CORRELATING FACTORS OF U.S. PRESIDENTIAL SPEECHES WITH STOCK MARKET MOVEMENTS: A MACHINE LEARNING APPROACH.

Pablo Rees

Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Second draft formally presented with intention of fulfilment of the requirements for completion of the MCom Economics Programme (full thesis only) at Stellenbosch University.

Under the supervision of:
Dr Dawie van Lill

Stellenbosch University Department of Economics

July 2022

Abstract

Abstract to be written here. The abstract should not be too long and should provide the reader with a good understanding what you are writing about. Academic papers are not like novels where you keep the reader in suspense. To be effective in getting others to read your paper, be as open and concise about your findings here as possible. Ideally, upon reading your abstract, the reader should feel he / she must read your paper in entirety.

Keywords: S&P 500, Machine Learning, Natural Language Processing, Presidential Speeches, Time Series Econometrics

JEL classification G17

1. Introduction

References are to be made as follows: Fama & French (1997: 33) and Grinold & Kahn (2000) Such authors could also be referenced in brackets (Grinold & Kahn, 2000) and together Grinold & Kahn (2000). Source the reference code from scholar.google.com by clicking on "cite' ' below article name. Then select BibTeX at the bottom of the Cite window, and proceed to copy and paste this code into your ref.bib file, located in the directory's Tex folder. Open this file in Rstudio for ease of management, else open it in your preferred Tex environment. Add and manage your article details here for simplicity - once saved, it will self-adjust in your paper.

> I suggest renaming the top line after @article, as done in the template ref.bib file, to something more intuitive for you to remember. Do not change the rest of the code. Also, be mindful of the fact that bib references from google scholar may at times be incorrect. Reference Latex forums for correct bibtex notation.

To reference a section, you have to set a label using "\label' ' in R, and then reference it in-text as e.g. referencing a later section, Section 5.

Writing in Rmarkdown is surprizingly easy - see this website cheatsheet for a summary on writing Rmd writing tips.

Data

Notice how I used the curly brackets and dash to remove the numbering of the data section.

Discussion of data should be thorough with a table of statistics and ideally a figure.

In your tempalte folder, you will find a Data and a Code folder. In order to keep your data files neat, store all of them in your Data folder. Also, I strongly suggest keeping this Rmd file for writing and executing commands, not writing out long pieces of data-wrangling. In the example below, I simply create a ggplot template for scatter plot consistency. I suggest keeping all your data in a data folder.

2. Literature review

The aim of this project is to determine whether U.S Presidential speeches have predictive power over stock market movements. A positive result would be powerful and could add to the ability of traders to accurately predict stock market movements. Broadly, machine learning has been selected as the method of modelling and the S&P 500 index has been chosen as representative of the 'stock market'.

The undertaking of this project assumes 2 important conjectures. That there is a correlation between the linguistic factors[1] of speech employed by U.S. presidents in their speeches and U.S. stock market movements; and that the prominence of speech factors can be quantified by using machine learning algorithms. The following literature review defends the two conjectures, confirms the novelty of the project, surveys methods of data cleaning that may be applicable and to discover which machine learning methods are most appropriate for this project.

2.1. Conjecture defence

The following section references the findings of four peer-reviewed articles in defence of the conjectures made in section 1. More evidence exists and is reviewed in later sections but is not necessary to defend the conjectures made here.

2.1.1. FOMC speeches and U.S. financial market reactions

Hayo, Kutan & Neuenkirch (2008) performed a generalized autoregressive conditionally heteroscedastic (GARCH) analysis of the relationship between Federal Open Market Committee (FOMC) speeches and the U.S. financial market. The analysis was quantitative on the market side and qualitative on the speech side. They found that FOMC speeches influence trader behaviour, but that this effect is both asymmetrical (negative impacts were larger than positive impacts) and non-uniform across trader type (bond markets were affected far more than financial and forex markets). Further, more

[1] 'Linguistic factors' in this sense is intended to mean any and all patterns that can be detected in spoken language including verbiage, lexicon, tone, register, sentence length, word combination etc.

formal modes of communication have a larger impact on both returns and conditional variance and more prominent speakers have a greater impact on bond markets. Finally, they found that volatility in 3- and 6-month T-bills was reduced on the day of a speech.

It was commented that heteroskedasticity left something to be desired when assessing the effects of monetary shocks. Further, it was found that speeches alone were not sufficient to create significant effects for financial markets. It is important that news agencies propagate the news for market repercussions to occur. In a brief, informal interview with a few bond traders it was discovered that they tended to "read monetary policy statements and listen to speeches by Greenspan (Bernanke) themselves. Other types of communications are rather neglected and the traders tend to rely on newswire information" (Hayo et al., 2008:27). Further, it was noted that news articles fail to take a neutral stance on the contents of speeches implying that the market effect may be distorted by the sentiment of news agencies.

This article shows that the sentiments of communications influence the effect on markets. Thus, analysing sentiment is an important factor in an accurate appraisal of the relationship between speeches and markets. The finding that speeches need to be propagated by news agencies in order to have significant impacts on financial markets is counter to the hypothesis of this project but the FOMC is less publicly scrutinized than the U.S. president which may render this finding inconsequential to this project.

