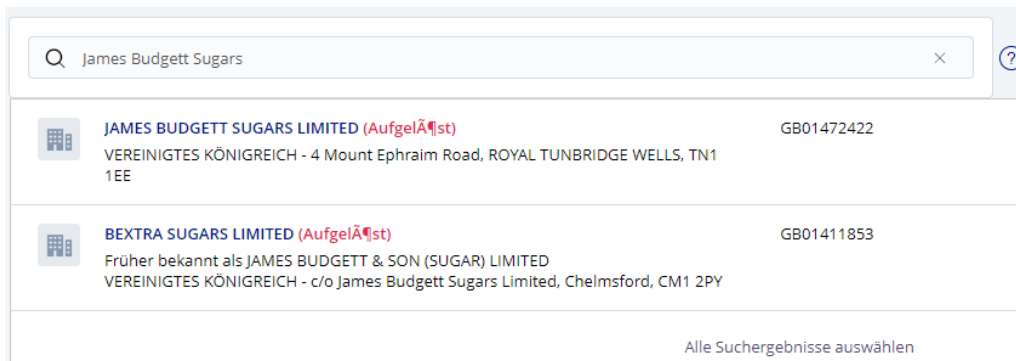


Figure 8: Example Orbis free text search

Orbis itself is already able to give a lot of insights of given companies. As shown in Figure 9, Orbis displays a brief overview of the company in a short summary. It provides the current status of the company (for the example “Aufgelöst” in red, indicates that the company has been dissolved), the Bureau van Dijk ID (BvD ID), the Orbis ID as well as the registered location of the firm and if it is a public or private company.

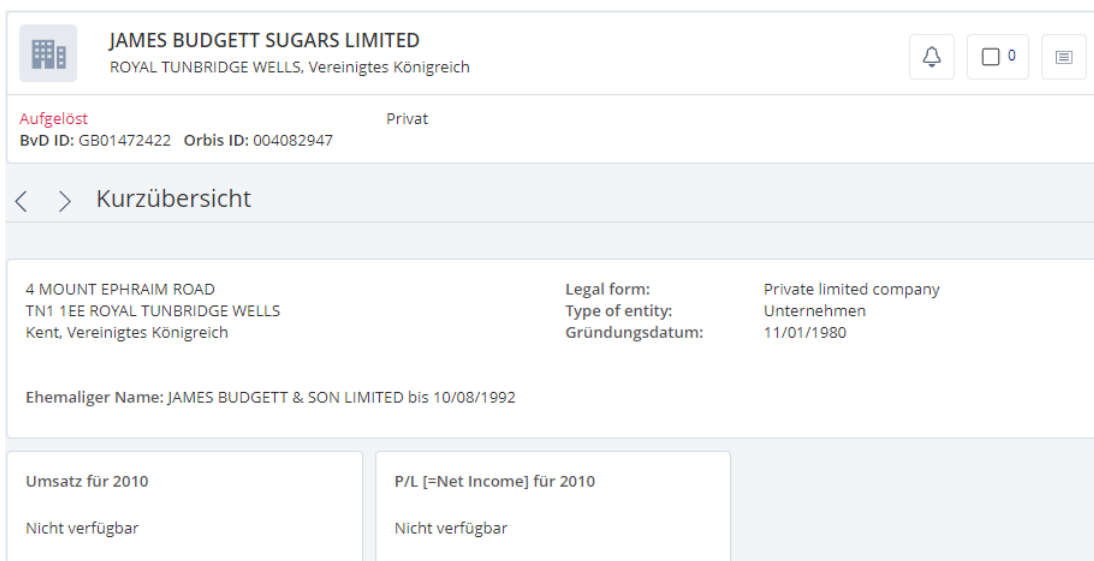


Figure 9: Short summary of Orbis for “James Budgett Sugars Limited”

Furthermore, data such as the founding date, the legal form and type of entity is also given by Orbis. The “Short Summary” page, however, is only one of multiple pages shown for a given company. Orbis additionally provides an own page for the contact information of the company including additional addresses, the industry and activities, several identification numbers, foundation and status information, financial closing information and numerous more pages. To follow the purpose of making use of the previously created excel exports, the usage of the Orbis tool “bundled search” is required. With this function the excel files can be imported and a new mapping can be generated. As depicted in Figure 10 it then lets the user choose if there are specific columns for the company name, the city, the country, and an ID-number. To achieve the best quality of the provided data and to find as many correct matches as possible there will be three different approaches in this thesis for the Orbis upload. The first search will only include the prepared company names without countries and without cities. The second approach will additionally include the company countries. The third and final approach will then include the company names, the company countries as well as the company cities. The results will be compared with each other, and the consequent steps will be continued with the highest matching result.

Link search fields with columns in your uploaded file

| Search fields | Columns in file |
|------------------|-----------------|
| Unternehmensname | Party Name(s) ▼ |
| Stadt | ▼ |
| Land | ▼ |
| ID Nummer | ▼ |
| Own ID | ▼ |

Figure 10: Orbis mapping selection for bundled search

To have the three different approaches the previously created excel files “2. Companies clean.xlsx” and “3. Companies clean2.xlsx” are uploaded three times to Orbis to create three different bundled search batches for the mentioned scenarios to compare to each other. Table 5 presents a comparison among three scenarios, denoted as “Scenario 1”, “Scenario 2”, and “Scenario 3”, respectively. In “Scenario 1”, only the company name is included, while “Scenario 2” includes both the company name and country, and “Scenario

3” includes the company name, country, and city. The data in the table unmistakably reveals that “Scenario 1” yields a higher match rate, capturing almost 5% more companies compared to “Scenario 2”, as well as almost 20% more than “Scenario 3”. This suggests that “Scenario 1” is more suitable for subsequent steps than both “Scenario 2” and “Scenario 3”. It is likely that the additional data limits the search engine and therefore lower the hit-rate. The findings from this comparison have implications for the choice of scenario in the further analysis and processing of the data and the results from “Scenario 1” will be considered.

Table 5: Matched companies based on country and city

| | Name | Country | City | Total | Matched | Not matched | Matched % |
|-------------------|------|---------|------|-------|---------|-------------|-----------|
| Scenario 1 | Yes | No | No | 1734 | 1131 | 603 | 65,22 |
| Scenario 2 | Yes | Yes | No | 1734 | 1051 | 683 | 60,61 |
| Scenario 3 | Yes | Yes | Yes | 1734 | 791 | 943 | 45,62 |

In addition to its automated matching process, Orbis also provides the option for manual correction of values that do not align with their data, allowing users to manually correct matches as needed. However, it should be noted that manual corrections have not been made for the data presented in Table 5, but they are considered for the data that is subsequently passed on to Refinitiv. This manual correction feature in Orbis offers flexibility and customization for users to ensure accurate and reliable data matching and processing in their analysis. In “Scenario 1,” a total of 1131 companies were provided by Orbis. The manual feature was utilized to select the matches found on Orbis with a “B-Score.” The “B-Score” is assigned by Orbis when there is uncertainty whether the identified company is the same as the one searched for. Through this process, a total of 1290 companies were found on the Orbis database, resulting in an increase of the “Matched %” to 74.44%. To merge the Orbis table with the original input, the Orbis “Matching Table” needs to be saved since it retains the original index as well as the found Orbis ID. This step leads to the creation of the two files “Export_Companies clean.xlsx” and “Export_Companies clean2.xlsx” and is essential to ensure that all the necessary columns and data are still available when merging the final Orbis export with the original cartel data. As shown in Figure 11, one notable feature of Orbis is that it allows users to select the columns they want to display. Given the purpose of this tool in providing company identifiers, such as the BvD-Number, Orbis ID, and ISIN-number, these

columns are particularly relevant for this thesis. Additionally, the “Flags” column in the Orbis data indicates if the business is listed on the stock exchange, inactive, or a branch, which can be used for further analysis and categorization of the companies under consideration. This flexibility in column selection and availability of relevant data in Orbis enhances the usability and effectiveness of the tool for the analysis of stock prices and stock price returns for the impacted companies. Upon downloading the Orbis result as an Excel file (“Orbis_export.xlsx”), a crucial step in the process of obtaining financial information via Refinitiv has been completed. This preliminary work of extracting data from Orbis and exporting it to a usable format sets the foundation for further analysis and enables seamless integration with Refinitiv for obtaining the relevant financial data for the impacted companies. This efficient and streamlined approach allows for a smooth transition from Orbis to Refinitiv, facilitating the subsequent steps in the analysis of stock prices and stock price returns for the companies under study.

| <input type="checkbox"/> | Unternehmensname Latin alphabet | | Flags | ISO Länder code | Letztes verf. Jahr | BvD ID Nummer | BvD Abschlussnummer | Orbis ID Nummer | ISIN-Nummer |
|-------------------------------|------------------------------------|---|-------|-----------------------|--------------------------|-----------------|------------------------|--------------------|--------------|
| × <input type="checkbox"/> 1. | TOTALENERGIES SE |    | | FR | 2022 | FR542051180 | FR542051180IC | 003846950 | FR0000120271 |
| × <input type="checkbox"/> 2. | BP PLC |    | | GB | 2022 | GB00102498 | GB00102498IC | 003902763 | GB0007980591 |
| × <input type="checkbox"/> 3. | SAMSUNG ELECTRONICS CO.,LTD. |    | | KR | 2022 | KR1301110006246 | KR1301110006246IC | 006529401 | KR7005930003 |
| × <input type="checkbox"/> 4. | EQUINOR ASA |    | | NO | 2022 | NO923609016 | NO923609016IC | 005600703 | NO0010096985 |
| × <input type="checkbox"/> 5. | MITSUBISHI CORPORATION |    | | JP | 2021 | JP5010001008771 | JP5010001008771IC | 051056945 | JP3898400001 |
| × <input type="checkbox"/> 6. | ENGIE |    | | FR | 2022 | FR542107651 | FR542107651IC | 071889487 | FR0010208488 |
| × <input type="checkbox"/> 7. | BASF SE |    | | DE | 2022 | DE7150000030 | DE7150000030IC | 060521000 | DE000BASF111 |

Figure 11: Orbis extract example

Once the final Orbis export has been saved, the next step is to unify the different data sources to have one complete set of information. To do so we read, prepare, and extract the different data sources.

```

# Read the data from the formatted cartel information
df_temp1 = pd.read_excel('2. Companies clean.xlsx', sheet_name='Sheet1')
df_temp2 = pd.read_excel('3. Companies clean2.xlsx', sheet_name='Sheet1')
df1 = pd.concat([df_temp1, df_temp2])
# Orbis Matching files received after uploading the
df_temp1 = pd.read_excel('Export_Companies clean.xlsx', sheet_name='Page 1')
df_temp2 = pd.read_excel('Export_Companies clean2.xlsx', sheet_name='Page 1')
df2 = pd.concat([df_temp1, df_temp2])

df3 = pd.read_excel('Orbis_export.xlsx', sheet_name='Results')
partial_merge = pd.merge(df1, df2, left_on='index', right_on='Own ID', how='left' )

```

Code-Snippet 16: Import and merge of cartel data and Orbis-matched data

The prepared cartel data is first read from two Excel files called “2. Companies clean.xlsx” and “3. Companies clean2.xlsx” using Code-Snippet 16, and then it is concatenated into a single DataFrame called “df1” using the “pd.concat()” function. The function then reads the matching data from two Excel files called “Export_Companies clean.xlsx” and “Export_Companies clean2.xlsx.” Once more, the code uses the “pd.concat()” function to combine the information from these two files into a single DataFrame called “df2”. The code then also takes information from the file named “Orbis_export.xlsx,” which contains the outcomes of the Orbis matching procedure and stores it in a DataFrame called “df3”. The “pd.merge()” function is then used to partially merge the “df1” and “df2” DataFrames. The “index” column of “df1” and the “Own ID” column of “df2” are used in the merge. The “how” option is set to “left”, which implies that all of the rows from the left DataFrame (“df1”) and any matching rows from the right DataFrame (“df2”) are included in the output DataFrame called “partial_merge”.

```

#generate custom IDs
new_ids = [f'LID{i+1}' for i in range(len(partial_merge[partial_merge['Matched BvD ID'].isnull()]))]

#replace empty values with custom IDs
partial_merge.loc[partial_merge['Matched BvD ID'].isnull(), 'Matched BvD ID'] = new_ids

full_merge = pd.merge(partial_merge, df3, left_on='Matched BvD ID', right_on='BvD ID number', how='left')
full_merge.to_excel("4. FULL_MERGE.xlsx", index=False)
full_merge.head()

