UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
FACULDADE DE CIÊNCIAS ECONÔMICAS
CURSO DE CIÊNCIAS ECONÔMICAS

PEDRO SOFIATI DE SÁ

DO POLITICAL NEWS AFFECT FINANCIAL REAL ESTATE MARKETS?

Porto Alegre
2022

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
FACULDADE DE CIÊNCIAS ECONÔMICAS
CURSO DE CIÊNCIAS ECONÔMICAS

PEDRO SOFIATI DE SÁ

DO POLITICAL NEWS AFFECT FINANCIAL REAL ESTATE MARKETS?

Work presented in partial fulfillment of the requirements for the degree of Bachelor in Economics.

Advisor: Prof. Dr. Nelson Seixas dos Santos

Porto Alegre
2022

CIP - Catalogação na Publicação

Sofiati de Sá, Pedro
Do political news affect financial real estate markets? / Pedro Sofiati de Sá. -- 2022.
67 f.
Orientador: Nelson Seixas dos Santos.

Trabalho de conclusão de curso (Graduação) -- Universidade Federal do Rio Grande do Sul, Faculdade de Ciências Econômicas, Curso de Ciências Econômicas, Porto Alegre, BR-RS, 2022.

1. Political events. 2. Real estate market. 3. Efficient markets. 4. Granger causality. I. Seixas dos Santos, Nelson, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

PEDRO SOFIATI DE SÁ

DO POLITICAL NEWS AFFECT FINANCIAL REAL ESTATE MARKETS?

Work presented in partial fulfillment of the requirements for the degree of Bachelor in Economics.

Aprovada em: Porto Alegre, 6 de outubro de 2021.

BANCA EXAMINADORA:

Prof. Dr. Nelson Seixas dos Santos – Orientador

UFRGS

Prof. Dr. Ronald Otto Hillbrecht

UFRGS

Prof. Dr. Carlos Schönerwald

UFRGS

ACKNOWLEDGMENTS

To my parents Carmen and Otavio, my close friends and Gabriel who helped me with coding and refactoring.

RESUMO

Este artigo investiga se as notícias políticas afetam o mercado financeiro imobiliário brasileiro. Para isso, foram desenvolvidos dois softwares. Primeiro, um software para extrair, transformar e carregar os bancos de dados. Em seguida, um software para realizar a exploração dos dados e aplicar um método não paramétrico usando o teste de causalidade de Granger. Concluiu-se que o volume de notícias políticas Granger não causa volatilidade nos retornos imobiliários no período de janeiro de 2011 a maio de 2022.

Palavras-chave: Eventos políticos. Mercado Imobiliário. Mercados eficiêmtes. Causalidade de Granger.

ABSTRACT

This paper investigates whether political news affect the Brazilian financial real estate market. Two softwares were developed to do so. First, a software to extract, transform and load the databases. Then, a software to perform data exploration and apply a non-parametric method using the Granger-causality test. It was concluded that the volume of political news does not Granger-cause the real estate returns volatility in the period ranging from January 2011 to May 2022.

Keywords: Political events. real estate markets. efficient markets. granger causality.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The Brazilian political landscape is known for its political scandals and unprecedented events. It is often implied in the media that political events impact all markets volatility and cause abnormal returns, which is in disagreement with the efficient market hypothesis presented by Fama (1970).

The literature relating to the empirical analysis of efficient markets hypothesis (EMH) is relatively extensive and often presents contradictory results, which demonstrates how fertile this research line still is. Marques and Santos (2016) concluded that Brazilian financial markets show characteristics of a semi-strong form of market efficiency, considering that very few political events (mostly related to elections) actually had an impact. This conclusion is in accordance with Smales (2015) that during the Australian electoral cycle volatility of equity and bond options increased. On the other hand, in (GABRIEL; RIBEIRO; RIBEIRO, 2013) the conclusion was that the "white-line" market had abnormal returns before and after an event of to tax reduction, which indicates indicates a rejection of the semi-strong efficiency hypothesis for the analysed market.

In this paper, it was aimed to test whether the Brazilian real estate market is affected by political news. To do so, two softwares were developed. The first software is responsible for the extraction, transformation and loading of all data used in the second software, which is responsible for data exploration and the application a non-parametric approach. The non-parametric approach fits the data in 21-day windows and applies the Granger-causality test to determine whether the volume of political news Granger-causes the real estate market volatility in the analysed period.

This paper is organized as follows. Section 2 summarizes Brazilian political institutions and the financial the real estate market system. Section 3 is a literature review. Section 4 regards data exploration of all databases used in this paper. Section 5 summarizes the empirical strategy used. Section 6 includes a description of all methods cited in the empirical strategy. Section 7 presents the results after the application of described methodology. Section 8 has the conclusions related to the results and the development of this paper. Section 9 is the Appendix which contains both softwares cited during the paper and tables related to the databases.

2 INSTITUTIONAL ENVIRONMENT

2.1 Brazilian Institutions and Financial Markets

Brazil is a federative republic with a presidential system (Brazil, 2016) that relies on the existence of three independent and harmonious powers, that is, there is autonomy between them and there is no hierarchy. The three branches are: Legislative, Executive and Judiciary. The president is elected to a four-year administration and can only be re-elected once in a row. The ministers and the president of the Central Bank are chosen by the president of the republic, however, there is no independence from the Central Bank. Therefore, its president must be approved by the senate and can be dismissed if decided.

The National Financial System (SFN) is composed of several entities and institutions whose objective is to promote financial intermediation, that is, the connection between assignors and creditors. Through the SFN, people (individuals and legal entities) and the government conduct a large part of its assets for various purposes. Table 2.1 below shows the composition of the SFN.

Table 2.1: Brazilian Financial System

Normative Institutions	**Supervising Institutions**	**Operating Institutions**		
National Monetary Council (CMN)	Central Bank of Brazil (BCB)	Banks	Credit cooperatives	Payment institutions
	Securities and Exchange Commission (CVM)	Consortium administrators	Brokerage and distribution companies	Other non-banking institutions
National Council for Private Insurance (CNSP)	Private Insurance Superintendence (SUSEP)	Insurance and reinsurance Companies	Open pension entities	Capitalization companies
National Council for Complementary Pension (CNPC)	National Complementary Pension Superintendence (PREVIC)	Closed complementary social security entities (pension funds)		

Source: elaborated by the author.

The guidelines that govern the financial system are present in Law n° 1.079 (BRASIL, 1950). The Normative institutions, formed by the National Monetary Council (CMN), the National Council for Private Insurange (CNSP) and the National Council for Complementary Pension, determine general rules for the proper functioning of the SFN. CMN is in charge of guiding the application of resources and promoting the improvement of financial institutions and financial instruments, as well as the coordination of monetary policies , credit, budgetary, fiscal and public debt (external or internal).

The Supervising institutions work to ensure that citizens and members of the fi-

nancial system follow the rules defined by regulatory bodies. These institutions are the Central Bank of Brazil (BCB), the Securities and Exchange Commission (CVM), the Private Insurance Superintendence (SUSEP) and the National Complementary Pension Superintendence (PREVIC).

Operating institutions deal directly with the public, in the role of financial intermediary.

2.2 Real Estate Market in Brazil

The Brazilian real estate market is the center of activities related to civil construction, as it is responsible for allotment activities, purchase, sale, lease, among others activities that guide the construction process. These activities have as a common objective the construction of an immovable property, which is the product marketed in the real estate market. In all cities, it provides growth for the local or regional economy, due to the large volume of direct or indirect employment generated by its added services.

The regulation of the Brazilian real estate market is enacted in Law nº 4.591 (BRASIL, 1964) in which the Housing Financing System (SFH) was created. The SFH intended to encourage and promote the construction and home ownership in Brazil. SFH had in its composition the National Housing Bank (BNH), which was responsible for managing the resources that were collected by the Service Time Guarantee Fund (FGTS) and savings accounts, financing housing projects, establishing terms, interest and payment terms.

The Real Estate Financing System (SFI) is established in 1997 by Law nº 9514 (BRASIL, 1997) and improved in 2004 by Law n° 10.931 (BRASIL, 2004) with the objectives to promote real estate financing and encourage new housing developments, since it allows real estate credits to reach the financial market with lesser bureaucracy. In the SFI, at the discretion of the CMN, agents such as: savings banks, real estate credit companies, savings and loan associations, companies mortgage companies and other entities. Agents can apply resources to the SFI through the following instruments: Real Estate Credit Note (CCI), Certificate of Real Estate Receivables (CRI) and Letter of Real Estate Credit (LCI).

CCI represents a real estate credit that transforms a private contract into a negotiable security, in which debtor undertakes to pay a real estate debt to a creditor. This type of security can be remunerated at a fixed or floating rate.

CRIs are securities backed by CCIs and, in practice, serve to provide funds for agents with real estate developments in progress. A securitization company transforms the debt of the entrepreneurial agent through the issuance of CRIs that will be acquired by investors.

LCIs are also backed by CCIs, but may have as collateral mortgages, fiduciary alienation and, unlike CRIs, as they are issued by banks, guarantee of the Credit Guarantee Fund of R$ 250,000.00 per citizen and per financial institution broadcaster. In this type of investment, investors lend money to the bank that transfers the capital to agents in need of financing.