2.1.2. Political speeches and stock market outcomes

Maligkris (2017) demonstrates that the speeches given by U.S. presidential candidates directly influence the stock market, particularly during the early months of their campaigns. These speeches often contain information about potential presidents' positions on policy changes and public issues. Thus, they can affect investor sentiment and in turn the stock market. The employed methodology was to analyse transcripts of presidential candidate speeches from the American Presidency Project archives and the U.S. Government Publishing Project from the 2004-2016 period according to the index developed in Baker, Bloom & Davis (2016) (explained in section 2.3). He shows, using regression analysis that there is an increase in excess market returns of 26 basis points following candidate speeches, however the direction and magnitude of this effect varies between candidates. He goes on to examine whether the difference in effect is due to heterogenous speech content. Finally, it was demonstrated that speeches laden with economic information tend to boost stock returns while also reducing volatility. Speeches with a negative tone have the opposite effect. The long run effect of speeches is dependent on market conditions.

This paper indicates that there is a correlation between presidential candidates' speeches and stock movements. It is then reasonable to believe that there is a correlation between presidential speeches and stock market reactions. It also highlights that tone affects the relationship and thus that the sentiment of a speech is important. Further, the archives of the American Presidency Project and the U.S. Government Publishing Project are good sources for U.S. political speech transcripts – the potential predictive data.

2.1.3. Measuring economic policy uncertainty

Baker, Bloom & Davis (2016) develop an Economic Policy Uncertainty Index based on the frequency of articles containing a trio of terms in 10 leading U.S. newspapers. These terms were: ‘ “economic” or “economy” ; “uncertain” or “uncertainty”; and one or more of “congress”, “deficit”, “Federal Reserve”, ‘legislation”, “regulation”, or “White House” ’ (Baker et al., 2016:1594). The terms were selected over a period of 24 months during which more than 15 000 news articles were read by humans in an auditing process. The index proved to be quite accurate, spiking near the expected events, including wars, tight presidential elections, fiscal policy battles and terrorist activity.

They went on to show that their index had a strong relationship with other economic uncertainty measures and policy uncertainty measures. Further, congruence in uncertainty prediction was found between left- and right- leaning newspapers.

This article shows that language processing can be used to predict economic events, particularly economic uncertainty and economic policy uncertainty. However, the type of language processing proposed for this project differs significantly. While Baker et al. (2016) used man hours this project will use deep learning – the result of which is likely to be greater accuracy and reduced man hour expenditure, if it is correctly applied.

2.1.4. Hope, change and financial markets: can Obama’s words drive the market?

Sazedj & Tavares (2011) asked whether the speeches made by former U.S. President Barack Obama affected stock market prices. By regressing the event of a speech during Obama’s first 11 months in office on the daily excess returns of the Dow Jones, the S&P 500 and the NASDAQ they found that the event of a speech had a generally insignificant effect on daily excesses. However, by regressing key terms contained in 43 speeches given during the first 11 months of Obama’s presidency it was found that the content of speeches can significantly affect daily excess returns nearly uniformly across all three indices. Notably, the NASDAQ’s correlation to the content of speeches was weaker – indicating that technology markets may be less susceptible to presidential rhetoric.

This paper highlights the relationship between the content of a presidential speech and stock market returns. This study was correlational rather than predictive in nature.

2.1.5. Conjecture defence summary

As seen in section 2.1.2. and section 2.1.4. it is true that there is a correlation between the linguistic factors of speech employed by U.S. presidents in their speeches and U.S. stock market movements (Sazedj & Tavares, 2011; Maligkris, 2017). Further, section 2.1. and section 2.3. show that sentiment and other linguistic factors of speech can be quantified using machine learning methods (Hayo, Kutan & Neuenkirch, 2008; Baker, Bloom & Davis, 2016). Thus, both conjectures hold and further investigation is warranted. This is not the total of all evidence supporting these conjectures but is sufficient. Other articles reviewed here can be seen for further evidence.

2.2. Novelty

Searching 'sentiment analysis stock market' on Google Scholar yields mostly articles linking Twitter data and the stock market. Searching 'presidential speeches affect stock market' on Google Scholar yields articles on the relationship between presidential speeches and the stock market but none using Machine Learning techniques. Khedr, Salama & Yaseen (2017) describe related work as including relating news or twitter data to stock market behaviour and prices and relating financial news to stock prices, but do not mention presidential speeches. Searching 'machine learning S&P 500' on Google Scholar yields an array of articles using machine learning as a technical analysis tool but most of these use other financial indicators and are not NLP based – bar one article analysing the effects of former U.S. president, Donald Trump's tweets on the S&P 500 and the DJIA. Searching 'political speech machine learning' on Google Scholar yields articles that focus on political speeches but have no link to stock markets. Other searches yielded similar results, thus as far as can be told – this research is novel in nature.

2.3. Data cleaning methods for NLP

Katre (2019), in his analysis of Indian political speeches, uses Natural language Toolkit (NLTK) and string methods to remove punctuation, HTML tags and English stopwords[2] , as well as converting speeches to lowercase and tokenizing[3] them. Zubair & Cios (2015), before correlating the sentiment in Reuters articles with S&P 500 movements, clean their text data by tokenizing it with NLTK. Kinyua et al. (2021) clean their twitter data (tweets from then U.S. president, Donald Trump) by deleting all tweets on days when the stock trading was closed, deleting all tweets that only contained standard stopwords and deleting all tweets that only contained URLs. Khedr, Salama & Yaseen (2017) tokenize,

[2]'Stopwords' are words that commonly occur across all speech and therefore only create noise in the data. Some examples are 'the', 'it', 'they' and 'and'.