```

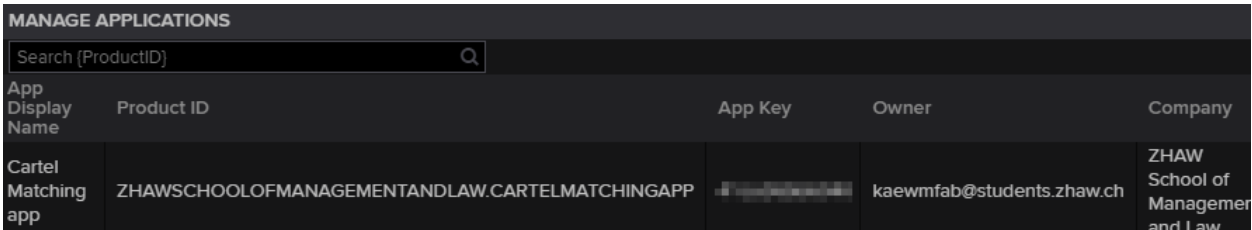
Code-Snippet 17: Creation of the fully merged DataFrame

Since not all companies were matched by Orbis, lot of the lines are still empty. This causes issues when merging the two DataFrames with each other, since all empty lines will be joined with every other empty line of the key. On the other side an inner join merge would not result into having the correct data, since the goal is still to keep all the data, as well if no match has been found. For the entries in the “partial_merge” DataFrame that do not have matching BvD IDs in the “Matched BvD ID” column, Code-Snippet 17 generates custom IDs. The “loc” function is then used to assign the new IDs to the rows that have null values in the “Matched BvD ID” column. The new IDs are created using list comprehension. The “pd.merge()” function is used to complete the entire merge between the “partial_merge” and “df3” DataFrames, and the “full_merge” DataFrame is then exported to an Excel file called “4. FULL_MERGE.xlsx” using the “to_excel()” function.

4.4 Receive RIC from ISIN

The Reuters Identification Code, or RIC, is a Thomson Reuters system for identifying financial assets such as stocks, bonds, commodities, currencies, and indexes. In the financial industry, RICs are frequently used as a unique identifier to simplify the electronic transfer of financial data, such as market data, news, and trading information. The RIC is a string of alphanumeric characters that normally consists of a root code followed by additional codes that identify various financial instrument properties such as exchange code, currency code, and expiration date. For example, a stock's RIC could be “AAPL.O”, where “AAPL” is the root code for Apple Inc. and “.O” signifies that it is listed on the NASDAQ exchange. Financial professionals, such as traders, investors, and analysts, use RICs for a number of objectives. One of the primary applications of RICs is the distribution of real-time market data. RICs are used by financial data vendors, exchanges, and other market participants to uniquely identify financial instruments when transmitting data, including as quotations, trades, and order book information, in order to promote efficient communication and data processing across different systems and platforms. In summary, the Reuters Identification Code (RIC) is a widely used system for uniquely identifying financial instruments, facilitating efficient financial data communication and processing, and supporting various functions in the financial industry such as market data dissemination, trading, and news services (MyRefinitiv, 2023). The RIC (Reuters Instrument Code) serves as a critical criterion in numerous functions within

the Refinitiv Eikon library. As other identifiers, such as ISIN or other codes, may not be accepted in certain functions, the availability of the RIC becomes indispensable. The RIC acts as a unique identifier that allows seamless integration with the Eikon library, enabling efficient and accurate retrieval of financial data for further analysis and processing. Its significance lies in its widespread use and compatibility with various functions and tools within the Refinitiv Eikon ecosystem, making it an essential component for obtaining comprehensive financial information and conducting in-depth analyses. The use of a python script to automatically retrieve RIC (Reuters Instrument Code) for ISIN identifiers via the Eikon API key enhances the efficiency of data retrieval. The script executes a function that returns the RIC based on the provided ISIN. To make API calls, the user needs to have a valid API key generated from the Refinitiv “App Key Generator”, as shown in Figure 12, and ensure that the “Refinitiv Workspace” is open and running in the background for the script to execute successfully. This integration enables the communication between Orbis and Refinitiv Eikon, leveraging the power of both platforms for comprehensive financial analysis.



| MANAGE APPLICATIONS | | | | |
|---|--|------------|---------------------------|-----------------------------------|
| Search (ProductID) <input type="text"/> | | | | |
| App Display Name | Product ID | App Key | Owner | Company |
| Cartel Matching app | ZHAWSCHOOLOFMANAGEMENTANDLAW.CARTELMATCHINGAPP | [REDACTED] | kaewmfab@students.zhaw.ch | ZHAW School of Management and Law |

Figure 12: Refinitiv App Key Generator

Once the prerequisite conditions are met, the python script can be executed to facilitate the data retrieval process. As illustrated in Code-Snippet 18, the script begins by assigning the previously generated API key to a newly created variable “api_key”, which represents the authentication key required for interacting with the Eikon API. The function “set_app_key()” is then invoked, passing the “api_key” variable as a parameter to authenticate the script with the Eikon API. The next step involves importing the previously created DataFrame “full_merge” and passing it to a DataFrame labelled as “df”. Subsequently, a new empty column labeled as “RIC” is added to the imported DataFrame “df”, providing a space for storing the retrieved Reuters Instrument Codes (RICs) for the respective ISIN identifiers. This sets the stage for further data manipulation

and analysis using the integrated financial information from Orbis and Refinitiv Eikon in subsequent steps of the script.

```
#Set your EIKON API key and app ID
api_key = '41ca36bb60404b288a4d103722fa405de3d5d732'

# Authenticate with EIKON API by passing app ID and API key as parameters
ek.set_app_key(api_key)

# Read the data from the previously created full_merge DataFrame
df = full_merge

# Create a new column to store RIC codes
df['RIC'] = ''
```

Code-Snippet 18: Refinitiv Eikon API setup

After executing the previous code, the next step involves creating a loop that iterates through each row in the DataFrame “df”. The loop is depicted in Code-Snippet 19 and “index” and “row” represent the index and data in each row of the DataFrame, respectively. Inside the loop, the value in the column labelled “ISIN number” from the current row of the DataFrame is retrieved and stored in a variable named “isin”.

```
# Loop through each row in the DataFrame
for index, row in df.iterrows():
    # Get ISIN from the "ISIN" column
    isin = row['ISIN number']

    try:
        # Check if ISIN is not a whitespace
        if not isin.isspace():
            # Retrieve RIC for the ISIN
            RIC_codes = ek.get_symbology([isin],
from_symbol_type="ISIN",to_symbol_type="RIC")
            df.at[index, 'RIC'] = RIC_codes.iloc[0]['RIC']

    except Exception as e:
        df.at[index, 'RIC'] = 'no match available'

# Save the updated DataFrame to a new Excel file
df.to_excel('5. Orbis_export_with_RIC.xlsx', index=False)

new_df = df[df['RIC'] != 'no match available'].reset_index(drop=True)

# This list only contains companies with RIC
new_df.to_excel('6. Orbis_export_with_RIC_Clean.xlsx', index=False)
```

Code-Snippet 19: For-Loop and export of RIC-DataFrame

Within the “Try-Catch” block, the code then checks if the value in the “isin” variable is not just whitespace (i.e., spaces, tabs, or newlines) using the “isspace()” method, which returns “True” if all characters in the string are whitespace characters, and “False” otherwise. If this “If-Condition” is true, the next step is to call a function named “get_symbology()” from the Eikon library identified as “ek”. This function is used to retrieve RIC (Reuters Instrument Code) codes for the given ISIN. The “isin” value is passed as a list to the function, along with the source symbol type “from_symbol_type” specified as “ISIN” and the target symbol type “to_symbol_type” specified as “RIC”. The value in the “RIC” column of the current row in the DataFrame “df” then gets updated with the RIC code retrieved from the “RIC_codes” DataFrame. The “at” accessor is used to specify the row by its index (index variable) and the column label “RIC”, and the value is assigned using the “=” operator. The “iloc” accessor is then used to retrieve the value from the first row (iloc[0]) of the “RIC_codes” DataFrame in the column labelled “RIC”. In case the code inside the “Try-Catch” block throws an exception, the exception handling block catches any exceptions that may occur in the code. The “Exception” class is a generic base class for all exceptions in Python, and “as e” assigns the caught exception to a variable named “e”, which can be used to access the details of the exception. Additionally, in the exception block, the script updates the value in the “RIC” column of the current row in the DataFrame “df” with the string “no match available” in case an exception occurs in the code, indicating that the RIC code could not have been retrieved. It then saves the updated DataFrame “df” to a new Excel file named “5. Orbis_export_with_RIC.xlsx” in the current directory. The “index” parameter is set to “False” to exclude the index column from the saved Excel file. Since it is only possible to receive financial data from Refinitiv if the RIC is actually given, the next line drops the lines where there is no RIC available and exports this new DataFrame as “6. Orbis_export_with_RIC_clean.xlsx”. The process of retrieving RIC codes for the provided ISIN identifiers and updating the DataFrame with the retrieved information is now concluded. This results in an Excel export that encompasses comprehensive company data, including Name, relevant flags, ISO-Country code, BvD-ID, Orbis ID, ISIN number, and RIC code. The resulting export is now primed for the use in retrieving financial company data, facilitating further analysis and decision-making processes.

4.5 Retrieve financial data and first interpretation

The process of obtaining financial information through Refinitiv necessitates a valid license and the API key mentioned earlier. Various queries are executed to obtain diverse viewpoints of financial data. The first phase is acquiring the financial information for a singular company within the range of imported dates that includes the start cartel date, end cartel date, and decision date. Due to the fact that the cartel data involves cases that commenced even prior to 1960 and many companies from the cartel data are not listed on the stock market, there is no stock price information available for those firms. Moreover, data availability is restricted by intervals, and hence, the monthly interval provides information further back in time than the daily stock price interval. The company “Etablissement Maurel et Prom SA” will be used as an illustration. Its RIC-ticker is “MAUP.PA” and the monthly stock data is available from 1985 to the present. The cartel case in which this company was involved began on January 1, 1996, and lasted until December 31, 2000. On December 21, 2005, the European Commission made a decision regarding this case. Since all three dates fall within the range of available stock dates, this is a suitable example. Using this specified stock's RIC ticker symbol, Code-Snippet 20 collects financial information from Refinitiv and computes the monthly returns. The variables used in the code are `decision_date`, `start_cartel`, and `end_cartel`, which are Timestamp variables that represent the start and end dates of a cartel period. `RIC_ticker` is a string variable that represents the RIC ticker symbol for the stock. The “`ek.get_timeseries()`” function uses the stock's RIC ticker symbol, start date, end date, and desired data interval to obtain the stock's financial data from Refinitiv. The interval is currently set to “monthly” in this scenario. The index of the returned data is converted to a datetime format using the “`pd.to_datetime()`” function. The “`pct_change()`” function is then used to compute the stock's monthly returns, which are subsequently added as a new column to the “req” DataFrame. The “`dropna()`” function is then used to remove any rows that lack return values by setting the subset argument to “Returns”.

```
RIC_ticker = 'MAUP.PA'
start_cartel = pd.Timestamp('1996-01-01')
end_cartel = pd.Timestamp('2000-12-31')
decision_date = pd.Timestamp('2005-12-21') # New decision date

# Retrieve the Refinitiv data for the given RIC-Ticker
req = ek.get_timeseries([RIC_ticker], start_date = "1900-01-01", end_date = "2022-03-01",
interval="monthly")

# Convert index to datetime
req.index = pd.to_datetime(req.index)

# Calculate the stock returns
req['Returns'] = req['CLOSE'].pct_change()
req.dropna(subset=['Returns'], inplace=True)
Code-Snippet 20: Refinitiv gather financial data for MAUP.PA
```

After that the received data is used to obtain a first visual on both the stock closing price as well as the stock closing price returns by initially plotting the closing price. As illustrated in Figure 13 the line graph portrays the historical closing price of the “MAUP.PA” stock over time. The x-axis of the graph represents the duration in months (However only years are displayed), whereas the y-axis represents the stock price in USD. The graph exhibits the stock price as a fluctuating line over time, with some periods of growth and decline. The start date of the cartel case is denoted by a dotted line, while the end date is represented by a dashed line. An analysis of the graph, without considering seasonality, time-events, or other occasions, reveals that after a peak followed by a downward trend, the stock price appears to have been gradually increasing right before the cartel period. This suggests that the market had a positive outlook on the company's performance, which was reflected in the stock price. It is difficult to determine the exact impact of the cartel period on the stock price based solely on the graph, as there doesn't seem to be a clear change in the trend during this period. However, depending on the nature and duration of the cartel, it's possible that it had some impact on the stock price. After the cartel period, the stock price seems to be relatively stable, with minor fluctuations. This implies that the market had adjusted to the changes caused by the cartel and that the company was able to maintain a stable performance.