3 MARKET EFFICIENCY

3.1 Efficient Markets Hypothesis

The benchmark for market informational efficiency is found in (FAMA, 1970). The study concerns the analysis of theory, hypotheses and tests on capital markets efficiency and their forms. The conception of market efficiency, especially regarding public information such as political events and news, is a cornerstone to other studies on the subject of this paper. As defined by the author:

> THE PRIMARY ROLE of the capital market is allocation of ownership of the economy's capital stock. In general terms, the ideal is a market in which prices provide accurate signals for resource allocation: that is, a market in which firms can make production-investment decisions, and investors can choose among the securities that represent ownership of firms' activities under the assumption that security prices at any time "fully reflect" all available in- formation. A market in which prices always "fully reflect" available information is called "efficient. (FAMA, 1970, p.2)

Fama (1970) runs through the theoretical assumptions of each form of efficiency (weak, semi-strong, and strong) and its evidence. In the weak form, the subset of information of interest is just the history of past prices (or returns). The semi-strong form also considers all publicly available information and testing is often concerned with speed of price adjustment. Finally, the strong form considers private information that investors or groups have monopolistic access. The central focus is that, in an efficient market, information must be fully reflected in prices, considering that the market operates in "fair game". An important test is presented in (SIDNEY, 1964) in which a trading rule is tested on price indices from 1897 to 1959, whose conclusion is that the application of the tested trading strategy does not win buy and hold and that the results are consistent with the random walk hypothesis. A random walk is a stochastic (or random) process and its hypothesis is that extremely important for financial theory and especially for the definition of efficient markets. Fama explains that the approach of traders and academics in relation to market behavior and forecasts is fundamentally different and irreconcilable:

> In sum the theory of random walks in stock market prices presents important challenges to both the chartist and the proponent of fundamental analysis. For the chartist, the challenge is straightforward. If the random walk model is a valid description of reality, the work of the chartist, like that of the astrologer, is of no real value in stock market analysis. The empirical evidence to date provides strong support for the random walk model. [...] If the random walk theory is valid and if security exchanges are "efficient" markets, then stock prices at any point in time will represent good estimates of intrinsic or fundamental values. Thus, additional fundamental analysis is of value only when the analyst has new information which was not fully considered in forming

> current market prices, or has new insights concerning the effects of generally available information which are not already implicit in current prices. If the analyst has neither better insights nor new information, he may as well forget about fundamental analysis and choose securities by some random selection procedure. (FAMA, 1965, p.2)

3.2 Semi-strong efficiency tests

The most relevant approaches are in relation to the methodology used in the semi-strong form tests. The methodology presented by the author is essentially the same which is still used today, widely used in the Event Study methodology. The approach chosen for the test is based on the following market model suggested in (MARKOWITZ, 1959):

$$\tilde{r}_{j,t+1} = \alpha_j + \beta_j \tilde{r}_{M,t+1} + \tilde{u}_{j,t+1} \tag{3.1}$$

$r_{j,t+1}$ is the rate of return on security j per month t. $r_{M,t+1}$ is the return on the index of market given by M. β_j are parameters that may vary according to the security, and $\tilde{u}_{j,t+1}$ is a random error. The author assumes that if a stock split is associated with an abnormal behavior, this would be reflected in the estimated regression residuals for the months close to the deployment. For each split, a m is defined as the month the split actually took place, a $m + 1$ being the next month after the split and $m - 1$ as the month before splitting. Therefore, the abnormal return for all splits in month m is defined as:

$$u_m = \sum_{j=1}^{N} \frac{\hat{u}_{jm}}{N}, \tag{3.2}$$

$\hat{u}_{jm}$ is the regression residue for security j in month m and N is number of ramifications. Then, the cumulative average abnormal return is defined as:

$$U_m = \sum_{k=-29}^{m} u_k. \tag{3.3}$$

The average abnormal return, in the context of the paper, can be interpreted as

the mean deviation of returns after stock splits in relation to the normal return, and the abnormal return average cumulative can be interpreted as the cumulative deviation.

3.3 Recent studies and findings

The literature relating to the empirical analysis of Efficient Markets Hypothesis (EMH) is relatively extensive specially on emerging countries with political uncertainty. A fact that makes this research line rather interesting is the contradictory results from different studies, which shows how open to invention and development it still is. Another aspect to be observed is the different methodological approaches. Some studies apply the Event Study methodology, as originally described in (FAMA, 1970) and further developed in (MACKINLAY, 1997). Some studies like (MARQUES; SANTOS, 2016), which will be further detailed below, implement a methodology that consists of applying a GARCH filter to gather the volatility of the series to later apply parametric and non-parametric approaches to identify abnormal periods of return.

Kumara and Fernando (2020) investigated the impact of uncertain political events on the daily index return on the Colombo Stock Exchange (CSE) in Sri Lanka. The authors selected 15 major events and compared its impact for 2, 7, 15, and 30 days before and after the event using the mean-adjusted return model. After analysing the pre and post abnormal returns, the events were categorized as good, bad or uncategorized (both good and bad influence). Summing up, only four out of the 15 selected events have significant impacts on the market. Among these there's the end of a civil war and the end of two presidential elections. The authors concluded that CSE is inefficient to capture noisy information because it takes around a month to return to its original position (which is in disagreement with EMH).

Parveen (2021) applied the event study methodology to analyse the reactions from the stock market dynamics to information in political events considering the impact of result announcement of the Lok Sabha Elections 2019 on the Indian Stock market. In opposition to the last study cited, the author concludes that the announcement of the election results didn't create market volatility in the observed period and average abnormal returns seemed to be random. Thus, the author concluded that the stock market studied can be considered semi-strong efficient.

(MARQUES; SANTOS, 2016) had similar conclusions with a different methodology. The methodology consists of applying a GARCH filter to a sample of some Brazilian

stock market index returns and short-term interest rates to identify abnormal volatility periods. Later the authors associate those periods with political events using a parametric and a non-parametric method. The conclusions are similar to (SMALES, 2015), which are that in periods of election there's an increase in market volatility. The adopted approaches associated only 12 dates with abnormal returns, which, considering the Brazilian political landscape, is a rather small volume. Therefore, the results point to the acceptance of the semi-strong efficient market hypothesis.

A study by Gabriel, Ribeiro and Ribeiro (2013) also uses samples from the Brazilian stock market but specifically stock prices of companies that belong to segment of "white line" during an event of tax reductions. In opposition to the conclusions of (MARQUES; SANTOS, 2016), the authors concluded that the market had abnormal returns before the event, which indicates information leak and a market imperfection. Thus, it was possible to reject, in this specific market, the semi-strong efficiency hypothesis.

Considering the recent events related to the Covid-19 ongoing global pandemic, it was expected to identify its impacts in the databases used in this paper. Caldas et al. (2021) investigates the effects of Covid-19 on the performance of the shares of B3´s sectors and concludes that most felt a hard impact of the disease which caused six circuit breakers and reflected on changes in their average of returns and traded quantities. The conclusion was that the market showed characteristics of weak efficiency.

4 DATA

4.1 Imob Index Price

The Imob Index, which can be found on the B3 page, was built based on a theoretical portfolio of real assets. Its objective is to be an indicator of the average performance of the quotations of the assets with greater representation in the Brazilian real estate and civil construction sector. A table that contains each sector, asset (which includes common stocks), quantity and representation in the theoretical portfolio can be found in section 9, which is the appendix. Figure 4.1 below demonstrates the Imob Index Price series (Imob) between January 2011 and May 2022.

Figure 4.1: Imob Index Price series 2011-2022

Source: elaborated by the author using data from (B3, 2022).

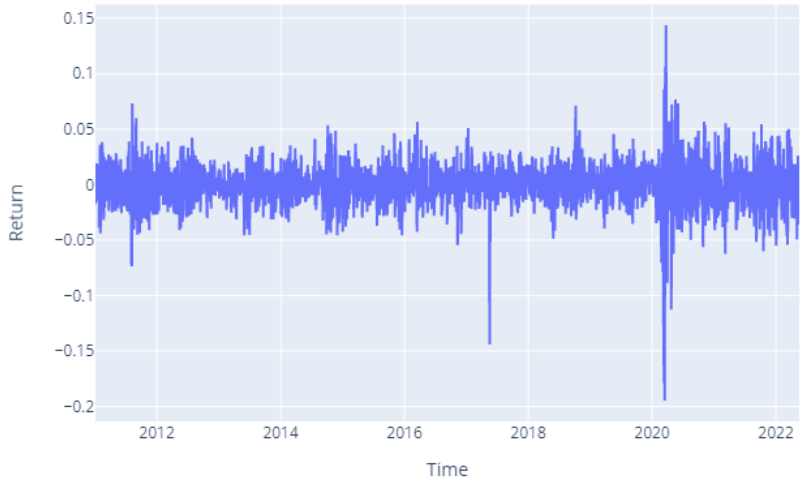
The behavior of the series demonstrates very important recent historical facts, such as the recession of the Brazilian economy between 2015 and 2016 and the Covid crisis in early 2020. To calculate the returns of the series, Equation 4.1 was used, as shown below:

$$Imob_r t = \ln(\frac{Imob_t}{Imob_{t-1}}) \tag{4.1}$$

Figure 4.2, as seen below, represents the calculated series of returns. The observed behaviour is of a stationary form. Another important observation is that the variance through the series is rather lean, which indicates the unusual form of its distribution. The period with noticeable increase in variance happens in 2020, which was expected

considering the behavior in the price series.

Figure 4.2: $Imob_r$ series 2011-2022



Source: elaborated by the author.

Two tests were used to determine the stationarity of the series: the ADF (Augmented Dickey-Fuller) test (1979) and the KPSS (Kwiatkowski–Phillips–Schmidt–Shin) test (1992). The test uses the following null and alternative hypotheses:

- Null Hypothesis (H0): a unit root is present in $Imob_r$.
- Alternative Hypothesis (HA): a unit root is not present in $Imob_r$. Thus, the time series is stationary.

Table 4.1: ADF Critical Values ($Imob_r$)

	Value
statistic	-19.200653
p-value	0.000000
1pct	-3.432682
5pct	-2.862570
10pct	-2.567318

Source: elaborated by the author.

The results of the ADF test present in Table 4.1 indicate the rejection of the null hypothesis at 5% significance. Thus, by the ADF test, one can infer the non-existence of a unit root in $Imob_r$ during the analyzed period. The KPSS test was also applied to test stationarity. The test uses the following null and alternative hypotheses:

- Null Hypothesis (H0): a unit root is not present in $Imob_r$. Thus, the time series is stationary.
- Alternative Hypothesis (HA): a unit root is present in $Imob_r$.

The results of the KPSS test (shown in the Table 4.2) allows the same inference, as we accept the null hypothesis, that is, that the series does not have a unit root and is stationary.

Table 4.2: KPSS Critical Values ($Imob_r$)

	Value
statistic	0.058778
p-value	0.1000000
1pct	0.216000
5pct	0.146000
10pct	0.119000

Source: elaborated by the author.

Table 4.3 below contains the descriptive statistics of both Imob and $Imob_r$ series:

Table 4.3: Descriptive Statistics

Statistic	N	Mean	St. Dev.	Min	Max
Imob	2816	775.437983	187.845733	417.280000	1526.200000
$Imob_r$	2815	−0.000132	0.020105	−0.194627	0.143761

Source: elaborated by the author.