[3]'Tokenizing' refers to the splitting of words into tokens that have linguistic importance, for example the words 'terrorism', 'terrorist' and 'terror' may all be tokenized to 'terror'. Thus, the core concept of the word is captured while also simplifying the dataset.

standardize by converting to lowercase, remove stopwords from and stem[4] their textual data before processing the abbreviations (replacing abbreviations with the full phrase) and filtering out words that consist of two or less characters.

2.4. Machine learning methods

The following section looks first at the literature informing the sentiment analysis space and then at the literature around stock market prediction in order to determine methods that suit the intersection between the two.

2.4.1. Sentiment analysis methods

Ren, Wu & Liu (2019) analyse news articles at the sentence level by assigning a sentiment polarity (using software designed for the Chinese language) to each word followed by a sentiment score for each sentence in a document. Each document is then categorized and a sentiment score between -1 and 1 for all news for that day is generated. Zubair & Cios (2015) use the positive and negative valence categories from the Harvard General Enquirer (HGI) to assign each word in a Reuters news article a positive or negative label. They then sum the positives and negatives into a tuple and divide that tuple by the number of words in an article in order to create a vector that represents each news article. The vectors are organized into time series, normalized by dividing all vectors by the first vector, parsed through a Kalman filter and then correlated to S&P 500 returns using Pearson correlation (for both the positive and negative scalar in the vector). Kinyua et al., (2021) use the Valence Aware Dictionary for Sentiment Reasoning (VADER) to create a sentiment feature for former U.S. president Donald Trump's tweets which was then used as a regression feature in linear, decision tree and random forest regressions. Khedr, Salama & Yaseen (2017) use N-gram (n=2) to extract key phrases from their corpus of news text data, then term-frequency inverse-document-frequency is used to determine the importance of those phrases within the corpus, and finally use a naïve-Bayes classifier to assign positive and negative labels to each news document. Purevdagva et al. (2020) use a variety of features present in both data and metadata to predict fake political speech. Two features relevant to this project were 'speaker job' and 'context' (press, direct or social) which were labelled using universal serial encoders. For the actual sentiment analysis they used the linguistic inquiry and word count (LIWC) tool to categorize and count words into emotional, cognitive and structural text components. Various further attempts to extract sentiment from the text did not yield increased prediction accuracies. They go on to use an extra tree classifier for feature selection and then support vector machine (SVM), multilayer perceptron, convolutional neural network, decision trees, fasttext and bidirectional encoder representations from transformers (BERT) for prediction

[4]'Stemming' refers to the removal of suffixes to reduce the complexity of a dataset.

with highest accuracy coming from the SVM. Dilai, Onukevych & Dilay, (2018) use SentiStrength – an automatic sentiment analysis tool - to compare the sentiment in speeches between former U.S. president Donald Trump and former Ukrainian president Petro Poroshenko.

2.4.2. Stock market prediction methods

Ren, Wu & Liu (2019) use an SVM and fivefold cross validation approach to achieve a prediction accuracy of 98% when predicting fake news in political speech. They combined sentiment data and market indicators as their input data. Kinyua et al. (2021) use linear, decision tree and random forest regressions to predict S&P 500 and DJIA directional changes. Random forest regression performed best for both datasets. Khedr, Salama & Yaseen (2017) use open, high, low and close prices from their stock market data as features after labelling the change from the previous day as positive, negative or equal. Jiao & Jakubowicz (2017) extracted lag and window features from the S&P 500 and the global 8 index before running time series random forest, neural network and gradient boosted trees to predict movements of individual stocks in the S&P 500. Liu et al. (2016) used forward search feature selection to select features for SVM, naïve-Bayes, Gaussian discriminant analysis and logistic regression from a set of economic features including the crude oil daily return, currency exchange rates and major stock indices daily returns in order to forecast the S&P 500 movement.

2.5. Summary of literature review

Table 2.1 represents the literature review in a condensed format which allows for easy comparison of the data and methods used and resulting accuracies. Table 2.2 represents the metadata, linked through 'Paper number' for Table 2.1.

Table 2.1: Condensed literature review

Paper number	ML Method	Cleaning method	Data type	Index predicted	Max ccuracy
1	Support vector machine with fivefold cross validation		Daily online stock reviews	SSE 50	0.98
1	Support vector machine with rolling windows				0.9
1	Logistic regression with fivefold cross validation				0.87

Paper number	ML Method	Cleaning method	Data type	Index predicted	Max ccuracy
2	Non-ML:	Tokenized and mined using the Harvard General Inquirer dictionary	Reuters textual data	S&P 500	Corr: -0.91
3	Random forest	Date validation, stop word and URL removal	Tweets	INDU	0.98
3	Decision tree				0.97
3	Logistic regression				0.81
3	Random forest			S&P 500	0.92
3	Decision tree				0.88
3	Logistic regression				0.77
4	N-gram, TF-IDF, Naïve Bayes, K-NN	Tokenize, stopwords, stemming, abbreviation processing	News articles and financial reports	Three tech stocks	0.9
5	TS logistic regression	Feature extraction: lags, window features	Financial indicators	Individual S&P 500 stocks	0.79
5	TS random forest				0.78
5	TS neural network				0.78
5	TS gradient boosting				0.78
6	Logistic regression	Date validation forward search feature selection	Market indices, exchange rates	S&P 500	0.61
6	Gaussian discriminant analysis				0.61
6	Naïve Bayes				0.6
6	Linear SVM				0.6
6	Radial Basis Function SVM				0.63
6	Polynomial SVM				0.6
7	SVM	Feature extraction and feature selection	Political speeches and metadata	Liar dataset	0.74
7	Multilayer perceptron				0.55
7	Convolutional neural network				0.61