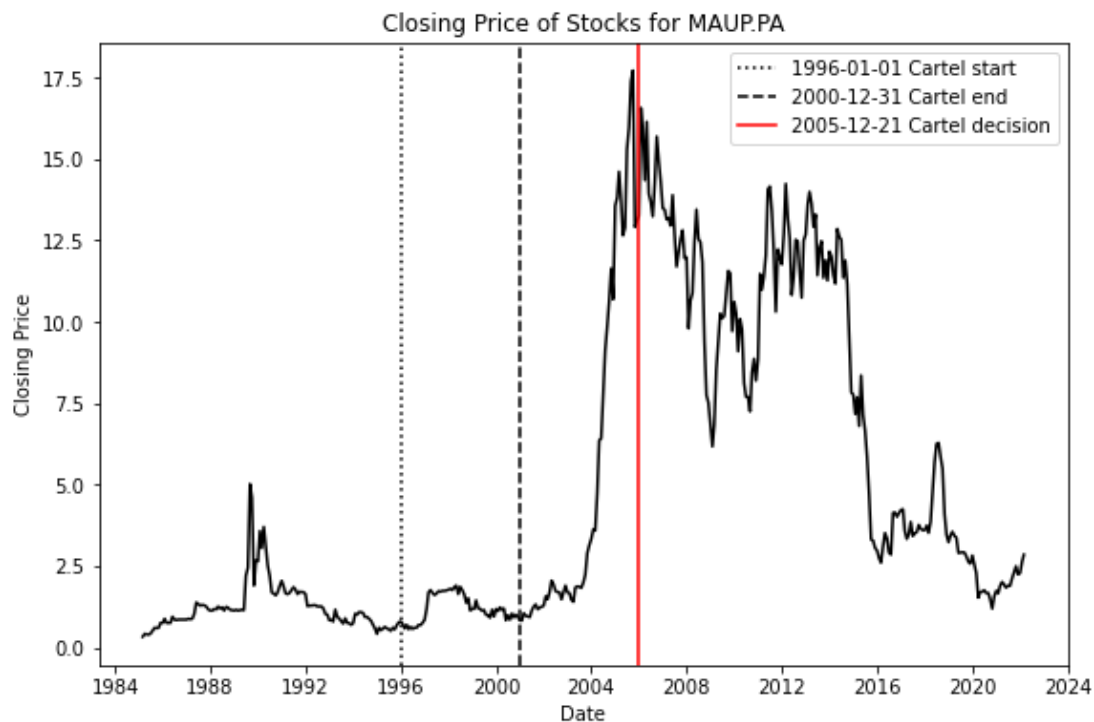


Figure 13: development of stock closing price for MAUP.PA including case dates

The stock price appears to be gradually increasing before the cartel decision date. Similar to before the cartel, this may indicate that the market had a positive outlook on the company and its performance. The stock price however experienced a sharp decline immediately after the cartel decision. This suggests that the market had a negative reaction to the news of the conviction, which likely had a significant impact on the company's performance and reputation. After the initial decline, the stock price appears to fluctuate around a lower average value, which may indicate that the market was adjusting to the new reality of the company's situation. Overall, the sharp decline in stock price suggests that the cartel decision had an impact on the company's performance and investor confidence. The same representation for the stock closing price returns can be found in the appendix A, however without an interpretation. The same strategy is now applied to loop the gathered data to receive a definite number of plots for companies whose start, end and decision date lay within the available stock price data received by Refinitiv. To do so the code executed in Code-Snippet 21 creates an empty DataFrame named "Out_of_scope" with columns "Ticker", "Start Cartel Date", "End Cartel Date", and "First Available Date" to store the data where the "cartel start_date" is not available in the stock data, indicating that it would not be possible to check if the cartel start had an impact on the stock price.

```

# Create an empty DataFrame to hold out-of-scope data
Out_of_scope = pd.DataFrame(columns=['Ticker', 'Start Cartel Date', 'End Cartel Date',
'First Available Date'])

# Define variables for the maximum number of plots to display and the current plot count
max_plots = 8
plot_count = 0
for i, row in new_df[["index", "Party Name(s)", "Matched company name", "ISIN num-
ber", "RIC", "Investigation date", "Decision date", "C_Start date", "C_End date"]].iterrows():
    try:
        RIC_ticker = row['RIC']
        start_cartel = row['C_Start date']
        end_cartel = row['C_End date']
        decision_date = row['Decision date']

        req = ek.get_timeseries([RIC_ticker], start_date="1900-01-01", end_date="2022-03-
01", interval="yearly")
        req.index = pd.to_datetime(req.index)

        # Check if either start_cartel or end_cartel is not available
        if pd.isna(start_cartel) or pd.isna(end_cartel):
            continue # Move on to the next row

        # Check if the first entry for this RIC is after the start_cartel date
        if req.index[0] > pd.to_datetime(start_cartel):
            # Add the relevant data to the Out_of_scope DataFrame
            Out_of_scope = Out_of_scope.append({'Ticker': RIC_ticker,
                                                'Start Cartel Date': start_cartel,
                                                'End Cartel Date': end_cartel,
                                                'First Available Date': req.index[0]}, ig-
nore_index=True)
            continue # Move on to the next row
        # Calculate the stock returns
        req['Returns'] = req['CLOSE'].pct_change()

```

Code-Snippet 21: Initial code for multi company search via the Refinitiv Eikon API

The next two lines of code create variables “max_plots” and “plot_count” that will be used later in the code. “max_plots” is set to 8, and “plot_count” is set to 0. Then the For-loop iterates over the rows in the “new_df” DataFrame, selecting only the columns “index”, “Party Name(s)”, “Matched company name”, “ISIN number”, “RIC”, “Investigation date”, “Decision date”, “C_Start date”, and “C_End date” using the “iterrows()” method. Inside the For-loop, the code tries to extract some data from the row. Specifically, it extracts the “RIC ticker”, “start_cartel date”, “end_cartel date”, and “decision date”, assigning them to variables with corresponding names. The code then uses the RIC ticker to query stock prices using the Eikon Data API (via get_timeseries())

function). It queries the stock prices with yearly interval from January 1st, 1900, to March 1st, 2022, and converts the index to datetime format. The resulting data is assigned to a new variable named “req”. Next, the code checks if either “start_cartel” or “end_cartel” is not available. If either is missing, the code skips to the next row using the “continue” statement because the data won’t be usable in this case. If both “start_cartel” and “end_cartel” are available, the code checks if the first entry for this RIC is after the “start_cartel date”. If it is, it adds the relevant data to the “Out_of_scope” DataFrame and continues to the next row using the “continue” statement. Furthermore Code-Snippet 21 also displays in the last line, that the code calculates the stock returns by taking the percentage change of the “CLOSE” price using the “pct_change()” method. Then, the code (not further displayed in the Code-Snippet) plots two graphs side by side. It first checks if the current plot count is less than the maximum number of plots to display and if the plot count is even (using the modulo operator). If both conditions are satisfied, it creates a new figure with two subplots using the “subplots()” function. It then adds a subplot for the stock price plot to the left-hand side plot. It plots the “CLOSE” prices against the datetime index, adds the “start_cartel”, “end_cartel”, and “decision date” as vertical lines, sets the plot title, xlabel, ylabel, and legend. It also adds an additional subplot for the returns plot with a secondary y-axis to the right-hand side plot, plots the “Returns” against the datetime index, sets the plot title, xlabel, ylabel, and legend again. The code generates plots that facilitate a side-by-side comparison of the stock price change and return for the same companies before and after the cartel period, as demonstrated in Figure 14. The plot shows that the stock prices and returns for MAERSKb.CO were relatively stable before the cartel period. However, during the cartel period, there appears to be an increase in stock prices and returns, which could be attributed to the collusion that led to higher prices and profits for the company. The stock prices and returns for MAERSKb.CO remained steady during the cartel period until the European Commission made its decision regarding the cartel. Subsequently, the stock prices and returns for the company declined initially, which could be attributed to fines or other penalties imposed as a result of the investigation. In general, the chart for MAERSKb.CO implies that the company encountered an increase in stock prices and returns during the cartel period, but this was followed by a decline once the cartel decision was made.

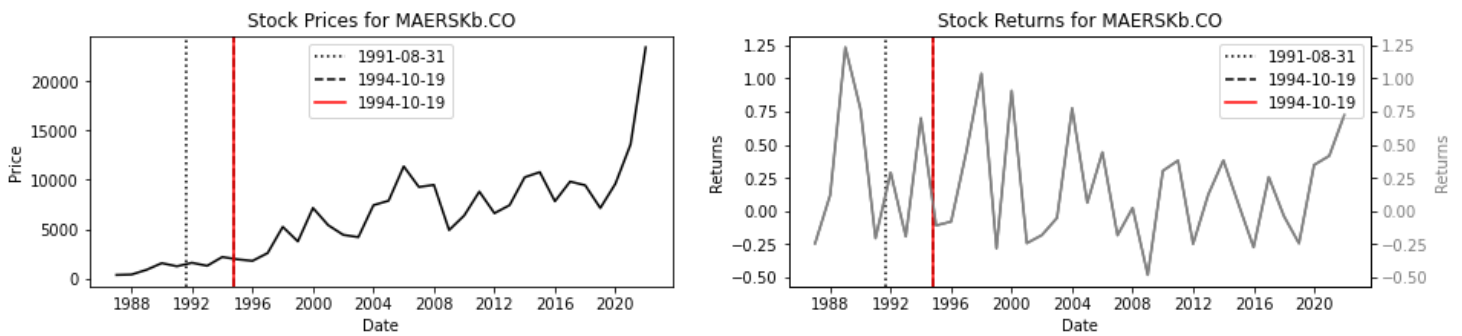


Figure 14: Stock Closing Price and Returns development before, during and after the illegal collusion for MAERSKb.CO

A noteworthy pair of plots was generated for the company with ticker 9107.T, as shown in Figure 15. Prior to the cartel, the plots indicate relatively stable stock prices and returns. However, after the cartel began, there was a noticeable decline in both stock prices and returns, likely due to increased competition and decreased profits resulting from the cartel's behaviour. During the cartel period, the stock prices and returns for 9107.T remained relatively stable, but with a downward trend that became more pronounced as the cartel end date approached. Following the end of the cartel, the stock price and returns decreased slightly before stabilizing. Interestingly, the stock price remained at a similar level even after the European Commission's decision regarding the existence of the cartel. Following the Commission's decision, there was a slight increase in both stock prices and returns for the company. Overall, the plots for “9107.T” suggest that the company experienced a decline in stock prices and returns during the cartel period, which can be attributed to increased competition and lower profits resulting from cartel behaviour.