To further explore the stationarity of the series, the auto_arima function from the python library statsmodels was used to determine what stationary model fits the series best. The auto_arima algorithm determines the best model using the AIC/BIC criteria. The conclusion is rather interesting, since the model that best fits the series determined was an ARIMA(0,0,0), which is basically white-noise. In Figure 4.3 below the summary of the model fitted is shown:

Figure 4.3: ARIMA(0,0,0) summary

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 2815
Model:                        SARIMAX   Log Likelihood                7003.775
Date:                 Wed, 07 Sep 2022   AIC                         -14005.550
Time:                         19:55:28   BIC                         -13999.607
Sample:                              0   HQIC                        -14003.406
                                  - 2815
Covariance Type:                    opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         0.0004   4.09e-06     98.810      0.000       0.000       0.000
===================================================================================
Ljung-Box (L1) (Q):                   1.97   Jarque-Bera (JB):             16846.17
Prob(Q):                              0.16   Prob(JB):                         0.00
Heteroskedasticity (H):               2.47   Skew:                            -0.89
Prob(H) (two-sided):                  0.00   Kurtosis:                        14.85
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Source: elaborated by the author.

Another aspect to be analysed regards the distribution of the series. Figure 4.4 shows the histogram of the $Imob_r$. Most of the returns are concentrated in a small range surrounding zero and the mean, as seem in Table 4.3, is pretty close to zero. As a result, the distribution have noticeable heavy tails, which points towards α-stable distribution with an αparameter smaller than 2. A distribution is considered stable if a linear combination of two random independent variables, with said distribution, also has the same distribution. Stable distributions have an α parameter between 0 and 2, with the case of α= 2 being the Gaussian distribution (MARSHALL, 1926). To test whether the distribution isn't normal, which implies in an α< 2, the Shapiro-Wilk test (SHAPIRO; WILK, 1965) was applied and the results can be seen in Table 4.4.

The test uses the following null and alternative hypotheses:

- Null Hypothesis (H0): the $Imob_r$ sample came from a normally distributed population.
- Alternative Hypothesis (HA): the $Imob_r$ sample did not come from a normally distributed population.

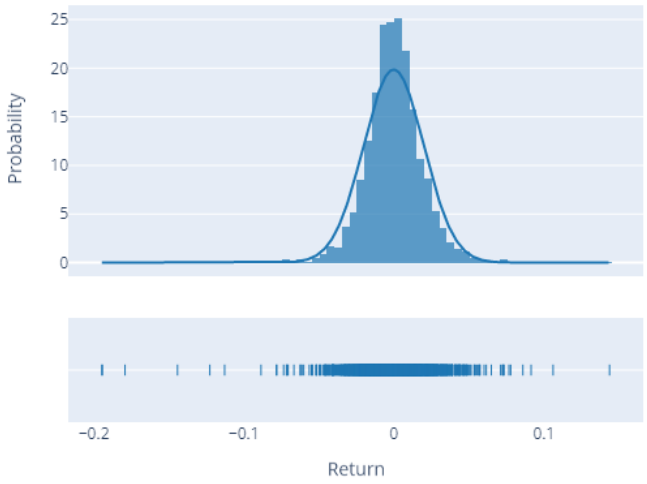
Table 4.4: Shapiro Wilk test results ($Imob_r$)

Statistic	0.918630
p-value	$1.649149e{-}36$

Source: elaborated by the author.

Since the p-value is inferior to 5%, it is possible to reject the null hypothesis and infer that the Imob Index Returns distribution is not Gaussian.

Figure 4.4: $Imob_r$ probability distribution

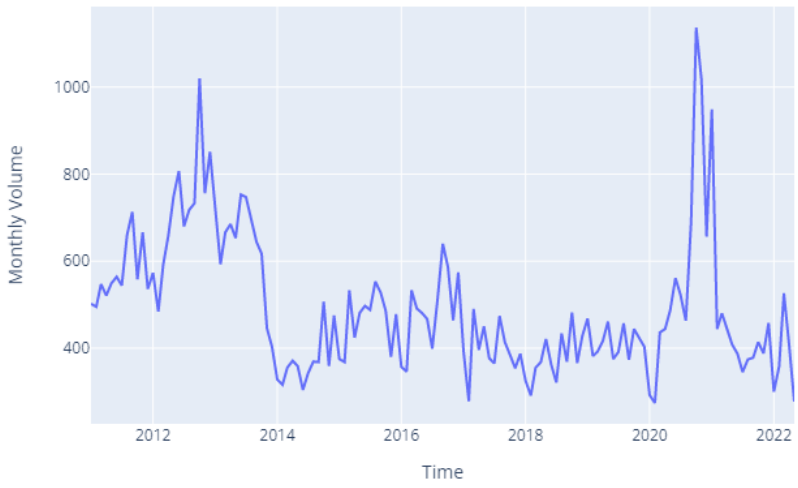


Source: elaborated by the author.

4.2 G1 news database

The G1 news database has over 67.000 news headlines in the analysed period. The monthly volume of news has an interesting behavior, as shown in Figure 4.5.

Figure 4.5: Monthly volume of G1 news



Source: elaborated by the author.

What we expected to see were peaks in years of presidential election, but the peaks in the database happened in 2012 and 2020. There's also a noticeable decline in the volume of news between 2014 and 2019. In fact, the peaks in October 2012 and October 2020 are related to the elections. Therefore, there might be some possible explanations for

this behaviour and solutions will be further detailed in Section 8. As detailed in Section 6, a keyword filtering was implemented to transform the database. The Table 4.5 below shows the top 10 keywords identified in news headlines throughout the period.

Table 4.5: Top 10 most frequent keywords in G1 headlines

Keyword	N° of headlines
$governo$	7957
$camara$	3588
$prefeito$	3578
$presidente$	2992
$dilma$	2870
$vereador$	2505
$ministro$	2077
$candidato$	1990
$deputado$	1557
$eleicoes$	1398

Source: elaborated by the author.

The most frequent keyword throughout the whole period (being present in almost 12% of the headlines) is "governo", which in Portuguese means "government". The presence of "dilma", used to match headlines regarding the ex-president Dilma Rousseff, in fifth place (present in 4% of the headlines) is probably a consequence of the fact that she is the president with the longest term of office in the period covered by the database (from 2010 to 2016). The keyword "eleicoes", which in Portuguese means "elections", is in 10th place. An interesting behavior was found regarding this keyword. The day with the greatest amount of headlines containing the keyword was January 1st 2021. Compared to other years, this specific date has 471 headlines about elections because of a large amount of news regarding politicians (specially mayors) taking office. This pattern was not observed in earlier years and it may be related to a change on how G1 covered the elections.

Figure 4.6: Top 5 most frequent keywords in G1 headlines yearly

2011

Keyword	N° of headlines
governo	967
dilma	667
presidente	373
fmi	328
ministro	276

2012

Keyword	N° of headlines
governo	992
dilma	645
candidato	467
presidente	451
ministro	288

2013

Keyword	N° of headlines
governo	901
presidente	559
dilma	542
camara	388
prefeito	315

2014

Keyword	N° of headlines
governo	364
candidato	317
dilma	286
presidente	234
camara	207

2015

Keyword	N° of headlines
governo	648
dilma	466
camara	441
prefeito	303
vereador	226

2016

Keyword	N° of headlines
governo	570
candidato	364
prefeito	334
camara	316
temer	308

2017

Keyword	N° of headlines
governo	595
camara	310
temer	309
vereador	286
prefeito	264

2018

Keyword	N° of headlines
governo	382
vereador	245
camara	239
prefeito	209
candidato	206

2019

Keyword	N° of headlines
governo	672
bolsonaro	422
camara	377
vereador	274
presidente	219

2020

Keyword	N° of headlines
eleicoes	1093
governo	909
prefeito	517
candidato	370
bolsonaro	353

2021

Keyword	N° of headlines
prefeito	794
governo	746
camara	446
bolsonaro	297
vereador	259

2022

Keyword	N° of headlines
governo	211
camara	118
bolsonaro	106
vereador	91
prefeito	76

Source: elaborated by the author.

In relation to Figure 4.6, some observations can be made. As seen before, the keyword "governo" is the main keyword in every year except 2020 and 2021 in which it topped at second place. This happens because the of volume of news about elections ("eleicoes") in late 2020 and mayors ("prefeito") taking office in early 2021, as observed before. Another interesting observation is that the keyword "dilma" is present in every year of the president's mandate except in 2016 when she was impeached. Instead, in 2016, we see "temer" referencing Michel Temer which took over the presidency after the end of the impeachment process in late 2016. Having in mind that the last presidential election occured in late 2018, from 2019 on we can observe that the keyword "bolsonaro", referencing the current president Jair Messias Bolsonaro, appears every year. These observations give us reliability that the database represents major political events of each year.

5 EMPIRICAL STRATEGY

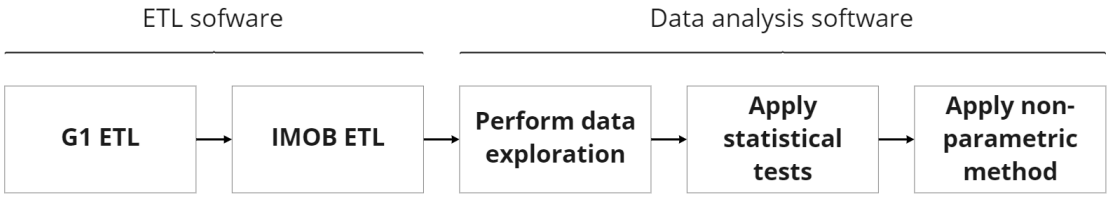
The empirical strategy present in this study, which aims to test the impacts of political news on the Brazilian real estate market, consists, initially, in the extraction of data that will be used in the statistical tests.

Two types of data were collected: the series of returns from the Imob Index Prices (between 2011 and 2022) and political news headlines from the G1 portal within the same period. All data was extracted, transformed and loaded using a software that was developed for this purpose.

A second software is used to perform data exploration, statistical tests and apply a non-parametric method to test whether political and economic news had an affect on the volatility of the real estate market using 21-day windows. Both softwares can be found in the github repository (Sá, 2022).

In Figure 5.1 below there's a flow that summarizes the empirical strategy implemented in this paper:

Figure 5.1: Empirical strategy flow



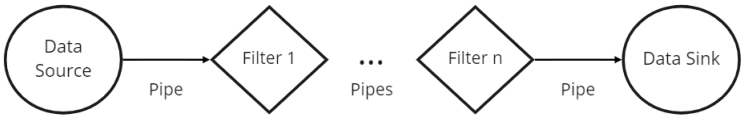
Source: (Sá, 2022).

6 METHODS

6.1 ETL Software

It was concluded that, to gather the data used, the best alternative was to develop a software which would enable the whole empirical strategy to be replicable or even amplified, since it could still be used in years to come and new findings could be made. The Python programming language was used to implement the software and execute the ETL flow, which consists of extraction, transformation and loading. The software architecture used, which is the one that most fits this chosen approach, is called pipes and filters. In Pipe-and-Filter architectures, the computational components (or scripts) are the filters that receive an input, transform according to one or more means, and then generate an output for a communication channel. These input and output conductors are called pipes. The Figure 6.1 below explicits the intended flow:

Figure 6.1: Pipes and filter flow



Source: elaborated by the author.