Paper number	ML Method	Cleaning method	Data type	Index predicted	Max ccuracy
7	Fasttext				0.66
7	BERT				0.66

Table 2.2: Table 2.1 metadata

Paper number	Title	Citation	DOI
1	Forecasting Stock Market Movement Direction Using Sentiment Analysis and Support Vector Machine	(Ren, Wu & Liu, 2019)	10.1109/JSYST.2018.279446
2	Extracting News Sentiment and Establishing its Relationship with the S&P 500 Index	(Zubair & Cios, 2015)	10.1109/JSYST.2018.279446
3	An analysis of the impact of President Trump's tweets on the DJIA and S&P 500 using machine learning and sentiment analysis	(Kinyua et al., 2021)	10.1016/j.jbef.2020.100447
4	Predicting Stock Market Behaviour using Data Mining Technique and News Sentiment Analysis	(Khedr, Salama & Yaseen, 2017)	10.5815/ijisa.2017.07.03
5	Predicting Stock Movement Direction with Machine Learning:an Extensive Study on S&P 500 Stocks	(Jiao & Jakubowicz, 2017	10.1109/BigData.2017.82585
6	Forecasting S&P 500 Stock Index Using Statistical Learning Models	(Liu et al., 2016)	10.4236/ojs.2016.66086
7	A machine-learning based framework for detection of fake political speech	(Purevdagva et al., 2020)	10.1109/BigDataSE50710.20

3. Data collection and cleaning process

The following is a description of the data collection and cleaning process and the feature extraction process. It begins by explaining the three datasets collected for this study, namely control, meta and test data. Control refers to autoregressive features extracted from the S&P 500, meta refers to S&P 500 adjacent financial data and test refers to vectorised presidential speeches which are the focus of this study. The section begins by elaborating on the collection and cleaning steps for each of these datasets. The next subsection describes the feature engineering methods (sentiment analysis and word vectorization) that were used to draw variables with potentially strong signals from the text data. The final subsection describes the reasoning, methods and results used in a time series analysis of the financial time series (S&P 500) data in order to extract useable autoregressive features from it.

3.1. Data

Three types of data were gathered for this project, namely: presidential speeches/text data (all the transcripts from the Presidency Project website including formal and informal and written and verbal addresses), a history of the S&P 500 index and a history of 6 S&P 500 adjacent price histories – 2 sets of financial data.

The S&P 500 index and the metadata was downloaded from Yahoo Finance while the presidential speeches were scraped from the American Presidency Project (Yahoo Finance, n.d.; Woolley & Peters, n.d.). The S&P 500 is downloaded using the 'fin_data_downloader' Python module and will always download the entire history of the S&P 500 index at a daily interval. The same goes for the metadata. The presidential speeches on the other hand were scraped using the 'WebScrapeAndClean' Python module which will also always scrape the entire corpus available on the American Presidency Project website. This allows for perfectly updated data to be collected at any time.

The S&P 500 data contains seven variables, namely: 'Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', and 'Volume'. 'Open' records the opening price for each day, while 'Close' records the closing price. 'High' records the daily high and 'Low' the daily low. 'Volume' records the dollar amount of stock traded in the S&P 500 on each day and 'Adj Close' is irrelevant because it never differs from 'Close'. Notably, opening and closing prices are only differentiated between after April 20th 1982 while volume was only recorded after 1950.

After scraping, the Presidency Project data contains five variables. These are 'Type' which records the category and sub-category of each speech, 'Name' which records the title and name of the main speaker, 'Date' which records the date the speech occurred on, 'Title' which records the title of each speech and 'Transcript' which contains the raw HTML transcript of the speech.

3.2. Cleaning

The S&P 500 index did not require any cleaning after download, besides the removal of the redundant 'Adj Close' variable. Conversely, the presidential speeches required extensive cleaning. Text that has been web-scraped contains HTML tags[5] , thus the first step was to remove the HTML tags from the text. Similarly, the test contained reactions from the crowds listening to the speeches[6] ,

[5]HTML tags include paragraphing and spacing indicators for computers to interpret such as '

[6]These were tags such as 'Laughter' and 'Applause'.

which were also removed. Next, the transcripts were converted to lower case and the question sections removed. Next a ‘No Stops Transcript’ variable was created by removing the stopwords[7] in the Natural Language Tool Kit (NLTK) stop words dictionary from the clean transcript. The original transcripts were also kept. Table A1 in Appendix A shows the shape of the data at this point. The cleaning of both the speech data and both financial datasets was done in their original collection scripts, i.e. the ‘fin_data_downloader’ Python script and the ‘WebScrapeAndClean’ Python script respectively.

3.3. Text feature engineering

To increase the strength of the signals coming out of the speech data and reduce the computational power required to run the machine learning algorithms – feature engineering is required. Feature engineering refers to the emphasis of certain signals within the available data and creation of new variables which capture these signals. It results in the addition of extra features (variables) to the dataset. There are three broad methods of feature engineering: feature selection, feature extraction and the creation of new features (Géron, 2019:27). In this section the creation of new features occurred and took the form of sentiment analysis and word vectorization for the text data.