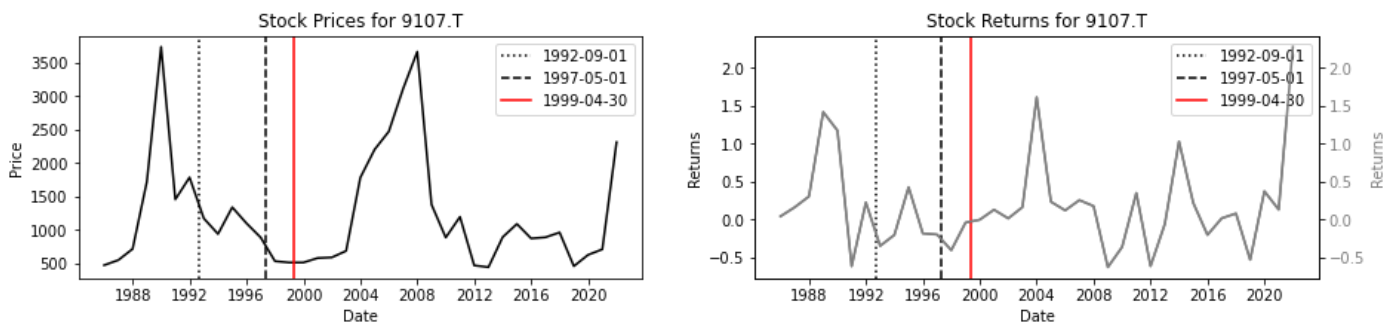


Figure 15: Stock Closing Price and Returns development before, during and after the illegal collusion for 9107.T

In order to not only plot the data, but also prepare it for further analysis, a similar approach to the plotting is utilized with some differences. As demonstrated in Code-Snippet 22, the

code reads data from the previously prepared excel file named “6. Orbis_export_with_RIC_Clean.xlsx” using the “read_excel()” function from the pandas library and assigns it to a DataFrame named “Base”.

```
# Read data from Orbis_export_with_RIC_Clean.xlsx file
Base = pd.read_excel('6. Orbis_export_with_RIC_Clean.xlsx', sheet_name='Sheet1')

ticker_dfs = {} # create an empty dictionary to store dataframes for each ticker

for i, row in Base.iterrows():
    decision_date = row['Decision date']
    start_cartel = row['C_Start date']
    end_cartel = row['C_End date']
    RIC_ticker = row['RIC']

    try:
        # read data from Eikon API using RIC ticker
        req = ek.get_timeseries([RIC_ticker], start_date="1980-01-01", end_date="2022-03-01", interval="daily")
        req.index = pd.to_datetime(req.index)

        # Create a DataFrame with all dates between 1980-01-01 and 2020-01-01
        all_dates = pd.date_range(start='1980-01-01', end='2020-01-01', freq='B')
        all_dates_df = pd.DataFrame(index=all_dates)

        # Merge the two DataFrames using a left join and fill missing values with NaN
        df = pd.merge(all_dates_df, req, left_index=True, right_index=True, how='left')
        df['CLOSE'].fillna(value=np.nan, inplace=True)

    [...]
```

Code-Snippet 22: First part of the data gathering for the Smart Data Analytics

Next, a dictionary called “ticker_dfs” is created to store DataFrames for each ticker. Similar to the previous approach, the For-loop iterates over each row in the Base DataFrame using the “iterrows()” method. Within the For-loop, the code extracts the values of “Decision date”, “C_Start date”, “C_End date”, and “RIC” from each row and assigns them to variables named “decision_date”, “start_cartel”, “end_cartel”, and “RIC_ticker”, respectively. To handle any exceptions that may arise while retrieving data from the Eikon API, a try-except block is utilized. Within the try block, data is retrieved again using the “get_timeseries()” method of the Eikon API and is assigned to a DataFrame called “req”. The index of the “req” DataFrame is then converted to a pandas Datetime Index using the “pd.to_datetime()” function. As the next step, a new DataFrame

called “all_dates_df” is created using the “pd.date_range()” method with the start and end dates of January 1, 1980 and January 1, 2020, respectively, and a frequency of business days (Monday-Friday). The “req” DataFrame is merged with the “all_dates_df” DataFrame using a left join with the “pd.merge()” function. The resulting DataFrame is assigned to a new variable named “df”. The “fillna()” method is then used to add “NaN” values to the “CLOSE” column of the “df” DataFrame to replace any missing items. It is specified that empty fields should be filled with “NaN” by setting the value argument to “np.nan”. The DataFrame “df” is being edited directly as opposed to being copied, as the “inplace” parameter is set to True.

After that the DataFrame must be first set up by making a new column for the “ticker” and giving it the RIC_ticker, as shown in Code-Snippet 23. After verifying whether the DataFrame index is lower than the “start_cartel date”, the “pre_cartel” column is created by converting the resulting Boolean array into integer values using the “astype()” method.

```
[...]
# Add the ticker column
df['ticker'] = RIC_ticker
# Add pre_cartel column
df['pre_cartel'] = (df.index < pd.to_datetime(start_cartel)).astype(int)
# Add post_cartel column
df['post_cartel'] = (df.index > pd.to_datetime(end_cartel)).astype(int)

# Add cartel column
cartel = ((df.index >= pd.to_datetime(start_cartel)) &
          (df.index <= pd.to_datetime(end_cartel))).astype(int)
df['cartel'] = cartel

# Drop unnecessary columns
df.drop(['HIGH', 'LOW', 'OPEN', 'VOLUME'], axis=1, inplace=True)
# Add the RETURN column
df['RETURN'] = df['CLOSE'].pct_change()
# Reorder columns
df = df[['ticker', 'CLOSE', 'RETURN', 'pre_cartel', 'cartel', 'post_cartel']]
# Add the dataframe to the dictionary
ticker_dfs[RIC_ticker] = df
```

Code-Snippet 23: Second part of the data gathering for the Smart Data Analytics

Similar to this, the “astype()” method is used to convert the resulting Boolean array into integer values for the “post_cartel” column after detecting whether the DataFrame index is greater than the “end_cartel date”. Following that, the “cartel” column is added by

evaluating whether the DataFrame index falls within the “start_cartel” and “end_cartel” date ranges, and then converting the resulting Boolean array into an “astype()” function. Following this, unnecessary columns such as “HIGH, LOW, OPEN, and VOLUME” are dropped from the “df” DataFrame using the “drop()” method with the “axis” parameter set to 1. A “RETURN” column is then added to the “df” DataFrame by calculating the percentage change of the “CLOSE” column using the “pct_change()” method. The column order in the “df” DataFrame is then rearranged using the syntax “df = df[['column1', 'column2', ...]]”. Finally, the resulting “df” DataFrame is added to the “ticker_dfs” dictionary with the “RIC_ticker” key. In case of an exception during the try block, the error variable is assigned to the exception object, and the continue statement is used to skip to the next iteration of the for loop.

The last step entails combining all the smaller DataFrames into a single DataFrame that has all the essential data after the relevant steps have been completed. The Code-Snippet 24 illustrates how the code originally uses the “pd.concat()” function to combine all of the DataFrames in the “ticker_dfs” into a single, sizable DataFrame called “big_df.”

```
# concatenate all dataframes into one big dataframe
big_df = pd.concat(ticker_dfs.values(), keys=ticker_dfs.keys(), names=['ticker', 'Date'])
big_df.reset_index(level='Date', inplace=True)

big_df.index.name = None
# Convert 'CLOSE' column to numeric values
big_df['CLOSE'] = pd.to_numeric(big_df['CLOSE'], errors='coerce')
big_df['RETURN'] = pd.to_numeric(big_df['RETURN'], errors='coerce')

chunk_size = 1000000
for i, chunk in enumerate(np.array_split(big_df, len(big_df)//chunk_size + 1)):
    filename = f'7. BIG_DF_{i}.xlsx'
    chunk.to_excel(filename, index=False)

big_df['CLOSE'] = big_df['CLOSE'].astype(float)
big_df['RETURN'] = big_df['RETURN'].astype(float)

# Create a MultiIndex for the panel data
big_df = big_df.set_index(['ticker', 'Date'])
```

Code-Snippet 24: Concatenate the gathered data into one DataFrame

Each original DataFrame in “ticker_dfs” is given a special identification number by the “keys” argument, and the “names” parameter specifies the names of the two levels of the

resulting MultiIndex. The “Date” level of the MultiIndex is then upgraded to a standard column using the “reset_index()” method. With the “index.name” attribute, the index name is set to “None”. Additionally, using the “to_numeric()” function, the “CLOSE” and “RETURN” columns are converted into numeric values. The For-loop is then used to divide the “big_df” into chunks using the “np.array_split()” method after setting the “chunk_size” variable to 1,000,000. The “to_excel()” method is then used to write each segment to the disk as an Excel file with a name that matches to the current iteration number. Finally, a MultiIndex is created using the “ticker” and “Date” columns in “big_df” using the “set_index()” method, and the “CLOSE” and “RETURN” columns in “big_df” are converted to float values using the “astype()” function. This concludes the Step for the data gathering and preparation for the Smart Data Analytics as a done.

5. Results

After the data gathering and preparation, the subsequent step involves applying regression models. This includes verifying whether the stock closing price data and residuals have a unit root and conducting a panel linear regression for the stock price that encompasses FixedEffects (abbreviated as FE in the rest of this thesis) and trend. A comparable approach is also implemented for stock price returns.

5.1 Fundamental data

The data analysis process begins with the use of the pandas “df.describe()” function to generate a summary (Table 6). The count variable specifies the number of data points available for each column, which is essential in determining the sample size and includes the dataset size and the amount of missing data. The counts range from (n = 324,176) for the closing prices to 1.4 million for the cartel and pre/post-cartel columns. The mean is a measure of central tendency that provides the average value for each column. Including the mean in the analysis is crucial since it gives an idea of the typical value for each variable. In this case, the mean of the “CLOSE” column is 2,071.126, while the “CLOSE” has a minimum of 0.01 and a maximum of 278,584.2. The mean of the “RETURN” column is 0.001. The standard deviation is a measure of the data's spread, indicating how much the data varies from the mean. A high standard deviation means that the data is more spread out, while a low standard deviation implies that the data is clustered closer to the mean. In this case, the standard deviation of the “CLOSE” column is 10,238.33, and the standard deviation of the “RETURN” column is 0.167.

Table 6 : Summary statistics for the used DataFrame

| Statistic | N | Mean | St. Dev. | Min | Max |
|-------------|-----------|-----------|------------|--------|-------------|
| CLOSE | 324,176 | 2,071.126 | 10,238.330 | 0.010 | 278,584.200 |
| RETURN | 359,558 | 0.001 | 0.167 | -0.997 | 99.000 |
| pre_cartel | 1,429,869 | 0.124 | 0.329 | 0 | 1 |
| cartel | 1,429,869 | 0.077 | 0.267 | 0 | 1 |
| post_cartel | 1,429,869 | 0.335 | 0.472 | 0 | 1 |

A section of the DataFrame itself is displayed in Figure 16. The DataFrame has a MultiIndex that consist of a combination of “ticker” and “Date”. For each Ticker the Date starts at the 1st of January 1980 and stacks up to the 1st of January 2020. Each row contains

values for the stock closing price and returns, if not available the values become “NaN” as well as an indication if the current line is either in the pre-cartel, cartel or post-cartel period. The trend column simply indicates the year only, increasing each year.

| Ticker | Date | Variables | | | | | |
|---------|------------|-----------|--------|------------|--------|-------------|-------|
| | | CLOSE | RETURN | pre_cartel | cartel | post_cartel | trend |
| NSTG.BE | 1980-01-01 | NaN | NaN | 0 | 0 | 1 | 1980 |
| | 1980-01-02 | NaN | NaN | 0 | 0 | 1 | 1980 |
| | 1980-01-03 | NaN | NaN | 0 | 0 | 1 | 1980 |
| | 1980-01-04 | NaN | NaN | 0 | 0 | 1 | 1980 |
| | 1980-01-07 | NaN | NaN | 0 | 0 | 1 | 1980 |

Figure 16: Section of the constructed DataFrame

5.2 Auto Regression Test for Residuals: Assessing Weak Stationarity

Performing an auto regression test on the residuals makes sense in a panel data context, as it helps to ensure that the residuals are stationary, which is a necessary assumption for the validity of many panel data models, including the PanelOLS model that is used in this thesis. A common method for testing the stationarity of time series based on autoregressive (AR) models is the “Augmented Dickey-Fuller (ADF) Test,” which is based on the null hypothesis that a time series contains a unit root, indicating that it is non-stationary. The ADF test uses an AR modelling approach to test for stationarity of the residuals in the time series. It examines whether the coefficients of the AR model are significantly different from zero, indicating that the time series is nonstationary. If the coefficients are not significant, it suggests that the time series is non-stationary and requires further modelling. Overall, autoregression can be used both to model the stationary dependence within time series models and to test for stationarity in time series data. The null hypothesis of the unit root test on the residuals is that the residuals have a unit root and are non-stationary, which is estimated by the use of the following regression model:

$$y_t = \rho * y_{t-1} + e_t$$

where y_t is the residuals at time t , ρ is the coefficient of the lagged residuals (should be less than 1 for stationary series), y_{t-1} is the value of the residuals at time $t-1$, and e_t is an

error term (independent and identically distributed) at time t . The alternative hypothesis is that the residuals do not have a unit root and therefore are stationary. In other words, if the p -value of the test is below a certain significance level (e.g., 0.05), then the null hypothesis can be rejected and conclude that the residuals are stationary. If the residuals are found to be non-stationary, then this can indicate that the model is miss specified and may require further attention such as additional control variables, transforming the data, or using a different model specification altogether. Figure 17 displays the results of the Unit Root test performed with the ADF-test (Augmented Dickey Fuller). The results of the ADF test on the residuals suggest that the null hypothesis of a unit root can be rejected at the 1% significance level. This provides evidence supporting the notion of weak stationarity, indicating that the residuals exhibit a constant mean, variance, and autocovariance structure over time. This means that there is evidence that the residuals are stationary over time and that they do not have a unit root. (OpenAI's ChatGPT AI Language Model, personal communication, 8 May 2023)