The IMOB Index and the G1 news database need different methods to perform the extraction and transformation, given that the data sources and data structure are completely different. In both cases, the technique applied to extract the data is called Web Scraping, which consists of collecting data other than through APIs or manually. To do so, three main python libraries were used: requests (to extract the raw source code from pages), selenium (to automate tasks in web pages) and bs4 (to parse the source code). The next step consists of the transformation through cleaning, structuring and filtering the obtained data. The library used to do so was pandas. Finally, to load the data into a database (to further analyze it), the sqlite3 library did the job. The Table 6.1 below summarizes all libraries used:

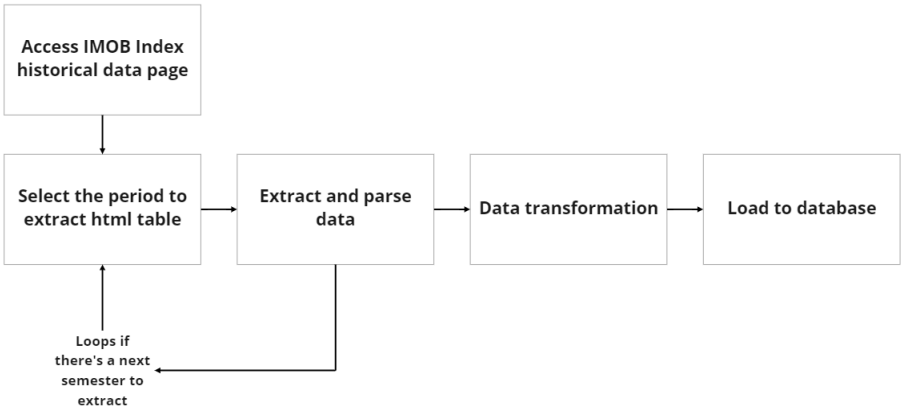
Table 6.1: ETL Software libraries

Library	Role
$requests$	Perform GET requisition to urls and collect their source code.
$selenium$	Automatize tasks in web pages to perform data extraction from the source code.
$bs4$	Parse the source code (in html) and transform it into an easily iterable python object.
$pandas$	Transform the extracted data.
$sqlite3$	Load the data into a sqlite database that can be queried.

Source: elaborated by the author.

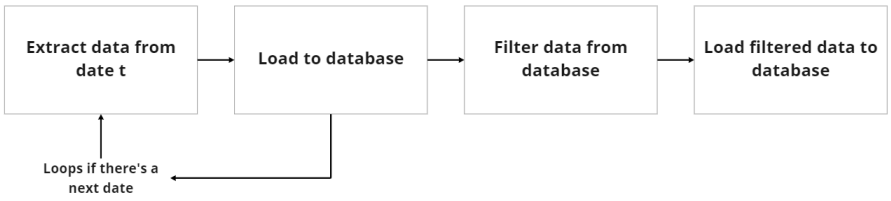
This section will give more detail about each database and how its extraction and transformation was implemented in the software. We'll start with the Imob Index Prices: in the extraction stage, the page where the historical data of the index is located (B3's website) is accessed. Since the data is made available by semester and there is the need to click on a dropdown list to select each year and semester, the selenium library is very handy since we can automate and select all years and semesters since the first available period and extract each html, parse it and transform it into a pandas dataframe (a tabular pandas object that allows us to perform transformations and filters). Then, transformations are performed to pivot the table and adjust the data types of the columns to be readable and used in further statistical tests. Below there's Figure 6.2 that summarizes the flow from the ETL algorithm implemented:

Figure 6.2: Imob ETL flow



Source: Source: (Sá, 2022).

Figure 6.3: G1 news database flow



Source: Source: (Sá, 2022).

The G1 political news ETL demands a more complex and time consuming algorithm, since news headlines since the beginning of 2011 are extracted, transformed and loaded into a sqlite database. The time consumed to do the task is rather extensive since it is necessary to iterate over each day and each page filled with results (which can be up to forty). Afterwards, the data is inserted in a database which structure prevents data duplication. Later, to guarantee that the news headlines in the database are related to politics, a keyword filtering is performed and then the data is replaced in the database. Most of the keywords selected were found in (CÂMARA et al., 2020), which is an e-book provided by the Brazilian legislative power that contains the 150 most important terms related to politics. An analysis based on the volume of headlines was made to determine the most relevant keyword among the ones cited in the e-book. Additionally, names of former presidents, terms related to economic policy, the main Brazilian political parties, the main taxes levied in Brazil, terms related to corruption and names of the main state-owned companies were added to the keyword list. All keywords can be found in Section 9. Figure 6.3 summarizes the ETL flow implemented.

6.2 Data Analysis Software

A second software was developed to apply all needed statistical tests, perform data transformation regarding data analysis and apply the non-parametric approach, which will be further explained in the next subsection. The software also generates and saves all test results (as csv files) and plots (as png files) present in this paper. To manipulate the collected data and generate dataframes, two libraries were used: numpy and pandas. To apply statistical tests two libraries are used: statsmodels and pmdarima. Two libraries were used to plot data: matplotlib and plotly. Table 6.2 below summarizes all libraries used:

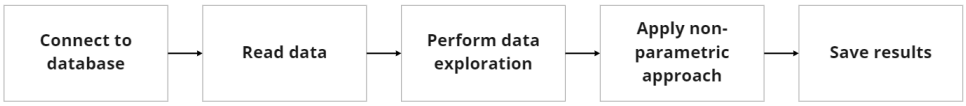
Table 6.2: ETL Software libraries

Library	Role
$pandas$	Data manipulation.
$numpy$	Mathematical transformations.
$statsmodels$	ADF, KPSS, Granger causality application.
$pmdarima$	ARIMA fitting.
$matplotlib$	Save ARIMA summary as png.
$plotly$	Generate and save plot as png.

Source: elaborated by the author.

Figure 6.4 below summarizes the algorithm:

Figure 6.4: Data analysis software flow



Source: Source: (Sá, 2022).

6.3 Non-parametric method

Considering the Imob Returns series stationarity and distribution it was not possible to apply a parametric approach, such as in (MARQUES; SANTOS, 2016). Having that in mind, it was decided to apply a non-parametric method that consists of:

1. Generate the volatility series of Imob Returns using 21-day windows ($Imob_w$).
2. Generate the volume of news using 21-day windows ($G1_w$).
3. Apply the Granger causality test (GRANGER, 1969) using both series.

The intuition behind the Granger causality test is that a X time series Granger-causes another Y time series if predictions of the value of Y based on past values of Y and X are better than predictions of Y based only on Y's past values. As defined by Granger, causality is based on two principles:

1. The cause happens prior to its effect.

2. The cause has unique information about the future values of its effect.

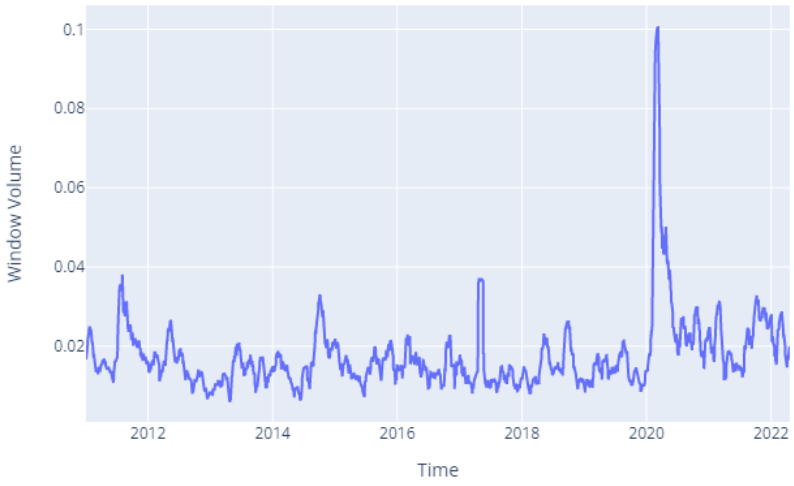
The test uses the following null and alternative hypotheses:

- Null Hypothesis (H0): $G1_w$ time series doesn't Granger-cause $Imob_w$ time series.
- Alternative Hypothesis (HA): $G1_w$ time series Granger-causes $Imob_w$ time series.

7 RESULTS AND DISCUSSION

The results points towards the acceptance of Fama's semi-strong efficiency market hypothesis. First, the Imob returns volatility series ($Imob_w$) was generated by the data analysis software. The result is in Figure 7.1 below:

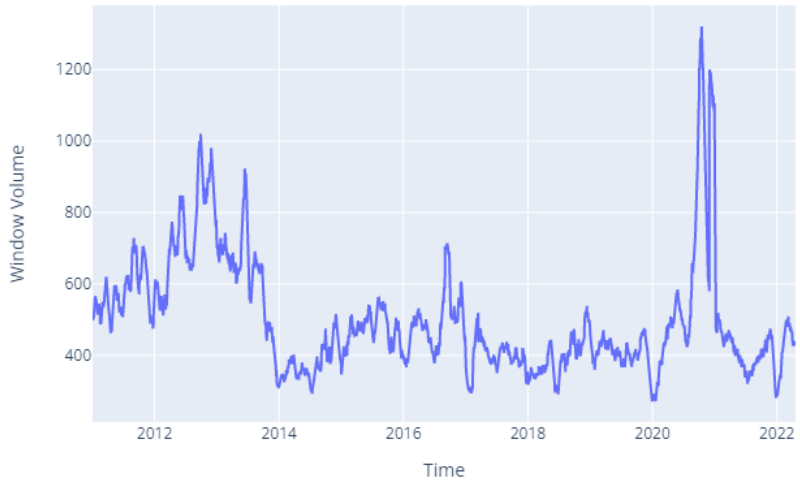
Figure 7.1: $Imob_w$ series 2011-2022



Source: elaborated by the author.

As expected, we can observe that the period with greater volatility happens during the first burst in 2020. The volume of news using 21-day windows ($G1_w$) is in Table 7.2 below:

Figure 7.2: $G1_w$ series 2011-2022



Source: elaborated by the author.

With both series transformed, it was possible to apply the Granger causality test

(GRANGER, 1969). It was decided to apply the test with a maximum lag equals to 1. The results are in Table 7.1.

Table 7.1: Granger causality test results

Statistic	1.059733
p-value	0.364993

Source: elaborated by the author.