3.3.1. Sentiment analysis

Two versions of sentiment analysis were carried out. First, NLTK’s Valence Aware Dictionary for Sentiment Reasoning (VADER) was used to extract a sentiment analysis tuple, in this instance containing four scores, namely, negativity, neutrality, positivity and compound. VADER is a lexicon based system of sentiment analysis (Sohangir, Petty & Wang, 2018). Each of negativity, neutrality and positivity describe a transcript independently of the other scores while compound describes a transcript comprehensively (combining the other three scores). VADER, when compared with alternative NLP feature extraction techniques has performed better on social media transcripts and generalized better to other areas (Hutto & Gilbert, 2014; Elbagir & Yang, 2019). VADER has been used in vectorizing text relative to financial data in Pano & Kashef (2020) for Bitcoin price predictions, Agarwal (2020) which found a strong correlation between VADER sentiment scores and stock price changes and Sohangir et al. (2018) which shows the superiority of lexicon based approaches (specifically VADER) over ML approaches for sentiment classification.

Next, the TextBlob package’s sentiment analysis tool was used. This yields two scores (in a tuple) describing the sentiment of a transcript, namely, a polarity score (ranging from -1 to 1) and a subjectivity score (ranging from 0 to 1). Polarity describes whether the emotions expressed are negative or positive, with lower scores indicating negativity while subjectivity indicates the extent of the usage

[7]Stopwords are words that commonly occur in the English language and are therefore unlikely to contain any sort of signal and thus constitute only noise.

of subjective words (Chudy, n.d.). Biswas, Sarkar, Das, Bose & Roy (2020) use Textblob sentiment scores in their analysis of the effects of Covid-19 on stock markets. Textblob was also tested in Sohangir et al. (2018) but did not perform as well as VADER although it did outperform ML methods in terms of area under the curve (AUC) scores. It is expected that the VADER sentiment tuple will outperform Textblob's sentiment tuple as a predictor of the S&P 500 data.

3.3.2. Speech vectorization

Converting human readable text into machine readable data requires the conversion from words to numbers. This vectorization can be done in various ways but in order to preserve the meaning of the texts the Word2Vec and Doc2Vec Python packages provided by Gensim were used (Řeh uřek & Sojka, 2010). However, Word2Vec was originally published in two papers by Mikolov, Chen, Corrado & Dean (2013a,b) while Doc2Vec was suggested by Le & Mikolov (2014).

3.3.2.1. Word2Vec. Gensim's Word2Vec Python package's skip-gram model is used for Word2Vec vectorization. Using the full vocabulary of words in a corpus of speeches and one-hot encoding for word vectors, Word2Vec trains a single hidden layer neural network to predict words based on the words around them. The parameters used for training are available in Appendix A.

The model contextually embeds each word in the entire corpus of speeches by running the speeches through the single hidden layer predictive neural network (NN). The NN is provided with the pseudo-task of predicting the word of focus from the words surrounding it each time it occurs in the corpus. Thus, a hidden layer is trained to contain the information that contextually embeds each word in the corpus. These hidden layer vectors (rather than the predictions) are the real output of the Word2Vec model. Words that appear in similar contexts throughout the corpus will have similar representational vectors (hidden layers) and thus can be said to have similar meanings in the corpus(Mikolov et al., 2013a,b).

An example phrase might be 'The quick brown fox jumps'. If the word 'brown' is the focus word, the words 'quick' and 'fox' would be fed into the neural network which would then be trained to map the input to the word 'brown'. Doing this for every instance of 'brown' in a corpus creates a hidden layer that contains all the contextual information required to predict the word 'brown' in the given corpus. Figure 1 depicts this process.

The model is extremely good at relating words that appear in similar contexts to each other. For example, when asked for the three words most similar to 'oil' the model trained on the presidential speeches corpus returns 'crude', 'gas' and 'petroleum'. The input 'gold' returns 'silver', 'bullion' and 'coin'; whilst 'virus' returns 'covid19', 'infection' and 'pandemic'. In this study the representational vectors each contain 200 elements (because the hidden layers were set to contain 200 elements). Thus each word is described by a vector containing 200 elements. In order to create a similarly sized vector for each speech, the vectors describing all the words in each speech were averaged. Thus each speech has been reduced to a 200 element vector averaging the contextual embeddings of each word contained therein. This averaging technique was also used in Vargas, de Lima & Evsukoff (2017) and Qin & Ji (2018). However, this method fails to preserve word order in a document vector (Le & Mikolov, 2014).

Notably, the algorithm also makes room for phrases such as 'asset backed' or 'short selling' – which are considered as 'assetbacked' and 'shortselling'. When two words that occur irregularly in the vocabulary occur together frequently the algorithm interprets them as a phrase and treats them as such. There is, however, only room for two words in each phrase if the algorithm has only been executed once – which is the case here.

Word2vec is used by (Shi, Zhao & Xu, 2019) for the improvement of sentiment classification which implies that the final vectors in this study may contain sentiment information. It is also used by Vargas et al. (2017) and Qin & Ji (2018)in their predictive modelling of S&P 500 changes based on twitter data. Vargas et al. (2017) also used 200 element vectors while Qin & Ji (2018) used 300 element vectors. Both studies achieved prediction accuracy around 65%.