Unit Root Test on the Residuals

| Results | Values |
|------------------|---|
| ADF Test Results | (-19.815189549446927, 0.0, 132, 1429736, {'1%': -3.430354573789665, '5%': -2.861542021564094, '10%': -2.566771076004217}, 19221749.705890946) |

Figure 17: ADF-test results for the residuals

Based on the ADF test, we observe a large deviation from the null hypothesis as indicated by the test statistic value of -19.815, while the p -value of 0.0 provides strong evidence against the null hypothesis, indicating support for the alternative hypothesis of stationarity. The critical values at the 1%, 5%, and 10% significance levels are -3.430, -2.862, and -2.567, respectively. Since the test statistic value of -19.815 is significantly lower than the critical value at the 1% level, we can confidently reject the null hypothesis of a unit root. Therefore, based on these results, there is no evidence of autoregression in the residuals of the model. In conclusion, the results of the ADF test demonstrate that the residuals in the model are stationary over time, indicating no presence of a unit root. This

suggests that the model adequately captures the relationships among the variables and there is no need for further modelling to account for autocorrelation in the residuals. This is a desirable characteristic for time series data, suggesting that the model is not influenced by spurious trends or non-stationary behaviour (*OpenAI's ChatGPT AI Language Model*, personal communication, 8 May 2023).

5.3 Unit Root Test for Stock Closing Price

The ADF-test does not account for the fact that the dependent variable exhibits stationarity due to the panel data structure. Therefore, an alternative approach is employed to confirm the absence of a Unit-Root in the closing price. This is done by creating a lagged version of the closing price and then using PanelOLS from the linearmodels to estimate the regression model with fixed entity and time effects (FE) to test for the presence of a unit root in the closing price. It is then tested whether the autoregressive parameter (ρ) in the I(1) process is equal to 1 or not. If it is equal to $|1|$, then the process has a unit root and is non-stationary. If it is less than $|1|$, then the process is stationary. The regression model for the unit root process of order one I(1) with drift is:

$$y_t = \alpha_0 + \rho_1 * y_{t-1} + e_t$$

Where y_t is the variable of interest, which is in this case the closing price and α_0 is the drift term (similar to an intercept), ρ_1 is the coefficient of the lagged variable y_{t-1} which is in this case the closing price lag(1). The e_t displays as before the error term, which satisfies the assumption of being independent and identically distributed (i.i.d.). By estimating the coefficient ρ_1 using the PanelOLS model, it is possible to test whether the series is a unit root process of order one I(1) by checking if $|\rho_1| = 1$. If $|\rho_1| = 1$, then the series has a unit root and is not stationary, which leads to the following null hypothesis H_0 :

$$H_0: \rho = 1$$

The alternative hypothesis H_1 would occur if $|\rho_1| < 1$, then the series is stationary and does not have a unit root:

$$H_1: \rho < 1$$

Figure 18 displays the received PanelOLS Estimation Summary which shows that the coefficient for the lagged variable $CLOSE_{lag1}$ is estimated to be 0.9984 with a standard error of 8.62e-05. This suggests that the lagged variable is highly significant in explaining the variation in CLOSE. In this case, the coefficient of the lagged variable is 0.9984, which is very close to 1, so the process is non-stationary and has a unit root of order 1.

| PanelOLS Estimation Summary | | | | | | |
|-----------------------------|------------------|------------------------------|-------------|--|--|--|
| Dep. Variable: | CLOSE | R-squared: | 0.9977 | | | |
| Estimator: | PanelOLS | R-squared (Between): | 1.0000 | | | |
| No. Observations: | 313403 | R-squared (Within): | 0.9977 | | | |
| Date: | Tue, May 23 2023 | R-squared (Overall): | 0.9991 | | | |
| Time: | 20:18:35 | Log-likelihood | -2.23e+06 | | | |
| Cov. Estimator: | Unadjusted | F-statistic: | 4.471e+07 | | | |
| Entities: | 137 | P-value | 0.0000 | | | |
| Avg Obs: | 2287.6 | Distribution: | F(3,304944) | | | |
| Min Obs: | 0.0000 | F-statistic (robust): | 4.471e+07 | | | |
| Max Obs: | 2938.0 | P-value | 0.0000 | | | |
| Time periods: | 10436 | Distribution: | F(3,304944) | | | |
| Avg Obs: | 30.031 | | | | | |
| Min Obs: | 0.0000 | | | | | |
| Max Obs: | 130.00 | | | | | |

| | Parameter | Std. Err. | T-stat | P-value | Lower CI | Upper CI |
|-------------------------|-----------|-----------|-----------|---------|----------|----------|
| Intercept | 3.1276 | 15.212 | 0.2056 | 0.8371 | -26.687 | 32.942 |
| CLOSE_lag1 | 0.9984 | 8.62e-05 | 1.158e+04 | 0.0000 | 0.9982 | 0.9985 |
| cartel[T.1] | -0.1235 | 24.136 | -0.0051 | 0.9959 | -47.429 | 47.182 |
| post_cartel[T.1] | -0.0927 | 28.647 | -0.0032 | 0.9974 | -56.240 | 56.055 |

F-test for Poolability: 0.4964
P-value: 1.0000
Distribution: F(8455,304944)
Included effects: Entity, Time

Figure 18: PanelOLS Estimation Summary for the alternative Unit Root Test for the stock closing price

The presence of a unit root of order one in the closing price series means that the series is non-stationary. According to Wooldridge (2013), the unit root process implies that the closing price series may exhibit nonstationarity, where the regression coefficients can be biased and unreliable, resulting in the inability to obtain BLUE estimates (Best Linear Unbiased Estimators). In the case of the panel linear regression models, including fixed time or entity effects and trend in the model will not solve the problem of spurious regression. This is because the inclusion of these effects only accounts for differences across entities and time periods but does not address the fundamental issue of non-stationarity in the closing price series (Wooldridge, 2013). In summary, the non-

stationarity of the closing price series implies that the panel linear regression models for the closing price may result in incorrect test assumptions, and to accurately model the relationship between the variables of interest, it is necessary to transform the data to achieve stationarity. Therefore, the regressions for the stock price may be biased and declared as a non-finding, since a stationary process would be required for such an approach. One possible solution to address non-stationarity is to take the first differences of the variables. By applying first differences, the regressions for the stock price can mitigate bias and enable a more meaningful analysis, as it requires a stationary process. The regression models and its interpretation can still be found in the Appendix B, C, D and F.

5.4 Unit Root Test for Stock Closing Price Returns

Stock price returns are typically expressed as a “first difference” because they are calculated as the difference between the current stock price and the previous stock price, divided by the previous stock price. Expressing stock price returns as a first difference helps to remove any time-invariant factors that may affect the stock price, such as changes in the overall market conditions or macroeconomic trends. By taking the difference between two consecutive stock prices, we can eliminate the effects of these time-invariant factors and focus only on the changes that occur from one time period to the next. Although taking first differences can help to remove time-invariant factors and stabilize the variance of the series, it might eliminate the possibility of a unit root. In other words, the first difference of a series can still exhibit a unit root, indicating that the series is not stationary and has a long-term trend. After encountering that the stock closing price does have a unit root of order one and is therefore not stationary, the next step is to ensure that the stock price return is stationary. The unit root test provides an alternative approach to assess the presence of a unit root in the return series. According to Wooldridge (2013), the test involves estimating an autoregressive process of order one, commonly referred to as an “AR(1)” model for $\{y_t\}$. He furthermore states that this approach does not impose a unit root, but if one is present, $\hat{\rho}$ converges in probability to one as n gets large. However, $\hat{\rho}$ can be significantly different from one, particularly if the sample size is small (Wooldridge, 2013). The crucial assumption for weak dependence of an AR(1) process is the stability condition $|\rho_1| < 1$. Then, we say that $\{y_t\}$ is a stable AR(1) process. To follow the AR(1) approach, first, a lagged version of the return is created and then a

PanelOLS model from the linearmodels is used to estimate the regression model with FE to test for the presence of a unit root in the return. As before the model tests whether the autoregressive parameter (ρ) in the AR(1) process is equal to 1 or not. If it is equal to 1, then the process has a unit root and is non-stationary. If it is less than 1, then the process is stationary. The regression model for the unit root process of order one I(1) with drift is:

$$\Delta y_t = \alpha_0 + \rho_1 * \Delta y_{t-1} + e_t$$

Where y_t is the variable of interest, which is in this case the stock price return and α_0 is the drift term. ρ_1 is the coefficient of the lagged variable y_{t-1} which is in this case the return lag1. The e_t displays as before the error term. By estimating the coefficient ρ_1 using the PanelOLS model, it is possible to test whether the series is a unit root process of order one I(1) by checking if $|\rho_1| = 1$. If $|\rho_1| < 1$, then the series is stationary and does not have a unit root. If $|\rho_1| = 1$, then the series has a unit root and is not stationary. According to the PanelOLS Estimation Summary displayed in Figure 19, the coefficient estimate for RETURN_lag1 is -0.0004, which is less than 1 in absolute value. Therefore, based on this model, it can be concluded that the series is stationary and does not have a unit root of order 1.

| PanelOLS Estimation Summary | | | | | | |
|-----------------------------|------------------|-----------------------|-------------|---------|----------|----------|
| Dep. Variable: | RETURN | R-squared: | 1.998e-07 | | | |
| Estimator: | PanelOLS | R-squared (Between): | -0.0501 | | | |
| No. Observations: | 359422 | R-squared (Within): | 9.566e-09 | | | |
| Date: | Tue, May 23 2023 | R-squared (Overall): | -1.106e-05 | | | |
| Time: | 19:48:34 | Log-likelihood | 1.565e+05 | | | |
| Cov. Estimator: | Unadjusted | F-statistic: | 0.0234 | | | |
| Entities: | 137 | P-value | 0.9952 | | | |
| Avg Obs: | 2623.5 | Distribution: | F(3,350677) | | | |
| Min Obs: | 0.0000 | F-statistic (robust): | 0.0234 | | | |
| Max Obs: | 8607.0 | P-value | 0.9952 | | | |
| Time periods: | 10437 | Distribution: | F(3,350677) | | | |
| Avg Obs: | 34.437 | | | | | |
| Min Obs: | 0.0000 | | | | | |
| Max Obs: | 136.00 | | | | | |
| | Parameter | Std. Err. | T-stat | P-value | Lower CI | Upper CI |
| Intercept | 0.0003 | 0.0033 | 0.0940 | 0.9251 | -0.0062 | 0.0068 |
| RETURN_lag1 | -0.0004 | 0.0017 | -0.2223 | 0.8241 | -0.0037 | 0.0029 |
| cartel[T.1] | -0.0001 | 0.0086 | -0.0141 | 0.9888 | -0.0170 | 0.0167 |
| post_cartel[T.1] | 0.0006 | 0.0062 | 0.0991 | 0.9210 | -0.0115 | 0.0128 |

F-test for Poolability: 5.7295

P-value: 0.0000

Distribution: F(8741,350677)

Included effects: Entity, Time

Figure 19: PanelOLS Estimation Summary for the Unit Root Test for the stock closing price return

5.5 Regression for Stock Price Returns including FixedEffects

After controlling for non-stationarity, we employ a first difference estimator to examine the relationship between stock price returns and the variables of interest (cartel and post-cartel) since the returns are indeed stationary. By including FE for entity and time, one can control for unobserved heterogeneity and time-invariant confounders that may affect the outcome variable. Important to note is that we are unable to quantify the impact of the primary explanatory variable on y using FE if it remains constant across time. FE is mechanically the same as allowing a different intercept for each cross-sectional unit (Wooldridge, 2013). To do so we first specify the linear regression model formula to help isolating the effects of the independent variables of interest (cartel and post_cartel) on the dependent variable (stock price returns), while holding constant any other factors that may influence the outcome:

$$RETURN_{it} = \beta_0 + \beta_1 cartel_{it} + \beta_2 post_cartel_{it} + \alpha_i + \delta_t + e_{it}$$

$RETURN_{it}$ represents the return for a specific stock at a specific time and is the dependent variable in this formula. As in the previous formula β_0 represents the intercept and β_1 and β_2 are the coefficients for cartel and post_cartel, respectively. To control for unobserved heterogeneity and time-invariance α_i represents individual FE for the entity effects while δ_t represents time FE, as well e_{it} represents the error term for the model. It is important to note that the “pre_cartel” variable is considered the base or reference level in this context.