Since the p-value is no less than 5%, it is not possible to reject the null hypothesis. Thus, it is possible to infer that the $G1_w$ series doesn't Granger-cause $Imob_w$ series.

Even though there seem to be some similar behaviour in 2020, at least by looking at the plots, the peaks happened in different windows for different reasons. As explained in section 4, the more noticeable peaks in $G1_w$ are related to the 2020 elections that happened in October and politicians taking office in early 2021. The peaks observed in $Imob_w$ are most likely related to the Coronavirus outbreak.

8 CONCLUSION

This paper intended to investigate whether political news affected the Brazilian real estate market returns during the period ranging from January 2011 to May 2022. To do so, two databases where generated: the Imob Index Price series from B3's web page and news headlines related to political events from G1 news portal (which is one of the biggest news portal in Brazil). All data used in this paper was extracted, transformed and loaded using an ETL software developed in python programming language. Data exploration and the application of the non parametric method was performed by a second software, also developed in python.

The non parametric method consisted of generating the volatility series of Imob Returns ($Imob_w$) and the volume of news headlines ($G1_w$) using 21-day windows. With both series in hand, is was possible to apply the Granger causality test to determine whether $G1_w$ Granger-causes $Imob_w$. It was concluded that it is possible to infer that the $G1_w$ series doesn't Granger-cause $Imob_w$ series, which points towards the acceptance of Fama's semi-strong efficiency market hypothesis, at least when public information is political news.

As commented in section 4, the news headlines database presented an odd behaviour and counter intuitive results regarding the volume of news in certain periods. Some speculation can be made: G1's portal, when searching for news related to political events, might not be presenting all results, which affects the volume of headlines. To be more confident on the results presented, it is intended to improve the ETL software by extracting data from different sources, since collecting from only one news portal might affect the results. Another improvement that can be implemented is related to speed and efficiency. It's intended to implement asynchronicity in functions, so that the speed of extraction is increased. Also, related to the first improvement, multiprocessing can be implement to perform parallel extraction from different news portals.

9 APPENDIX

9.1 ETL Software

```
# etl.py

from etl.etl import run_imob_etl, run_g1_etl
from etl.utils.database import get_database_name
from data_analysis.data_analysis import (
    ↪ perform_data_exploration,
    ↪ apply_non_parametric_approach)

from os.path import exists
from os import mkdir

if __name__ == '__main__':
    DB_NAME = get_database_name()

    run_imob_etl(DB_NAME)
    run_g1_etl(DB_NAME)

    if not exists('./results'):
        mkdir('./results')

    data = perform_data_exploration(DB_NAME)
    apply_non_parametric_approach(data)

# g1_extraction.py
from bs4 import BeautifulSoup
import requests
from datetime import datetime, timedelta
import pandas as pd
import time
```

```
import re

def all_days_since(start_date:str):
    '''Creates a list of date with all days since a given
       ↪ date.
    The start_date parameter must be a string
    date with this format: %Y-%m-%d.'''

    return [datetime.strftime(d, '%Y-%m-%d')
            for d in pd.date_range(start=start_date,
                                   end=datetime.strftime(
                                      ↪ datetime.today(),"%
                                      ↪ Y-%m-%d"))
                                   .to_pydatetime()
                                   .tolist()]

def g1_requisition(page:str, date:str):
    '''Requests the g1 search url. Returns the requests
       ↪ object
    if the status_code is equal to 200.'''

    time.sleep(0.5)

    try:
        url = f'https://g1.globo.com/busca/?q=pol tica%2C+
           ↪ economia%2C+incerteza&page={page+1}&order=
           ↪ recent&species=not%C3%ADcias&from={date}T00%3
           ↪ A00%3A00-0300&to={date}T23%3A59%3A59-0300'
        req = (requests.get(url, timeout=7.5))

        if req.status_code == 200:
            return req
        return False
```

```
    except Exception as e:
        err_name = type(e).__name__
        print(f'Requisition␣error␣to␣url␣{url}:␣{err_name}'
          ↪ )
        return False

def create_soup(req):
    '''Parses the source code and creates a soup object.'''

    return BeautifulSoup(req.content, 'html.parser')

def not_exists_results(soup):
    '''Returns if there are still results in the html page.
      ↪ '''
    if soup.find_all('p', {'class': 'widget--no-
      ↪ results__title'}):
        return True

def extract_date_from_div(div:str):
    '''Returns the date in cases of only having days,
    hours or minutes since the news.'''

    date_now = datetime.now()

    try:
        time_since_news = int(re.findall(r'\d', div)[0])

        if 'minutos' in div:
            return (date_now - timedelta(minutes=
              ↪ time_since_news)).date()
        elif 'horas' in div:
            return (date_now - timedelta(hours=
              ↪ time_since_news)).date()
        elif 'dias' in div:
```

```
            return (date_now - timedelta(days=
                ↪ time_since_news)).date()
        elif 'segundos' in div:
            return (date_now - timedelta(seconds=
                ↪ time_since_news)).date()
    except:
        print(div)
        return date_now.date()

def performs_scrap(soup, batch_df_list, page, date):
    '''Scrapps the data from the g1 news page.'''

    final_headline_data = []
    final_date_data = []
    id = 1 # News id counter

    # Scrapping all headlines in the page
    headlines = soup.find_all('div',
    {'class': 'widget--info__title␣product-color'})

    for div in headlines:
        index_1 = str(div).find('>') + 1
        index_2 = str(div).find('</')
        final_headline_data.append([id, datetime.strptime(
            ↪ date, '%Y-%m-%d'), page+1, str(div)[index_1:
            ↪ index_2].strip()])
        id += 1

    # Reseting the id counter
    id = id - len(headlines)

    # Scrapping all dates in the page
    dates = soup.find_all('div', {'class': 'widget--
        ↪ info__meta'})
```

```
    for div in dates:
        index_1 = str(div).find('>') + 1
        index_2 = str(div).find('</')
        if 'h ' in str(div):
            final_date_data.append([id,
                ↪ extract_date_from_div(str(div))])
        else:
            final_date_data.append([id, datetime.strptime(
                ↪ str(div)[index_1:index_2].strip(),'%d/%m
                ↪ /%Y␣%Hh%M')])
        id += 1

    final_database = (pd.DataFrame(final_headline_data,
        ↪ columns=['news_id','dt_loop','pg_loop','headline'
        ↪ ]).merge(pd.DataFrame(final_date_data, columns=['
        ↪ news_id', 'headline_dt']), how='inner', on='
        ↪ news_id'))
    batch_df_list.append(final_database)

    return batch_df_list

def extract_news_from_date(date:str):
    '''Extract the news from all pages with results
        ↪ filtering by a date.'''

    batch_df_list = []
    for page in range(0,40): # G1 only allows to search to
        ↪ page 40
        req = g1_requisition(page, date)
        if req:
            soup = create_soup(req)
```

```
            # If there's no results in the day, the loop
              ↪ breaks
            if not_exists_results(soup):
                break
            else:
                batch_df_list = performs_scrap(soup,
                batch_df_list,
                page,
                date)

    return pd.concat(batch_df_list, ignore_index=True)

# imob_extraction.py

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
from time import sleep
import pandas as pd

def create_google_driver():
    '''This function returns the chrome webdriver object.
      ↪ '''

    driver = webdriver.Chrome(executable_path="./etl/utils/
      ↪ chromedriver.exe")
    driver.implicitly_wait(.5)
    return driver

def get_html_selectors(driver:webdriver):
    '''This function returns the selector objects used in
      ↪ the automation.'''

    driver.switch_to.frame('bvmf_iframe')
```

```
    sel_year = Select(driver.find_element(By.ID, '
        ↪ selectYear'))
    sel_semester = Select(driver.find_element(By.ID, '
        ↪ semester'))

    return sel_year, sel_semester

def extract():
    '''This function performs the extraction of the IMOB
        ↪ index raw data from
    B3s website and returns a list of dataframes.'''

    # Creating driver
    driver = create_google_driver()

    # Launching the URL
    driver.get("https://www.b3.com.br/pt_br/
␣␣␣␣market-data-e-indices/indices/indices-de-segmentos-e-
    ↪ setoriais/
␣␣␣␣indice-imobiliario-imob-estatisticas-historicas.htm")

    # Sleep to make sure the page is fully loaded
    sleep(2)

    # Selectors
    sel_year, sel_semester = get_html_selectors(driver)

    # Year and Semester list to loop over
    years = ['2008', '2009', '2010', '2011', '2012', '2013'
        ↪ , '2014',
    '2015', '2016', '2017', '2018', '2019', '2020', '2021',
        ↪  '2022']
    semesters = ['1', '2']
```

```
    main_df = []

    for year in years:
        sel_year.select_by_visible_text(year)
        for semester in semesters:
            sel_semester.select_by_value(semester)
            sleep(2)
            tbl = (driver.find_element(By.XPATH,
            '''/html/body/app-root/app-daily-evolution/
            div/div/div[1]/form/div[2]/div/table''')
            .get_attribute('outerHTML'))
            main_df.append([year, pd.read_html(tbl)[0]])

    driver.close()
    driver.quit()

    return main_df

# g1_loading.py

import pandas as pd
from etl.utils.database import (create_or_connect_db,
    ↪ insert_ignore_into_table_g1, create_table_g1)

def load_data(df:pd.DataFrame, db_name:str):
    '''Loads the data to the db using insert ignore.'''
    conn = create_or_connect_db(db_name)
    cursor = conn.cursor()
    insert_ignore_into_table_g1(df, conn, cursor)

def replace_data(df:pd.DataFrame, db_name:str):
    '''Replaces the data of a table in the database.'''
    conn = create_or_connect_db(db_name)
```

```
    df.to_sql('news', conn, index=False, if_exists='replace
        ↪ ')

def create_db_and_table(db_name:str):
    '''Connects to a db if it exists. Otherwise creates it.
        ↪ '''
    conn = create_or_connect_db(db_name)
    cursor = conn.cursor()
    create_table_g1(cursor)

# imob_loading.py

from etl.utils.database import (create_or_connect_db,
                                create_table_imob,
                                insert_into_table_imob)

def load_imob_data_to_db(df, db_name):
    '''Loads the imob data to the database.'''
    # connection and cursor
    conn = create_or_connect_db(db_name)
    cursor = conn.cursor()

    # create the table
    create_table_imob(cursor)

    # insert data into table
    insert_into_table_imob(df[['date','price']], conn)

    # closing connection
    conn.close()

# g1_transformation.py

import pandas as pd
```

```
from unidecode import unidecode
from etl.utils.database import create_or_connect_db

def unidecode_headlines(news_data):
    '''Applies unidecode to all headlines.'''
    news_data.headline = news_data.headline.apply(lambda x:
        ↪ unidecode(x))
    return news_data

def lower_case_headlines(news_data):
    '''Applies lowercase to all headlines.'''
    news_data.headline = news_data.headline.str.lower()
    return news_data

def filter_headlines_with_keywords(news_data:pd.DataFrame):
    '''Performs a filter using keywords related to politics
        ↪ and economics.'''
    return news_data[news_data.headline.str
    .match(r'''politic|bolsonaro|inflacao|lula|dilma|temer|
        ↪ pib|public
|ministro|lava-jato|cassacao|imposto|assembleia|ativismo|
    ↪ audiencia
|camara|legislativ|campanha|corrup|judiciario|executivo|
    ↪ candidato
|governo|cidadania|clausula|comissao|congresso|stf|tribunal
    ↪ |eleitoral
|estadual|constituicao|conselho|decreto|decoro|democracia|
    ↪ deputado
|vereador|prefeito|senador|direito|desembargador|federal|df
    ↪ |ditadura
|estado|federacao|governador|ideologia|impeachment|
    ↪ inegibilidade|juiz
|legislatura|lei|lobby|medida␣provisoria|movimentos|mst|
    ↪ mtst|ministerio
```

```
|municipio|orcamento|parlament|partido|pt|psdb|mdb|ptb|pdt|
    ↪ pcdob|psb|pp
|psol|pstu|agir|psc|pmn|pv|pcb|prtb|pco|republica|guerra|
    ↪ uniao|patriota
|plebiscito|plenario|populismo|presidente|programa_social|
    ↪ promotor|quorum
|referendo|resolucao|fiscal|sancao|servidor|sistema|taxa|
    ↪ terrorismo
|totalitarismo|tcu|transparencia|tributo|veto|voto|propina|
    ↪ delacao
|monetaria|credito|cvm|deficit|superavit|deflacao|divida|
    ↪ endividamento
|exportacao|importacao|financ|investimento|fmi|juro|selic|
    ↪ receita
|poupanca|previdencia|reforma|renda|pobreza|risco|
    ↪ tributacao|mandato
|estatal|clt|moeda|incerteza|consumo|emprego|privatizacao|
    ↪ estatizacao
|estimulo|crescimento|industria|agro|classe|salario|
    ↪ moratoria
|recessao|nacao|nacionalismo|direita|esquerda|centro|
    ↪ progressista
|bndes|eletrobras|privado|regulacao|desenvolvimento|
    ↪ subsidio|tarifa
|petrobras|correios|banco_do_brasil|banco_central|bacen|
    ↪ diplomacia
|liberdade|demagogia|burocracia|igualdade|poder|civil|golpe
    ↪ |povo
|conservador|icms|ipi|irpj|iof|pis|cofins|fgts|itr|inss|
    ↪ pasep|ipva
|iptu|iss|itbi|itcmd|cide|posse|eleito|eleicao|eleicoes|
    ↪ debate
|gabinete|militar|protesto|manifestacao''')]
```

```
def gather_data(db_name:str):
    '''Gathers the data from the database.'''
    conn = create_or_connect_db(db_name)
    return data_from_db(conn, 'news')

def data_from_db(conn, table_name):
    '''Executes a query that selects all data from a table
      ↪ in the database.'''
    return pd.read_sql(f'select * from {table_name}', conn)

def transform(db_name:str):
    '''Performs the g1 headlines processing.'''
    news_data = gather_data(db_name)
    news_data = unidecode_headlines(news_data)
    news_data = lower_case_headlines(news_data)
    return filter_headlines_with_keywords(news_data)

# imob_transformation.py

import pandas as pd

def pivot_and_transform(imob_tables:pd.DataFrame):
    '''This function pivot the table and transforms the
      ↪ index values to float.'''
    # Treating and generating the final database
    final_database_list = []

    # Dictionary of months to properly
    # generate a date based on the month name
    months = {
        "Jan": '01',
        "Fev": '02',
        "Mar": '03',
        "Abr": '04',
```

```
        "Mai": '05',
        "Jun": '06',
        "Jul": '07',
        "Ago": '08',
        "Set": '09',
        "Out": '10',
        "Nov": '11',
        "Dez": '12'
    }

    for data in imob_tables:
        # Dropping useless rows
        df = data[1][~data[1].Dia.isin(['MÍNIMO', 'MÁXIMO
            ↪ '])]

        # Looping over each row to
        # treat the values and pivoting table
        treated_database_list = []
        for row in df.itertuples():
            for c in range(len(row) - 2):
                if str(row[c + 2]) != 'nan':
                    try:
                        number = str(int(row[c + 2]))
                        number = number[:3] + '.' + number
                            ↪ [3:]
                    except:
                        number = str(row[c + 2]).replace('.
                            ↪ ','')
                        number = number.replace(',','.')
                    if int(row[1]) < 10:
                        (treated_database_list
                        .append([f'''{data[0]}
                        -{months[df.columns[c + 1]]}
                        -0{row[1]}''', number]))
```

```
                    else:
                        (treated_database_list
                        .append([f'''{data[0]}
                        -{months[df.columns[c + 1]]}
                        -{row[1]}''', number]))
        final_database_list.append(pd.DataFrame(
            ↪ treated_database_list, columns=['date', '
            ↪ price']))

    return pd.concat(final_database_list, ignore_index=True
        ↪ )

def date_to_datetime(imob_treat:pd.DataFrame):
    '''This function transforms the date column into
        ↪ datetime.'''

    imob_treat['date'] = pd.to_datetime(imob_treat['date'],
        ↪  format='%Y-%m-%d')
    return imob_treat

def sort_by_date(imob_treat:pd.DataFrame):
    '''This function sort the values by date.'''

    return imob_treat.sort_values('date')

def cast_float(imob_treat:pd.DataFrame):
    '''This function transforms the price column into float
        ↪ .'''

    imob_treat.price = imob_treat.price.astype(float)
    return imob_treat

def transform(raw_df:pd.DataFrame):
    '''This function performs the pipeline to transform the
```

```
    data to be usable in statistical tests.'''

    imob_treat = pivot_and_transform(raw_df)
    imob_treat = date_to_datetime(imob_treat)
    imob_treat = sort_by_date(imob_treat)
    return cast_float(imob_treat)

# database.py

import sqlite3
import pandas as pd
from datetime import datetime

# general functions
def get_database_name():
    '''Returns the database name.'''

    return f'{datetime.strftime(datetime.now(), "%Y_%m_%d_%
        ↪ H_%M")}_database'

def create_or_connect_db(db_name:str):
    '''Connects to the database if it exists.
    Creates and connect if it doesn't.'''

    return sqlite3.connect(f'{db_name}.db')

# tables and inserts
# g1
def create_table_g1(cursor):
    '''Creates the g1 news database inside the database.'''
    cursor.execute('''CREATE TABLE IF NOT EXISTS news (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    dt_loop DATE,
                    pg_loop INTEGER,
```

```
                    headline STRING,
                    headline_dt DATE,
                    UNIQUE (dt_loop, headline));''')

def insert_ignore_into_table_g1(df:pd.DataFrame,
                                conn:sqlite3.connect,
                                cursor):
    '''Performs insert ignore into the database.'''

    cursor.execute('''DROP TABLE IF EXISTS temporaryData;
    ↪ ''')
    conn.commit()
    df.to_sql('temporaryData', conn)
    cursor.execute('''INSERT OR IGNORE INTO
    news (dt_loop, pg_loop, headline, headline_dt)
    SELECT dt_loop, pg_loop, headline, headline_dt
    FROM temporaryData''')
    conn.commit()

# imob
def create_table_imob(cursor):
    '''Creates the imob index table inside the database.'''

    cursor.execute('''CREATE TABLE IF NOT EXISTS imob (
                    date DATE,
                    price FLOAT);''')

def insert_into_table_imob(df:pd.DataFrame, conn:sqlite3.
    ↪ connect):
    '''Inserts imob index data into the database.'''

    df.to_sql('imob', conn, if_exists ='replace')
```