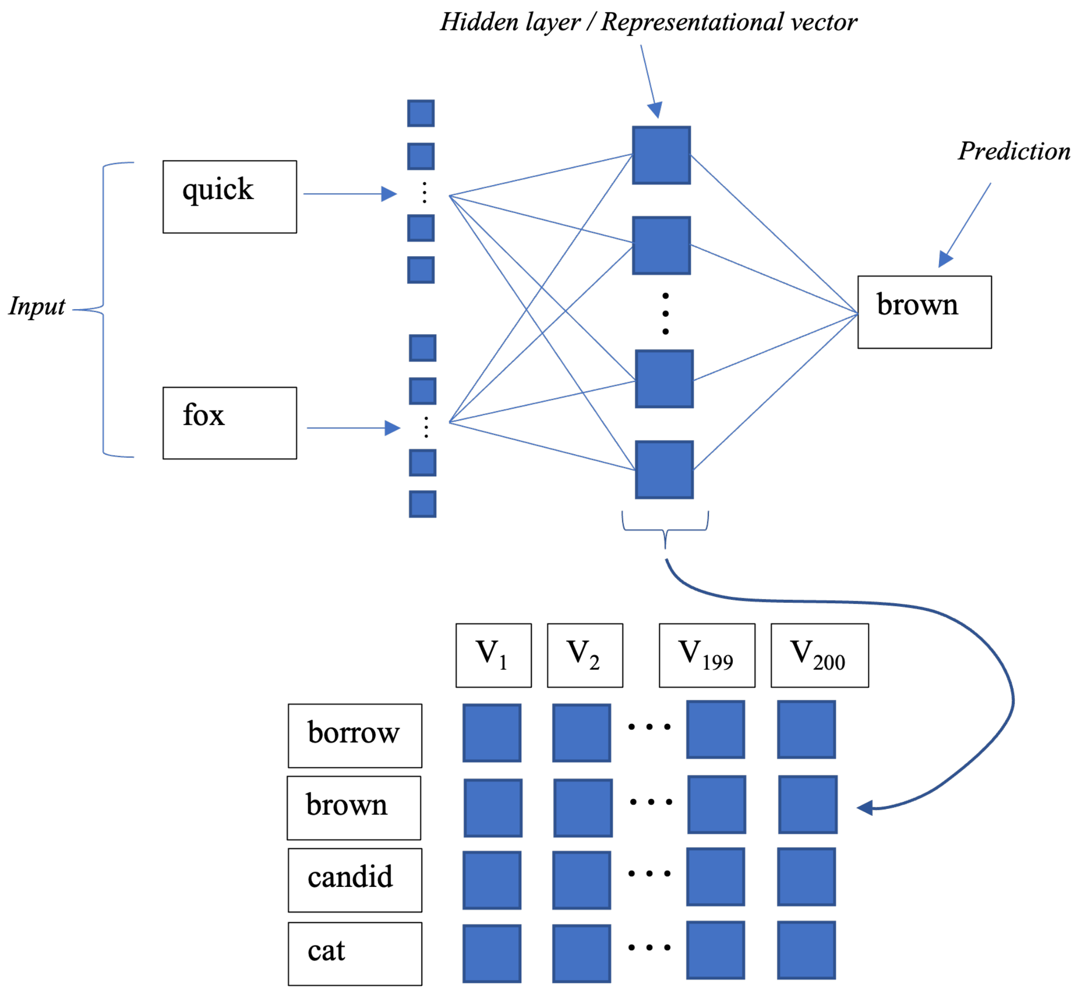

Figure 3.1: Word2Vec model

3.3.2.2. Doc2Vec. An alternative method to creating a vector representation of a sentence or document is Doc2Vec. This method is similar to the Word2Vec method described above but includes an additional floating vector when performing its pseudo-task. This vector is maintained across every word prediction task within a document and subjected to training for each instance of every word. Thus each document in a corpus is assigned a single comprehensive vector that embeds it in the corpus. Embedding within the broader corpus is maintained by tagging each document with a unique tagging phrase. This method outperforms other methods such as bag-of-words for text representations (Le & Mikolov, 2014).

There are two implementations of Doc2Vec, namely Distributed Bag of Words (DBOW) and Distributed Memory (DM) (Sohangir et al., 2018). Both have been used in this study. In both cases each document is assigned a paragraph tag which represents the paragraph to the Doc2Vec algorithm. DM Doc2vec does this in the same way that a word represents itself to the Word2Vec algorithm. For every word in a document, its paragraph tag is passed to the Doc2Vec algorithm along with the words relevant to the current prediction pseudo-task. Back propagation is employed in the same manner as in Word2Vec except that the document tag vector is optimized for separately from the word vectors. This document tag vector is the relevant output in this case. Because a single vector of weights is created for each document, this vector should constitute a vectorized representation of the document as a whole. Figure 2 depicts the architecture of DM Doc2Vec algorithms. Alternatively, DBOW Doc2Vec algorithms ignore the context of a word and use random sampling to predict words from a paragraph. Figure 3 depicts the architecture of a DBOW Doc2Vec algorithm.