The execution of the panel linear regression for stock price returns including FE leads us to the PanelOLS Estimation Summary displayed in Figure 20. The PanelOLS Estimation Summary provides important information about the regression model that has been estimated. The R-squared value shows how well the model fits the data. In this case, the R-squared value is very small, indicating that the independent variables in the model explain very little of the variation in the dependent variable. Looking at the parameter estimates, the intercept is very small and statistically insignificant. The coefficient for the “cartel” variable is negative but statistically insignificant. The coefficient for the “post_cartel” variable is positive but as the other two before statistically insignificant. These results suggest that neither “cartel” nor “post_cartel” have a significant impact on stock price returns. Since the p-values for “cartel” and “post_cartel” are large, we cannot

reject the null hypothesis that there is no relationship between these variables and the dependent variable. Therefore, based on this model, it appears that “cartel” and “post_cartel” do not have a significant impact on stock price returns.

| PanelOLS Estimation Summary | | | | | | |
|-----------------------------|------------------|------------------------------|-------------|--|--|--|
| Dep. Variable: | RETURN | R-squared: | 5.955e-08 | | | |
| Estimator: | PanelOLS | R-squared (Between): | -0.0499 | | | |
| No. Observations: | 359558 | R-squared (Within): | 4.212e-08 | | | |
| Date: | Tue, May 23 2023 | R-squared (Overall): | -1.098e-05 | | | |
| Time: | 19:57:47 | Log-likelihood | 1.567e+05 | | | |
| Cov. Estimator: | Unadjusted | F-statistic: | 0.0104 | | | |
| Entities: | 136 | P-value | 0.9896 | | | |
| Avg Obs: | 2643.8 | Distribution: | F(2,350813) | | | |
| Min Obs: | 444.00 | F-statistic (robust): | 0.0104 | | | |
| Max Obs: | 8608.0 | P-value | 0.9896 | | | |
| Time periods: | 8608 | Distribution: | F(2,350813) | | | |
| Avg Obs: | 41.770 | | | | | |
| Min Obs: | 1.0000 | | | | | |
| Max Obs: | 136.00 | | | | | |

| | Parameter | Std. Err. | T-stat | P-value | Lower CI | Upper CI |
|--------------------|-----------|-----------|---------|---------|----------|----------|
| Intercept | 0.0003 | 0.0033 | 0.0929 | 0.9259 | -0.0062 | 0.0068 |
| cartel | -0.0001 | 0.0086 | -0.0136 | 0.9892 | -0.0170 | 0.0167 |
| post_cartel | 0.0006 | 0.0062 | 0.1001 | 0.9203 | -0.0115 | 0.0128 |

F-test for Poolability: 5.7310
 P-value: 0.0000
 Distribution: F(8742,350813)
 Included effects: Entity, Time

Figure 20: PanelOLS Estimation Summary for Stock Price Returns including FE

5.6 Regression for Stock Price Returns including FixedEffects and trend

By adding a trend variable that counts up the years, it is possible to account for any overall upward or downward trend in stock prices over time. This is a common technique used in time series analysis to account for the influence of time. Including entity effects also allows to control for any differences in the stock price returns between different entities, such as companies or industries. Including a time trend variable in the model captures the linear relationship between the dependent variable (in this case, stock price returns) and time, which is the role of time effects. Therefore, the TimeEffects aren't included this time. To execute this panel linear regression the model formula is:

$$RETURN_{it} = \beta_0 + \beta_1 cartel_{it} + \beta_2 post_cartel_{it} + \beta_3 trend_t + \alpha_i + e_{it}$$

In this case $RETURN_{it}$ is the stock return for ticker i in year t , while β_0 displays the intercept. $\beta_1 cartel_{it}$ is a dummy variable equal to 1 if firm i belongs to a cartel in year t and 0 otherwise. $\beta_2 post_cartel_{it}$ is as well a dummy variable equal to 1 if firm i belongs to a cartel in year t , and 0 otherwise as well. $\beta_3 trend_t$ is the time trend variable that captures any systematic change in returns over time and α_i includes the entity FE to capture firm-specific characteristics that are constant over time. As usual e_{it} is the error term or disturbance.

The panel linear regression analysis for stock closing price returns, including FE and trend, was conducted, and the results are presented in Figure 21.

| PanelOLS Estimation Summary | | | | | | |
|-----------------------------|------------------|------------------------------|-------------|--|--|--|
| Dep. Variable: | RETURN | R-squared: | 3.094e-06 | | | |
| Estimator: | PanelOLS | R-squared (Between): | -0.2179 | | | |
| No. Observations: | 359558 | R-squared (Within): | 3.094e-06 | | | |
| Date: | Tue, May 23 2023 | R-squared (Overall): | -4.284e-05 | | | |
| Time: | 20:00:24 | Log-likelihood | 1.327e+05 | | | |
| Cov. Estimator: | Unadjusted | F-statistic: | 0.3707 | | | |
| Entities: | 136 | P-value | 0.7742 | | | |
| Avg Obs: | 2643.8 | Distribution: | F(3,359419) | | | |
| Min Obs: | 444.00 | F-statistic (robust): | 0.3707 | | | |
| Max Obs: | 8608.0 | P-value | 0.7742 | | | |
| Time periods: | 8608 | Distribution: | F(3,359419) | | | |
| Avg Obs: | 41.770 | | | | | |
| Min Obs: | 1.0000 | | | | | |
| Max Obs: | 136.00 | | | | | |

| | Parameter | Std. Err. | T-stat | P-value | Lower CI | Upper CI |
|--------------------|------------|-----------|---------|---------|----------|-----------|
| Intercept | 0.1736 | 0.1657 | 1.0480 | 0.2946 | -0.1511 | 0.4983 |
| cartel | 0.0001 | 0.0068 | 0.0196 | 0.9843 | -0.0132 | 0.0134 |
| post_cartel | 0.0017 | 0.0052 | 0.3286 | 0.7425 | -0.0085 | 0.0119 |
| trend | -8.634e-05 | 8.254e-05 | -1.0460 | 0.2956 | -0.0002 | 7.544e-05 |

F-test for Poolability: 0.6978

P-value: 0.9970

Distribution: F(135,359419)

Included effects: Entity

Figure 21: PanelOLS Estimation Summary for Stock Price Returns including EntityEffects and trend

The R-squared value indicates that the independent variables, namely cartel, post_cartel, and trend, have an insignificant effect on the variation of stock price returns. The F-statistic also suggests that the overall model is not statistically significant, with a p-value of 0.77. Furthermore, none of the individual coefficients are statistically significant, with p-values exceeding the conventional threshold of 0.05. Therefore, it can be concluded that the variables cartel, post_cartel, and trend have no significant impact on stock price returns. The intercept term, which reflects the average stock price return when all other variables are constant, is also not statistically significant with a p-value of 0.2946. Additionally, the panel data regression model incorporates entity effects to capture unobserved differences between the firms. The F-test for poolability is employed to determine whether the entity-specific effects are equal to zero. The p-value of 0.9970 suggests that we cannot reject the null hypothesis of poolability, indicating that no significant differences exist between the entities that are not accounted for in the model. Based on these findings, it can be concluded that the variables cartel, post_cartel, and trend have no statistically significant effect on stock price returns.

6. Conclusion

6.1 Impact of illegal collusion on stock price and stock price returns

The regression approach used to investigate the impact of illegal collusion on stock prices has limitations due to the non-stationarity of the closing price, which has a unit root of order 1. Therefore, the regression results, including the t-test, R-squared, and F-test, are all biased. This concludes that the analysis, which showed that cartels have a negative impact on stock prices, as evidenced by the statistically significant negative coefficients for the cartel and post-cartel dummy variables in the regression model are biased as well and do not display the truth. Those regression models are therefore placed in the appendix.

The other purpose of this analysis is to investigate whether illegal collusion has an effect on stock price returns. Two regression analyses were conducted to examine the relationship between stock price returns and the variables of interest, namely “pre_cartel”, “cartel” and “post_cartel”, while taking into account unobserved heterogeneity and time-invariant confounders. The first regression analysis employed a panel linear regression with FE. The findings indicate that the variables “pre_cartel”, “cartel” and “post_cartel” do not have a statistically significant impact on stock price returns. The variables had large p-values, which suggests that the null hypothesis cannot be rejected. The coefficient for “cartel” was negative, while that for “post_cartel” was positive, but neither coefficient was statistically significant. These results suggest that illegal collusion does not have a significant impact on stock price returns, according to this model. To account for broad trends in stock price movements over time, a trend variable was included in the second regression analysis. In line with the findings of the initial research, the variables “pre_cartel”, “cartel”, “post_cartel”, and trend do not have a statistically significant effect on stock price returns. All of the coefficients' p-values were higher than the usual cutoff point of 0.05, suggesting that they were not statistically significant. The F-statistic further indicates that the model as a whole lacks statistical significance. In order to take into consideration unobserved differences across the firms, entity effects were incorporated into the panel data regression model utilized in this study. The F-test for poolability was used to determine whether the entity-specific effects were equal to zero. The p-value of 0.9970 indicates that the entity-specific effects are equal to zero and thus the null hypothesis of poolability cannot be disproved.

In summary, based on the regression analyses, it appears that illegal collusion does not have a statistically significant impact on stock price returns. The coefficients for “cartel” and “post_cartel” were not statistically significant, and the p-values for all the coefficients exceeded the conventional threshold of 0.05 and therefore one of the research questions “*Does Illegal Collusion Determine Stock Prices Returns?*” can be answered with No; Illegal collusion does not determine stock price returns. It is worth noting, however, that these findings are contingent upon the particular model and variables employed in the analysis, and further research may be warranted to explore any alternative links between illegal collusion and stock price returns. The second research question “*Does Illegal Collusion Determine Stock Prices?*” could have not been answered with the used data, tools and methods due to the presence of a unit root of order 1.

6.2 Limitation

As I conducted my research on cartels and their impact on stock prices, I faced several limitations due to the size and completeness of the data available. Specifically, the size of the provided data was limited and incomplete for the whole topic of cartels, as they are often not uncovered and remain in the shadows. Initially, I had 1735 rows of companies to begin with, but after gathering data for those companies, only 230 companies had a RIC, meaning I could only search for financial data on Refinitiv for those 230 companies. Furthermore, a portion of those companies were not usable because the cartel start or end date was not within the data I received from Refinitiv. Additionally, Refinitiv only provides financial data starting around 1980, which meant that essentially another ~ 170 companies were incomplete. For those companies, the analysis could still be done; however, it might be missing dates within the range of the cartel start. This limited the sample size to around 60 companies that included all necessary data and 170 that lacked important information. This is questionable in terms of statistical relevance, which is another limitation of my research. Moreover, not all information was given for the 1735 companies provided in the data. For some companies, no country was given, or no cartel start or end date nor a decision date. For some dates, parts like day or month were missing, making it challenging to obtain complete information. It's important to note that the limitations of the sample size and incomplete data are common issues in conducting research on cartels. Many cartels are not even uncovered, leading to a big part not taken

into account. These limitations affect the statistical significance of the findings and the generalizability of the results.