9.2 Data Analysis Software

```
# data_analysis.py

import data_analysis.data_loader as data_loader
import data_analysis.data_statistics as data_statistics
from data_analysis.database import create_or_connect_db
from os.path import exists
from os import mkdir
import pandas as pd

def perform_data_exploration(DB_NAME):
    '''Performs data exploration:
       - Applies statistical tests and generates results
       - Generate plots'''

    conn = create_or_connect_db(DB_NAME)

    imob_data = data_loader.data_from_db(conn, 'imob')
    g1_data = data_loader.data_from_db(conn, 'news')

    imob_adjusted = data_statistics.adjust_imob_date_range(
       ↪ imob_data)

    (data_statistics
    .generate_and_save_imob_prices_plot(imob_adjusted))

    imob_with_returns = (data_statistics
                         .generate_imob_returns_column(
                            ↪ imob_adjusted))

    (data_statistics
    .generate_and_save_imob_returns_plot(imob_with_returns)
       ↪ )
```

```
(data_statistics
.generate_and_save_adf_test_df(imob_with_returns.
    ↪ log_ret[1:]))

(data_statistics
.generate_and_save_kpss_test_df(imob_with_returns.
    ↪ log_ret[1:]))

(data_statistics
.generate_and_save_descriptive_analysis(
    ↪ imob_with_returns))

arima_fit = (data_statistics
             .perform_auto_arima(imob_with_returns.
                 ↪ log_ret[1:]))

data_statistics.save_arima_summary_png(arima_fit)

(data_statistics
.generate_and_save_imob_returns_dist_plot(
    ↪ imob_with_returns.log_ret[1:]))

daily_volume_of_news = (data_statistics
                        .
                            ↪ np_method_generate_volume_of_news
                            ↪ (g1_data))

(data_statistics
.generate_and_save_g1_volume_of_news_daily_plot(
    ↪ daily_volume_of_news))

(data_statistics
```

```
    .generate_and_save_g1_volume_of_news_month_plot(
        ↪ daily_volume_of_news))

    return {'imob_data': imob_with_returns,
            'g1_daily_volume_of_news':daily_volume_of_news}

def apply_non_parametric_approach(data:dict):
    '''Applies the non-parametric approch using the imob
        ↪ index data
    and g1 news data and save the results.'''
    imob_windows = (data_statistics
                    .np_method_generate_windows_imob(data['
                        ↪ imob_data']))
    g1_windows = (data_statistics
                  .np_method_generate_windows_g1(data['
                      ↪ g1_daily_volume_of_news'],
                                                 imob_windows
                                                     ↪ ))

    imob_windows_df = pd.DataFrame([[x[0][0], x[1]] for x
        ↪ in imob_windows],
                                   columns=['window', '
                                       ↪ value'])

    g1_windows_df =  pd.DataFrame({
                     'window':pd.Series([x[0][0] for x in
                         ↪ imob_windows]),
                     'value':pd.Series(g1_windows)
                     })

    data_statistics.generate_and_save_windows_as_png({
        'Imob Index volatility':imob_windows_df,
        'G1 volume of news':g1_windows_df
    })
```

```
    data_statistics.perform_and_save_granger_causality_test
        ↪ (imob_windows, g1_windows)

if __name__ == '__main__':
    # if running as main define DB_NAME
    DB_NAME = input('Input the sqlite database: ')

    if not exists('./results'):
        mkdir('./results')

    data = perform_data_exploration(DB_NAME)
    apply_non_parametric_approach(data)

# data_loader.py

import pandas as pd
import sqlite3 as sql

def data_from_db(conn, table_name):
    '''Executes a query and transform its results into a df
        ↪ .'''
    return pd.read_sql(f'select * from {table_name}', conn)

# data_statistics.py

import pandas as pd
import plotly.graph_objects as go
from statsmodels.tsa.stattools import adfuller, kpss
import numpy as np
import warnings
from pmdarima import auto_arima
from statsmodels.tsa.stattools import grangercausalitytests
import matplotlib.pyplot as plt
```

```
import plotly.figure_factory as ff

warnings.filterwarnings("ignore")

def adjust_imob_date_range(imob_raw:pd.DataFrame):
    '''Adjusts the range of data between 2011-01-01 and
    ↪ 2022-05-20.'''
    return (imob_raw[(imob_raw.date >= '2011-01-01') & (
    ↪ imob_raw.date <= '2022-05-20')]
            .reset_index(drop=True))

def save_figure_as_png(fig:go.Figure, name:str):
    '''Save a plotly figure as a png file.'''
    fig.write_image(f"./results/{name}.png")

def generate_figure(x:pd.Series, y:pd.Series):
    '''Generates a plotly figure.'''
    return go.Figure(go.Scatter(x=x,y=y))

def update_fig_layout(fig:go.Figure, title:str):
    '''Updates a plotly figure layout.'''
    fig.update_layout(
        title={
            'text': title,
            'y':0.9,
            'x':0.5,
            'xanchor': 'center',
            'yanchor': 'top'})
    return fig

def generate_and_save_imob_prices_plot(imob_prices_series:
↪ pd.DataFrame):
    '''Generates the imob prices plot figure and saves it
    ↪ as png.'''
```

```
    fig = generate_figure(imob_prices_series.date,
        ↪ imob_prices_series.price)
    fig = update_fig_layout(fig, "Figure 4.1 IMOB Index
        ↪ Prices 2011-2022")
    save_figure_as_png(fig, 'imob_series')

def generate_imob_returns_column(imob_adjusted:pd.DataFrame
    ↪ ):
    '''Generates the imob log returns columns in the imob
        ↪ data df.'''
    imob_adjusted['pct_change'] = imob_adjusted.price.
        ↪ pct_change()
    imob_adjusted['log_ret'] = np.log(imob_adjusted.price/
        ↪ imob_adjusted.price.shift(1))
    return imob_adjusted

def generate_and_save_imob_returns_plot(imob_returns:pd.
    ↪ DataFrame):
    '''Generates the imob returns plot figure and saves it
        ↪ as png.'''
    fig = generate_figure(imob_returns.date[1:],
                          imob_returns.log_ret[1:])
    fig = update_fig_layout(fig, "Figure 4.2 IMOB Index
        ↪ Returns")
    save_figure_as_png(fig, 'imob_returns')

def save_df_as_csv(df:pd.DataFrame, file_name:str):
    '''Saves a df as a csv.'''
    df.to_csv(f'./results/{file_name}.csv')

def generate_and_save_adf_test_df(imob_returns_series:pd.
    ↪ Series):
    '''Applies the ADF test and save the results as a csv.
        ↪ '''
```

```
    result = adfuller(imob_returns_series, autolag='AIC')
    df_rows = []

    for value in result[4].values():
        df_rows.append(value)

    adf_df = pd.DataFrame([result[0], result[1], df_rows
        ↪ [0], df_rows[1], df_rows[2]],
                          index=['ADF statistic', 'p-value
                              ↪ ', '1pct', '5pct', '10pct'
                              ↪ ],
                          columns=['Value'])

    save_df_as_csv(adf_df, 'imob_adf_test')

def generate_and_save_kpss_test_df(imob_returns_series:pd.
    ↪ Series):
    '''Applies the KPSS test and save the results as a csv.
        ↪ '''
    result = kpss(imob_returns_series, regression='ct')
    df_rows = []
    for value in result[3].values():
        df_rows.append(value)

    kpss_df = pd.DataFrame([result[0], result[1], df_rows
        ↪ [0], df_rows[1], df_rows[3]],
                           index=['KPSS statistic', 'p-
                               ↪ value', '10pct', '5pct',
                               ↪ '1pct'],
                           columns=['Value'])

    save_df_as_csv(kpss_df, 'imob_kpss_test')
```

```
def generate_and_save_descriptive_analysis(imob_returns:pd.
    ↪ DataFrame):
    '''Generates the descriptive statistics from imob data
        ↪ and saves the df as a csv.'''
    imob_returns_price_df = (imob_returns.price
                             .describe()
                             .to_frame()
                             .drop(['25%', '50%', '75%']))

    imob_returns_log_ret_df = (imob_returns.log_ret
                             .describe()
                             .to_frame()
                             .drop(['25%', '50%', '75%']))

    descriptive = pd.concat([imob_returns_price_df,
        ↪ imob_returns_log_ret_df], axis=1)
    descriptive = descriptive.rename(columns={'price':'IMOB
        ↪ ␣Index␣Price',
                                              'log_ret': '
                                                  ↪ IMOB␣
                                                  ↪ Index␣
                                                  ↪ Returns
                                                  ↪ '},
                                     index={
                                             'count':'N',
                                             'mean':'Mean',
                                             'std':'Std.␣Dev
                                                 ↪ .',
                                             'min':'Min',
                                             'max':'Max'})
    descriptive = pd.pivot_table(descriptive,
                                 columns=['N', 'Mean', 'Std
                                     ↪ .␣Dev', 'Min', 'Max'
                                     ↪ ])
```

```
    descriptive['N'] = descriptive['N'].astype(int)

    return descriptive[['N', 'Mean', 'Std. Dev', 'Min', '
      ↪ Max']]

def save_arima_summary_png(arima_fit:auto_arima):
    '''Saves the summary of the ARIMA model fitted as a png
      ↪ .'''
    plt.rc('figure', figsize=(7.5, 4))
    plt.text(0.01, 0.05,
             str(arima_fit.summary()),
             {'fontsize': 10},
             fontproperties = 'monospace')
    plt.axis('off')
    plt.tight_layout()
    plt.savefig('./results/arima_summary.png',facecolor='
      ↪ white', transparent=False)

def perform_auto_arima(imob_returns:pd.Series):
    '''Performs the auto_arima algorithm to determine the
      ↪ model to be fitted.'''
    # Fit auto_arima function to AirPassengers dataset
    arima_fit = auto_arima(imob_returns, start_p = 1,
      ↪ start_q = 1,
                           max_p = 2, max_q = 2,
                           start_P = 0, trace = True,
                           error_action ='ignore',
                           suppress_warnings = True
                           )

    return arima_fit

def generate_and_save_imob_returns_dist_plot(imob_returns:
  ↪ pd.Series):
```

```
    '''Generates the imob distribution plot figure and
       ↪ saves it as png.'''
    hist_data = [imob_returns]
    group_labels = ['IMOB Index Returns'] # name of the
       ↪ dataset
    fig = ff.create_distplot(hist_data, group_labels,
       ↪ bin_size=.005, curve_type='normal')
    save_figure_as_png(fig, 'imob_distribution')

def np_method_generate_windows_imob(imob_returns:pd.
   ↪ DataFrame):
    '''Generates an array that contains the windows and
       ↪ standard deviation
    of the IMOB index from each window of dates.'''
    p0 = 0
    pf = 20

    imob_windows = []

    while pf < len(imob_returns):
        janela = imob_returns[p0:(pf+1)]
        imob_windows.append([[janela.date.min(),
                              janela.date.max()],
                              imob_returns[p0:(pf+1)].log_ret.
                                 ↪ std()])
        p0 += 1
        pf += 1

    return imob_windows

def np_method_generate_volume_of_news(g1_news:pd.DataFrame)
   ↪ :
    '''Generates the daily volume of news df.'''
    print(g1_news)
```

```
    return g1_news.groupby('dt_loop').count().reset_index()
        ↪ [['dt_loop', 'headline']]

def generate_and_save_g1_volume_of_news_daily_plot(
    ↪ g1_news_df:pd.DataFrame):
    '''Generates the daily volume of news df and saves it
        ↪ as a png.'''
    fig = generate_figure(g1_news_df.dt_loop,
                          g1_news_df.headline)

    fig = update_fig_layout(fig, 'Daily volume of news')

    save_figure_as_png(fig, 'G1 daily volume of news')

def generate_and_save_g1_volume_of_news_month_plot(
    ↪ volume_of_news:pd.DataFrame):
    '''Generates the monthly volume of news df and saves it
        ↪  as a png.'''
    daily_volume_of_news = volume_of_news.copy()

    daily_volume_of_news.dt_loop = (pd.to_datetime(
        ↪ daily_volume_of_news.dt_loop)
                                    .dt
                                    .strftime('%Y-%m'))
    df_g1_grouped_by_month = daily_volume_of_news.groupby('
        ↪ dt_loop').sum().reset_index()

    fig = generate_figure(df_g1_grouped_by_month.dt_loop,
                          df_g1_grouped_by_month.headline)

    fig = update_fig_layout(fig, 'Monthly volume of news')

    save_figure_as_png(fig, 'G1 volume of news')
```

```
def np_method_generate_windows_g1(volume_of_news:pd.
    ↪ DataFrame, imob_windows:list):
    '''Generates the g1 windows containing the volume of
        ↪ news of each window.'''

    windows = []

    for window in imob_windows:
        g1_window = volume_of_news[(volume_of_news.dt_loop
            ↪ >= window[0][0])
                                    & (volume_of_news.
                                        ↪ dt_loop <=
                                        ↪ window[0][1])
                                        ↪ ]
        windows.append((g1_window).headline.sum())

    return windows

def generate_and_save_windows_as_png(windows:dict):
    for df_name, df in windows.items():
        fig = generate_figure(df.window, df.value)
        fig = update_fig_layout(fig, f'{df_name} - 21-day
            ↪ windows')
        save_figure_as_png(fig, f'{df_name} - 21-day
            ↪ windows')

def perform_and_save_granger_causality_test(imob_windows:
    ↪ list, g1_windows:list):
    '''Performs the Granger causality test with imob and g1
        ↪ values from each
    window and saves the results as a csv.'''
    imob_std_series = [x[1] for x in imob_windows]
    df_final_raw = {
    'volatility_IMOB':imob_std_series,
```

```
    'news_volume':g1_windows
    }
    df_final = pd.DataFrame(df_final_raw)
    f_test_results = grangercausalitytests(df_final[['
        ↪ volatility_IMOB', 'news_volume']], maxlag=[1])
        ↪ [1][0]['params_ftest']
    f_test_results_df = pd.DataFrame([f_test_results[0],
        ↪ f_test_results[1]], index=['F␣test␣statistic', 'p
        ↪ -value'], columns=['Value'])

    save_df_as_csv(f_test_results_df, 'Granger␣causality␣
        ↪ test')
```