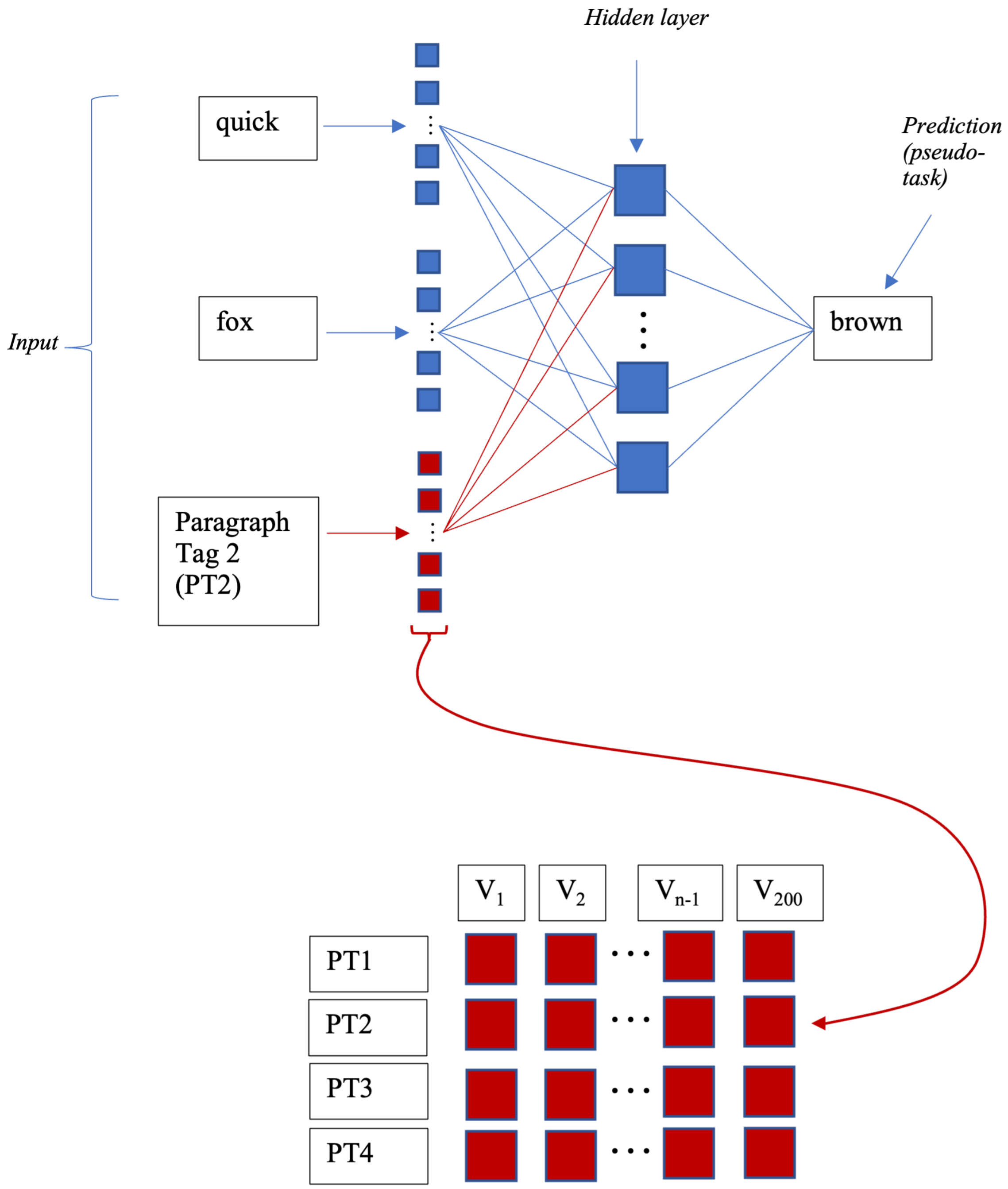

Figure 3.2: Doc2Vec model - distributed memory architecture: dm = 1

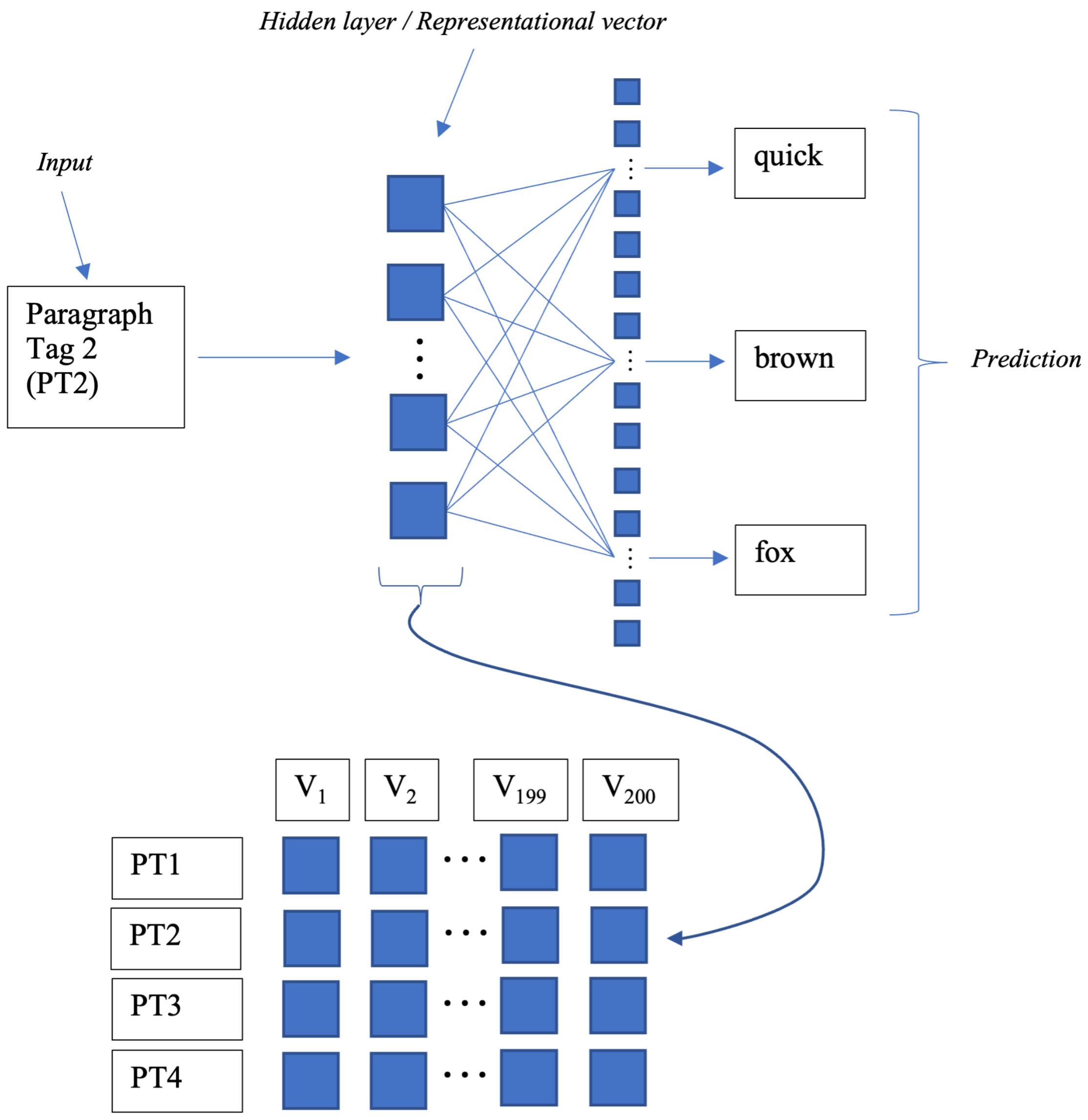

Figure 3.3: Doc2Vec model - distributed bag of words architecture: dm = 0

!!!!!!!!WHERE HAS DOC2VEC BEEN USEFUL FOR FINANCIAL PREDICTIONS!!!!!!!!!!

3.4. S&P 500 time series analysis

As part of feature extraction - econometric time series analysis has been run on the S&P 500 data. The aim of this analysis was to find the linear model of best fit to the S&P 500 data and then include relevant autoregressive variables in the final dataset under the assumption that they will be relevant in the highly non-linear ML models. An initial analysis was done on the S&P 500 data after April 20th 1982 because opening and closing prices are differentiated between from that date onwards.

This analysis found a large array of significant variables – more than half the days of the month, the month of September, a few specific years, 10 non-consecutive lags and the first lag of volume of trade. The significance of these variables, particularly the days of the month were difficult to explain rationally. However, they did indicate persistence of volatility. This volatility persistence and the fact that volume is a strong indicator of absolute price change encouraged a second round of analysis that was done on all data following the initial recording of volume in 1950. This second round of analysis was focussed on finding an autoregressive conditional heteroskedasticity (ARCH) model that fit the data well. The analysis concluded with the selection of an ARMA (1,0) GARCH (1,1) model. Thus, the core variables selected for inclusion in the final data set are the first lag of standardized volume, the first lag of daily percentage change, the first residual of daily percentage change (DPC) predicted by an ARMA (1,0) model and the first lag of variance of the DPC.

3.4.1. 1982 onwards

Running time series analysis on the daily percentage change in the S&P 500 after April 20th 1982 has revealed 10 significant non-consecutive lags. The 'Daily percentage change 'variable was created by taking the percentage difference between the 'Close' and 'Open' variables for each day of the data. Before April 20th 1982 the 'Open' and 'Close' variables