While this thesis analysed the impact of illegal collusion on stock prices and stock price returns, it's worth noting that there are multiple other tests and regressions that could have been performed on the cartel data. For instance, one could have analysed the number of cartel repeat offenders, the OECD sector, the country, the report route, whether it was a leniency case or not, or the nature of the formal decision in general. However, given the limited sample size and the focus of this thesis, it was decided to analyse only the impact of illegal collusion on stock prices and stock price returns. While this specific analysis provides valuable insights, it's important to acknowledge that there are other factors that may influence the impact of cartels on the market.

6.3 Discussion & recommendation

The executed analysis discusses mainly the results of two panel linear regression analyses conducted to examine the relationship between stock price returns and the variables cartel and post-cartel, which indicate whether a company was involved in a cartel or not. The first analysis for the stock price returns included FE for entity and time to control for unobserved heterogeneity and time-invariant confounders. The regression model shows that neither variable has a significant impact on stock price returns. The second analysis adds a trend variable that counts up the years to capture any overall upward or downward trend in stock prices over time. Including entity effects also allows for controlling any differences in the stock price returns between different entities, such as companies or industries. As mentioned before the regression model indicates that the variables cartel, post-cartel, and trend have no significant impact on stock price returns. Additionally, the panel data regression model incorporates entity effects to capture unobserved differences between firms. The F-test for poolability is employed to determine whether the entity-specific effects are equal to zero. The p-value suggests that no significant differences exist between the entities that are not accounted for in the model. Therefore, it can be concluded that the variables cartel, post-cartel, and trend have no significant impact on stock price returns. Although there are no significant findings on the correlation between illegal collusion and stock price returns itself, the scientific value has been added by proving that the stock closing price has a unit root and therefore the applied model cannot be used to estimate the correlation between illegal collusion and the stock closing price. It is

advisable to explore alternative approaches to determine whether there is a correlation between illegal collusion and stock prices, given the limitations of the current analysis. It should be noted that a significant portion of the data provided is historical and not readily available on current financial systems, which may have resulted in a smaller sample size. As more instances of illegal collusion are detected in the future, with access to financial data, similar methodologies can be employed to provide statistical insights with a larger sample size, shedding further light on this topic. Furthermore, there are multiple aspects on the topic of illegal collusion that have not been considered in this thesis, as described in the limitations. Based on those limitations outlined, there are several recommendations for further analysis that could be considered. Firstly, expanding the sample size beyond the current 60 companies that have all necessary data could increase the statistical relevance and generalizability of the results. This could involve exploring additional data sources beyond Refinitiv or using more comprehensive databases to gather information on the companies in the sample. Additionally, it may be beneficial to analyse the impact of cartels on other financial variables beyond just stock prices and returns, such as trading volume or volatility. Furthermore, as suggested in the thesis, there are various other tests and regressions that could be performed on the cartel data, such as analysing the number of repeat offenders or the impact of the nature of the formal decision. These additional analyses could provide a more comprehensive understanding of the impact of cartels on the market. Finally, it may also be worth exploring the impact of cartels in different countries and industries to gain a more complete understanding of the topic. Overall, while the current analysis provides valuable insights, further analysis could help to overcome the limitations outlined and improve the statistical significance and generalizability of the results.

Bibliography

Bloomberg Launches Online Request Utility and New Mapping Tools for the Financial Instrument Global Identifier (FIGI) | Press | Bloomberg LP. (n.d.). Retrieved 11 April 2023, from <https://www.bloomberg.com/company/press/bloomberg-launches-online-request-utility-and-new-mapping-tools-for-the-financial-instrument-global-identifier-figi/>

Bloomberg L.P. | About, Careers, Products, Contacts. (n.d.). Bloomberg L.P. Retrieved 7 February 2023, from <https://www.bloomberg.com/company/>

Bruneckienė, J., Pekarskienė, I., Guzavičius, A., Palekienė, O., & Šovienė, J. (2015). *The Impact of Cartels on National Economy and Competitiveness: A Lithuanian Case Study* (1st ed. 2015). Springer International Publishing: Imprint: Springer. <https://doi.org/10.1007/978-3-319-17287-3>

Carree, M., Günster, A., & Schinkel, M. P. (2010). European Antitrust Policy 1957–2004: An Analysis of Commission Decisions. *Review of Industrial Organization*, 36(2), 97–131. <https://doi.org/10.1007/s11151-010-9237-9>

Combe, E., Monnier, C., & Legal, R. (2007). Cartels: The Probability of Getting Caught in the European Union. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.1015061>

Company structures & ownership information | Bureau van Dijk. (n.d.). Retrieved 11 April 2023, from https://www.bvdinfo.com/en-gb/about-us?gclid=CjwKCAjwitShBhA6EiWAq3RqAxZMUWS_vSUDH6hvN_CnVGrtk9amqM5WycLceY5VX2GmfZozMxRRxoCtrUQAvD_BwE

Dresch, A., Lacerda, D. P., & Antunes Jr., J. A. V. (2014). *Design science and design science research*. Springer.

Goyvaerts, J., & Levithan, S. (2009). *Regular expressions cookbook* (1st ed). O'Reilly.

Graafland, J., & Verbruggen, H. (2022). Free-Market, Perfect Market and Welfare State Perspectives on “Good” Markets: An Empirical Test. *Applied Research in Quality of Life*, 17(2), 1113–1136. <https://doi.org/10.1007/s11482-021-09946-2>

Hevner, March, Park, & Ram. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75. <https://doi.org/10.2307/25148625>

Jaspers, J. D. (2017). Managing Cartels: How Cartel Participants Create Stability in the Absence of law. *European Journal on Criminal Policy and Research*, 23(3), 319–335. <https://doi.org/10.1007/s10610-016-9329-7>

Johannesson, P., & Perjons, E. (2014). *An introduction to design science*. Springer.

MIFID II MASTER RIC USER GUIDE. (2018).

MyRefinitiv. (2023). Use of Reuters Instrument Codes (RICs).
<https://my.refinitiv.com/content/mytr/en/policies/use-of-reuters-instruments-codes.html>

OpenAI's ChatGPT AI language model. (2023, May 8). [Personal communication].

OpenFIGI API | OpenFIGI. (n.d.). Retrieved 11 April 2023, from
<https://www.openfigi.com/api>

Provost, F., & Fawcett, T. (2013). Data science for business: What you need to know about data mining and data-analytic thinking (1. ed., 2. release). O'Reilly.

Refinitiv- About us. (n.d.). Retrieved 7 February 2023, from
<https://www.refinitiv.com/en/about-us>

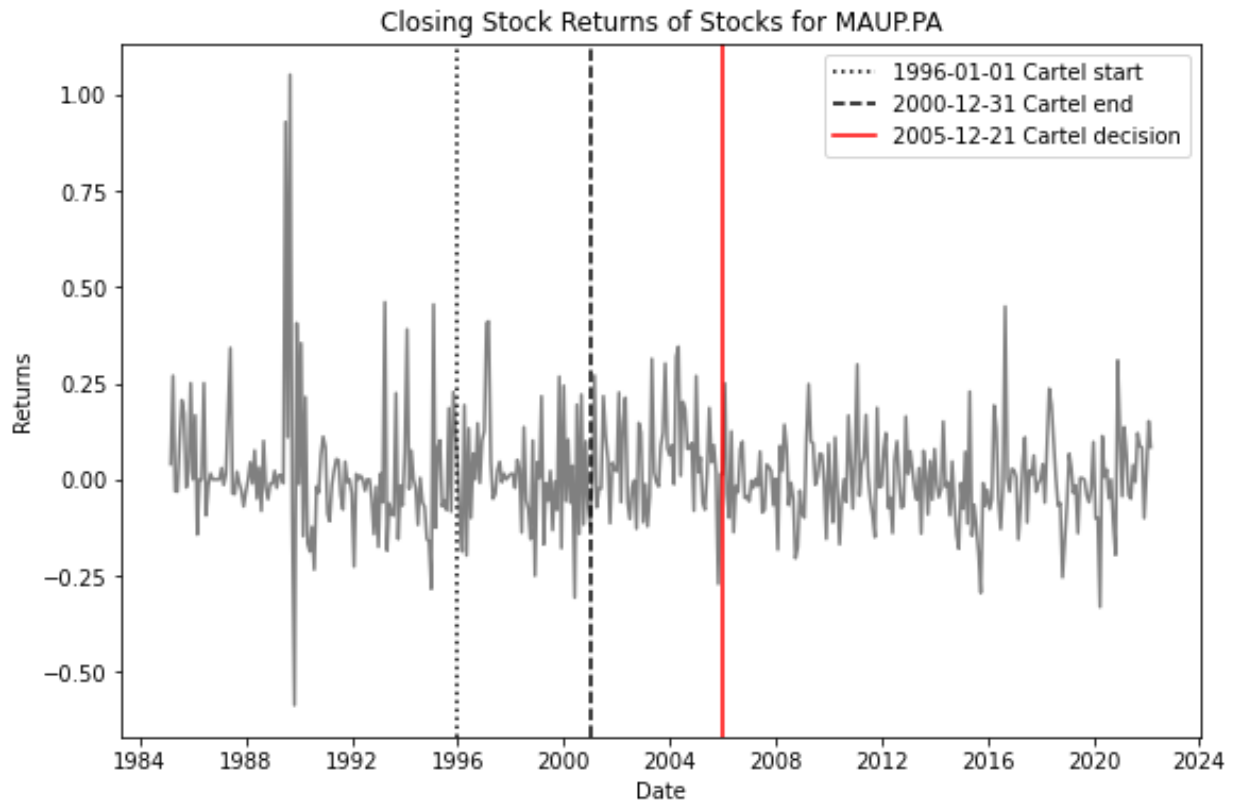
Stubblebine, T. (2007). Regular expression pocket reference (2nd ed). O'Reilly.

Wooldridge, J. M. (2013). Introductory econometrics: A modern approach (5th ed). South-Western Cengage Learning.

Appendix

A. Illustration of the stock closing price returns for the stock “MAUP.PA”

This statistic includes the cartel case start, end, and decision date



B. Panel Linear Regression for Stock price including Unit Root on residuals (Without fixed effects)

```

=====
PanelOLS Estimation Summary
=====
Dep. Variable:          CLOSE      R-squared:                3.534e-05
Estimator:             PanelOLS   R-squared (Between):      4.664e-05
No. Observations:     1429869   R-squared (Within):       2.534e-05
Date:                 Fri, May 05 2023   R-squared (Overall):      3.534e-05
Time:                 20:38:13      Log-likelihood            -1.472e+07
Cov. Estimator:       Unadjusted

                          F-statistic:                25.269
Entities:              137      P-value                  0.0000
Avg Obs:               1.044e+04   Distribution:             F(2,1429866)
Min Obs:               1.044e+04
Max Obs:               1.044e+04   F-statistic (robust):    25.269
                          P-value                  0.0000
Time periods:         10437   Distribution:             F(2,1429866)
Avg Obs:               137.00
Min Obs:               137.00
Max Obs:               137.00

```

```

=====
Parameter Estimates
=====

```

| | Parameter | Std. Err. | T-stat | P-value | Lower CI | Upper CI |
|-------------|-----------|-----------|---------|---------|----------|----------|
| Intercept | 1804.3 | 7.8151 | 230.87 | 0.0000 | 1789.0 | 1819.6 |
| cartel | -123.40 | 22.913 | -5.3856 | 0.0000 | -168.31 | -78.489 |
| post_cartel | 44.574 | 12.976 | 3.4351 | 0.0006 | 19.142 | 70.006 |

```

=====

```

```

=====
Unit Root Test on the residuals
=====
ADF Test Results: (-13.37636181013743, 5.079021097659124e-25, 132, 1429736, {'1%': -3.43035457378966
5, '5%': -2.861542021564094, '10%': -2.566771076004217}, 19084559.59970811)

```

C. Regression for Stock price including fixed effects (Biased)

$$close_{it} = \beta_0 + \beta_1 cartel_{it} + \beta_2 post_cartel_{it} + \alpha_i + \delta_t + e_{it}$$