9.3 Imob Index Portfolio Composition

Sector	Code	Stock	Type	Theoretical Qt.	%Sector
Cycling consumption / civil construction	CURY3	CURY S/A	ON NM	102.456.054	2,73%
Cycling consumption / civil construction	CYRE3	CYRELA REALT	ON NM	274.142.301	10,64%
Cycling consumption / civil construction	DIRR3	DIRECIONAL	ON NM	81.741.469	3,05%
Cycling consumption / civil construction	EVEN3	EVEN	ON NM	198.428.847	3,25%
Cycling consumption / civil construction	EZTC3	EZTEC	ON NM	94.318.339	4,49%
Cycling consumption / civil construction	GFSA3	GAFISA	ON NM	323.264.703	0,97%
Cycling consumption / civil construction	HBOR3	HELBOR	ON NM	65.595.478	0,46%
Cycling consumption / civil construction	JHSF3	JHSF PART	ON NM	299.019.330	5,61%
Cycling consumption / civil construction	LAVV3	LAVVI	ON NM	73.835.129	0,99%
Cycling consumption / civil construction	MRVE3	MRV	ON NM	298.436.955	9,24%
Cycling consumption / civil construction	TEND3	TENDA	ON NM	95.672.515	1,72%
Cycling consumption / civil construction	TRIS3	TRISUL	ON NM	71.349.496	0,85%
Financial	ALSO3	ALIANSCSONAE	ON NM	122.470.927	5,55%
Financial	BRML3	BR MALLS PAR	ON NM	828.273.884	17,77%
Financial	BRPR3	BR PROPERT	ON NM	175.439.087	3,59%
Financial	IGTI11	IGUATEMI S.A	UNT N1	179.981.368	8,68%
Financial	LOGG3	LOG COM PROP	ON NM	59.529.517	3,96%
Financial	MULT3	MULTIPLAN	ON N2	268.145.328	15,73%
Financial	SYNE3	SYN PROP TEC	ON NM	58.932.234	0,72%

Source: (B3, 2022)

.

9.4 G1 news headlines keywords

agir	demagogia	igualdade	mst	psc
agro	democracia	impeachment	mtst	psdb
assembleia	deputado	importacao	municipio	psol
ativismo	desembargador	imposto	nacao	pstu
audiencia	desenvolvimento	incerteza	nacionalismo	pt
bacen	df	industria	orcamento	ptb
banco central	dilma	inegibilidade	parlament	public
banco do brasil	diplomacia	inflacao	partido	pv
bndes	direita	inss	pasep	quorum
bolsonaro	direito	investimento	patriota	receita
burocracia	ditadura	iof	pcb	recessao
camara	divida	ipi	pcdob	referendo
campanha	eleicao	iptu	pco	reforma
candidato	eleicoes	ipva	pdt	regulacao
cassacao	eleito	irpj	petrobras	renda
centro	eleitoral	iss	pib	republica
cidadania	eletrobras	itbi	pis	resolucao
cide	emprego	itcmd	plebiscito	risco
civil	endividamento	itr	plenario	salario
classe	esquerda	judiciario	pmn	sancao
clausula	estado	juiz	pobreza	selic
clt	estadual	juro	poder	senador
cofins	estatal	lava-jato	politic	servidor
comissao	estatizacao	legislativ	populismo	sistema
congresso	estimulo	legislatura	posse	stf
conselho	executivo	lei	poupanca	subsidio
conservador	exportacao	liberdade	povo	superavit
constituicao	federacao	lobby	pp	tarifa
consumo	federal	lula	prefeito	taxa
correios	fgts	mandato	presidente	tcu
corrup	financ	manifestacao	previdencia	temer
credito	fiscal	mdb	privado	terrorismo
crescimento	fmi	medida provisoria	privatizacao	totalitarismo
cvm	gabinete	militar	programa social	transparencia
debate	golpe	ministerio	progressista	tribunal
decoro	governador	ministro	promotor	tributacao
decreto	governo	moeda	propina	tributo
deficit	guerra	monetaria	protesto	uniao
deflacao	icms	moratoria	prtb	vereador
delacao	ideologia	movimentos	psb	veto

Source: elaborated by the author.

REFERENCES

B3. **Índice Imobiliário (IMOB B3)**. 2022. Available from Internet: <https://www.b3.com.br/pt_br/market-data-e-indices/indices/indices-de-segmentos-e-setoriais/indice-imobiliario-imob.htm>.

BRASIL. Lei no. 1079 de 10 de abril de 1950. **Diário Oficial da União**, Imprensa Nacional, p. 5425, abril 1950. Available at <http://www.planalto.gov.br/ccivil_03/leis/L1079.htm>. Accessed: 2022-08-04.

BRASIL. Lei no. 4591 de 16 de dezembro de 1964. **Diário Oficial da União**, Imprensa Nacional, dezembro 1964. Available at <http://www.planalto.gov.br/ccivil_03/leis/l4591.htm>. Accessed: 2022-06-01.

BRASIL. Lei no. 9514 de 20 de dezembro de 1997. **Diário Oficial da União**, Imprensa Nacional, novembro 1997. Available at <http://www.planalto.gov.br/ccivil_03/leis/l9514.htm>. Accessed: 2022-06-01.

BRASIL. Lei no. 10931 de 2 de agosto de 2004. **Diário Oficial da União**, Imprensa Nacional, agosto 2004. Available at <http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2004/lei/l10.931.htm>. Accessed: 2022-06-01.

Brazil. **Constituição da República Federativa do Brasil: 49ª edição**. Edições Câmara, 2016. (Textos Básicos). ISBN 9788540203815. Available from Internet: <https://books.google.com.br/books?id=jeEhCQAAQBAJ>.

CALDAS, A. V. S. et al. Os efeitos da covid-19 sobre os desempenhos das ações dos setores da b3. v. 19, p. 15–28, jan 2021. Available from Internet: <http://periodicos.ufc.br/contextus/article/view/60146>.

CÂMARA, E. et al. **150 Termos para Entender Política**. Edições Câmara, 2020. (Cidadania). ISBN 9786587317090. Available from Internet: <https://books.google.com.br/books?id=1HP-DwAAQBAJ>.

DICKEY, D. A.; FULLER, W. A. Distribution of the estimators for autoregressive time series with a unit root. **Journal of the American statistical association**, Taylor & Francis, v. 74, n. 366a, p. 427–431, 1979.

FAMA, E. F. Random walks in stock market prices. **Financial Analysts Journal**, Routledge, v. 51, n. 1, p. 75–80, 1965. Available from Internet: <https://doi.org/10.2469/faj.v51.n1.1861>.

FAMA, E. F. Efficient market hypothesis: A review of theory and empirical work. **Journal of Finance**, v. 25, n. 2, p. 28–30, 1970.

GABRIEL, F. S.; RIBEIRO, R. B.; RIBEIRO, K. C. de S. HipÓteses de mercado eficiente: Um estudo de eventos a partir da reduÇÃo do ipi. **Revista de Gestão, Finanças e Contabilidade**, v. 3, n. 1, p. 36–52, 2013.

GRANGER, C. W. J. Investigating causal relations by econometric models and cross-spectral methods. **Econometrica**, v. 37, n. 3, p. 424–438, 1969.

KUMARA, H. V. U. D.; FERNANDO, P. N. D. Impact of political events on stock market return: Empirical evidence from sri lanka. 2020. Available from Internet: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3844273>.

KWIATKOWSKI, D. et al. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? **Journal of econometrics**, Elsevier, v. 54, n. 1-3, p. 159–178, 1992.

MACKINLAY, A. C. Event Studies in Economics and Finance. **Journal of Economic Literature**, v. 35, n. 1, p. 13–39, March 1997. Available from Internet: <https://ideas.repec.org/a/aea/jeclit/v35y1997i1p13-39.html>.

MARKOWITZ, H. M. **Portfolio Selection: Efficient Diversification of Investments**. Yale University Press, 1959. ISBN 9780300013726. Available from Internet: <http://www.jstor.org/stable/j.ctt1bh4c8h>.

MARQUES, T.; SANTOS, N. dos. Do political news affect financial market returns? evidences from brazil. **International Journal of Management, Accounting and Economics**, v. 3, n. 10, p. 545–571, 2016.

MARSHALL, J. Calcul des probabilités. by paul lévy. pp. 350. fr. 40. 1925. (gauthier-villars.). **The Mathematical Gazette**, Cambridge University Press, v. 13, n. 184, p. 214–214, 1926.

PARVEEN, S. Market efficiency of indian capital market: An event study around the announcement of results of lok sabha election 2019. **International Journal of Financial Research**, v. 12, n. 1, 2021. Available from Internet: <https://www.sciedu.ca/journal/index.php/ijfr/article/view/18214>.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, JSTOR, v. 52, n. 3/4, p. 591–611, 1965.

SIDNEY, S. A. Alexander, . price movements in speculative markets: Trends or random walks, no. 2. industrial management review, , . In: . [S.l.: s.n.], 1964.

SMALES, L. A. Better the devil you know: The influence of political incumbency on Australian financial market uncertainty. **Research in International Business and Finance**, v. 33, n. C, p. 59–74, 2015. Available from Internet: <https://ideas.repec.org/a/eee/riibaf/v33y2015icp59-74.html>.

Sá, P. S. de. **Empirical Strategy Softwares**. [S.l.]: GitHub, 2022. <https://github.com/ecompfin-ufrgs/political_news_real_estate>.