Where $close_{it}$ is the closing price for individual at time t , $cartel_{it}$ is the binary dummy variable indicating whether individual I was part of a cartel at time t , $post_cartel_{it}$ is the binary dummy variable indicating whether the individual I was part of the post-cartel period at time t , α_i as the individual fixed effect for individual I , δ_t is the time fixed effect for time t and e_{it} is the error term for individual i at time t .

```

PanelOLS Estimation Summary
=====
Dep. Variable:          CLOSE    R-squared:                3.506e-05
Estimator:              PanelOLS  R-squared (Between):      -0.0002
No. Observations:      1429869  R-squared (Within):       -8.763e-05
Date:                   Fri, May 05 2023  R-squared (Overall):      -0.0001
Time:                   18:56:20  Log-likelihood             -1.427e+07
Cov. Estimator:        Unadjusted

                               F-statistic:                24.884
Entities:               137      P-value                    0.0000
Avg Obs:                1.044e+04  Distribution:              F(2,1419294)
Min Obs:                1.044e+04
Max Obs:                1.044e+04  F-statistic (robust):     24.884
                               P-value                    0.0000

Time periods:          10437  Distribution:              F(2,1419294)
Avg Obs:               137.00
Min Obs:               137.00
Max Obs:               137.00

```

```

Parameter Estimates
=====
                Parameter  Std. Err.    T-stat    P-value    Lower CI    Upper CI
-----
Intercept      1863.8      8.9301     208.71    0.0000     1846.3     1881.3
cartel         -91.281     22.772     -4.0085    0.0001     -135.91    -46.649
post_cartel   -140.76     19.959     -7.0524    0.0000     -179.88    -101.64
=====

```

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

F-test for Poolability: 119.41

P-value: 0.0000

Distribution: F(10572,1419294)

Included effects: Entity, Time

The P-values in the “Parameter Estimates” table in the figure above represent the statistical significance of each independent variable in the model. The P-values for both “cartel” and “post_cartel” are less than 0.05, which suggests that both variables have a statistically significant effect on the stock price (CLOSE). Specifically, the coefficient of “cartel” is -91.281, which indicates that a one-unit increase in “cartel” is associated with a decrease in CLOSE by \$91.281 and similarly, the coefficient of “post_cartel” is -140.76, which indicates that a one-unit increase in “post_cartel” is associated with a decrease in CLOSE by \$140.76, for each holding all other variables constant. The intercept has a coefficient of 1863.8, which represents the expected value of the dependent variable when all independent variables are zero. Overall, the model suggests that firms that are part of a cartel have lower stock prices compared to firms that are not part of a cartel, and that stock prices tend to decrease further after the cartel period ends. However, the model does not explain a significant amount of the variation in the stock prices. It is also important

to keep in mind any limitations or potential issues with the data and model assumptions, such as potential omitted variables, endogeneity, or heteroscedasticity.

D. Regression for Stock price including fixed effects and the trend (Biased)

$$close_{it} = \beta_0 + \beta_1 cartel_{it} + \beta_2 post_cartel_{it} + \beta_3 trend_t + \alpha_i + e_{it}$$

Where $close_{it}$ is the stock price of company i at time t and β_0 is the intercept, which represents the expected value of the dependent variable, which is the closing price in this case, when all of the independent variables ($cartel$, $post_cartel$, and $trend$) are equal to zero. The dummy variables $\beta_1 cartel_{it}$ and $\beta_2 post_cartel_{it}$ equal to 1 if the observation is either in a cartel period or post-cartel period for firm i at time t , and otherwise 0.

```

=====
                        PanelOLS Estimation Summary
=====
Dep. Variable:                CLOSE      R-squared:                0.0003
Estimator:                    PanelOLS  R-squared (Between):     -0.0002
No. Observations:            1429869  R-squared (Within):      0.0003
Date:                        Fri, May 05 2023  R-squared (Overall):     4.508e-05
Time:                        20:36:21    Log-likelihood           -1.427e+07
Cov. Estimator:              Unadjusted

                               F-statistic:                128.13
Entities:                    137      P-value                  0.0000
Avg Obs:                     1.044e+04  Distribution:             F(3,1429729)
Min Obs:                     1.044e+04
Max Obs:                     1.044e+04  F-statistic (robust):    128.13
                               P-value                  0.0000
Time periods:                10437    Distribution:             F(3,1429729)
Avg Obs:                     137.00
Min Obs:                     137.00
Max Obs:                     137.00

```

```

=====
                        Parameter Estimates
=====
                Parameter  Std. Err.    T-stat    P-value    Lower CI    Upper CI
-----
Intercept    -1.527e+04    923.83    -16.528    0.0000    -1.708e+04    -1.346e+04
cartel        -154.25     21.441     -7.1942    0.0000     -196.28     -112.23
post_cartel  -162.68     19.196     -8.4745    0.0000     -200.30     -125.05
trend         8.5748      0.4640     18.481     0.0000      7.6654      9.4842

```

```

F-test for Poolability: 9317.1
P-value: 0.0000
Distribution: F(136,1429729)

```

```

Included effects: Entity

```

The Panel OLS estimation summary in the figure above shows that the regression model has an R-squared value of 0.0003, indicating that only a small percentage of the variation in the dependent variable (CLOSE) can be explained by the independent variables (cartel, post_cartel, and trend). The P-value of the F-statistic for the regression model is 0.0000, indicating that the model is statistically significant. The estimated coefficient for the cartel variable is -154.25, with a standard error of 21.441 and a corresponding P-value of 0.0000, indicating that the variable is statistically significant in explaining the variation in the dependent variable. Similarly, the post_cartel variable has a coefficient of -162.68, with a standard error of 19.196 and a corresponding P-value of 0.0000, indicating that it is also statistically significant in explaining the variation in the dependent variable. The trend variable has a coefficient of 8.5748, with a standard error of 0.4640 and a corresponding P-value of 0.0000, indicating that it is statistically significant in explaining the variation in the dependent variable. The F-test for poolability has a P-value of 0.0000, indicating that there is significant heterogeneity across entities. The model includes Entity effects as included effects.

The parameter estimates show that cartel and post_cartel have a negative effect on the stock prices. This means that when there is a cartel or post-cartel period, the stock prices tend to decrease. In contrast, the trend variable has a positive effect on the stock prices, indicating that over time, stock prices tend to increase. Overall, the model suggests that the cartel and post-cartel periods have a negative impact on the stock prices, while the trend has a positive impact. However, the model's overall explanatory power is quite low, and it may be worth considering additional variables to improve the model's fit. However due to non-stationarity of the closing price, this analysis is biased and should not be further used.

E. Panel Linear Regression for Stock Price Returns including Unit Root on residuals (Without fixed effects) (Biased)

| PanelOLS Estimation Summary | | | |
|-----------------------------|------------------|-----------------------|-------------|
| Dep. Variable: | RETURN | R-squared: | 4.377e-06 |
| Estimator: | PanelOLS | R-squared (Between): | 0.0034 |
| No. Observations: | 359561 | R-squared (Within): | -1.138e-07 |
| Date: | Fri, May 05 2023 | R-squared (Overall): | 4.377e-06 |
| Time: | 20:50:24 | Log-likelihood | 1.327e+05 |
| Cov. Estimator: | Unadjusted | | |
| | | F-statistic: | 0.7868 |
| Entities: | 136 | P-value | 0.4553 |
| Avg Obs: | 2643.8 | Distribution: | F(2,359558) |
| Min Obs: | 444.00 | | |
| Max Obs: | 8608.0 | F-statistic (robust): | 0.7868 |
| | | P-value | 0.4553 |
| Time periods: | 8608 | Distribution: | F(2,359558) |
| Avg Obs: | 41.771 | | |
| Min Obs: | 1.0000 | | |
| Max Obs: | 136.00 | | |

| Parameter Estimates | | | | | | |
|---------------------|-----------|-----------|---------|---------|----------|----------|
| | Parameter | Std. Err. | T-stat | P-value | Lower CI | Upper CI |
| Intercept | 0.0010 | 0.0004 | 2.4710 | 0.0135 | 0.0002 | 0.0018 |
| cartel | -0.0013 | 0.0046 | -0.2760 | 0.7826 | -0.0103 | 0.0078 |
| post_cartel | -0.0007 | 0.0006 | -1.2390 | 0.2153 | -0.0018 | 0.0004 |

| Unit Root Test | |
|-------------------|---|
| ADF Test Results: | (-64.75030969770404, 0.0, 84, 359476, {'1%': -3.4303681913304263, '5%': -2.8615480403470013, '10%': -2.566774279584212}, -265226.75767781236) |

F. Regression with different parameter (Biased)

$$CLOSE_{it} = \beta_0 + \beta_1 cartel_{it} + \alpha_i + \delta_t + e_{it}$$

$$CLOSE_{it} = \beta_0 + \beta_1 precartel_{it} + \alpha_i + \delta_t + e_{it}$$

$$CLOSE_{it} = \beta_0 + \beta_1 postcartel_{it} + \alpha_i + \delta_t + e_{it}$$

In all three cases the $CLOSE_{it}$ represents the closing price for a specific stock i at a specific time t and is the dependent variable in this formula. While β_0 is the intercept or constant term in all three formulas, each β_1 is assigned as one of the three continuous variables $cartel$, $precartel$ or $postcartel$ signifying if the company/individual is before, during or after a cartel period. In all three cases, the regression includes entity fixed effects as α_i and time fixed effects δ_t , which control for any unobserved heterogeneity across entities

(i.e., companies) and over time, respectively. As well e_{it} is for all three formulas the error term.

The attached figure displays the results of a panel linear regression analysis for stock closing prices considering three different parameters with fixed factors. Note that the figure only includes parameter estimates. Although not displayed in the figure, the R-squared values for all three PanelOLS Estimation summaries are very low, suggesting that the model does not explain much of the variation in the data. The R-squared value for the first summary, which includes only the “Cartel” variable, is 5.054e-09. The R-squared value for the second summary, which includes only the “pre_cartel” variable, is 1.341e-09. Finally, the R-squared value for the third summary, which includes only the “post_cartel” variable, is 1.542e-08. Regarding the parameter estimates displayed in the attached figure, the coefficient estimate for the “cartel” variable in the first summary is 16.428 with a p-value of 0.9681. The coefficient estimate for the “pre_cartel” variable is 8.2642 with a p-value of 0.9836. For the third summary, the coefficient estimate for the “post_cartel” variable is 0.0004, with a p-value of 0.9444. The p-values associated with the cartel and pre_cartel variables are both greater than 0.05, indicating insufficient evidence to reject the null hypothesis that these variables have no effect on the dependent variable. Similarly, the p-value associated with the post_cartel variable is also greater than 0.05, indicating insufficient evidence to reject the null hypothesis that this variable has no effect on the dependent variable.

In summary, the parameter estimates suggest that the independent variables (cartel, pre_cartel, and post_cartel) may not be statistically significant in explaining the dependent variable, which is the stock closing price. Nonetheless, the low p-value for the F-test suggests that there may be other differences between the models that warrant further investigation. However due to non-stationarity of the closing price, this analysis is biased and should not be further used.

Cartel Parameter Estimates

| | Parameter | Std. Err. | T-stat | P-value | Lower CI | Upper CI |
|-----------|-----------|-----------|--------|---------|----------|----------|
| Intercept | 2071.1 | 11.269 | 183.78 | 0.0000 | 2049.0 | 2093.2 |
| cartel | 16.428 | 411.36 | 0.0399 | 0.9681 | -789.82 | 822.68 |

F-test for Poolability: 60.932

P-value: 0.0000

Distribution: F(8599,315575)

Pre-cartel Parameter Estimates

| | Parameter | Std. Err. | T-stat | P-value | Lower CI | Upper CI |
|------------|-----------|-----------|--------|---------|----------|----------|
| Intercept | 2071.1 | 11.210 | 184.76 | 0.0000 | 2049.1 | 2093.1 |
| pre_cartel | 8.2642 | 401.73 | 0.0206 | 0.9836 | -779.12 | 795.64 |

F-test for Poolability: 60.936

P-value: 0.0000

Distribution: F(8599,315575)

Post-cartel Parameter Estimates

| | Parameter | Std. Err. | T-stat | P-value | Lower CI | Upper CI |
|-------------|-----------|-----------|---------|---------|----------|----------|
| Intercept | 2088.5 | 249.72 | 8.3634 | 0.0000 | 1599.1 | 2578.0 |
| post_cartel | -32.813 | 470.40 | -0.0698 | 0.9444 | -954.79 | 889.16 |

F-test for Poolability: 60.857

P-value: 0.0000

Distribution: F(8599,315575)