hold identical values, hence the time series analysis only being run after that date. As can be seen in Figure 1, 'Daily percentage change' is naturally stationary. This is supported by the results of an Augmented Dickey-Fuller unit root test which indicated that no unit root is present in the data. Running a partial autocorrelation function revealed 10 significant lags (these were lagged by 1, 2, 4, 12, 15, 16, 18, 27, 32, and 34 periods). Regressing 'Daily percentage change' on all 10 significant lags reinforced the finding by yielding significance above the 95% confidence interval for all 10 lags. Further, regressing the daily percentage change on weekday, monthday, month and year categorical variables yielded the significant correlations depicted in Table 1 (none of the weekdays were significant). Regressing on the categorical variables and the lags simultaneously yielded similar results with increased significance for 2002 (to the 99% level) and 2008 (to the 99,9% level) and the addition of 2018 (significant at the 95% level), further an extra 4 monthdays were deemed significant - bringing the total to 21 (out of 31) significant monthdays, finally, some of the significance levels on the lags changed. These statistics are depicted in Table 2. Adding a normalized (distributed standard normal) volume variable lagged by one period to the regression yields a significance at the 99% level on the lagged volume variable and alters the significance on the year variables as reported in Table 3.

While the years 2002 and 2008 are justifiable as significant because of the financial crashes that happened in each of those years (2002 – dot com bubble and 2008 housing bubble) and September is also relatable to the housing bubble; it is difficult to rationally justify the monthday variables as significant regardless of their quantitative significance. The high number of lags is also difficult to justify and implies rather that there may be persistence of volatility. Thus the following section focusses on the modelling of volatility over a time period that maximises the inclusion of volume statistics in the data. All of the analysis reported in this section was done in R using the packages ‘stats’, ‘dplyr’, ‘urca’, ‘tidyverse’, ‘ggplot2’ and ‘fixest’ (Marais, 2022).

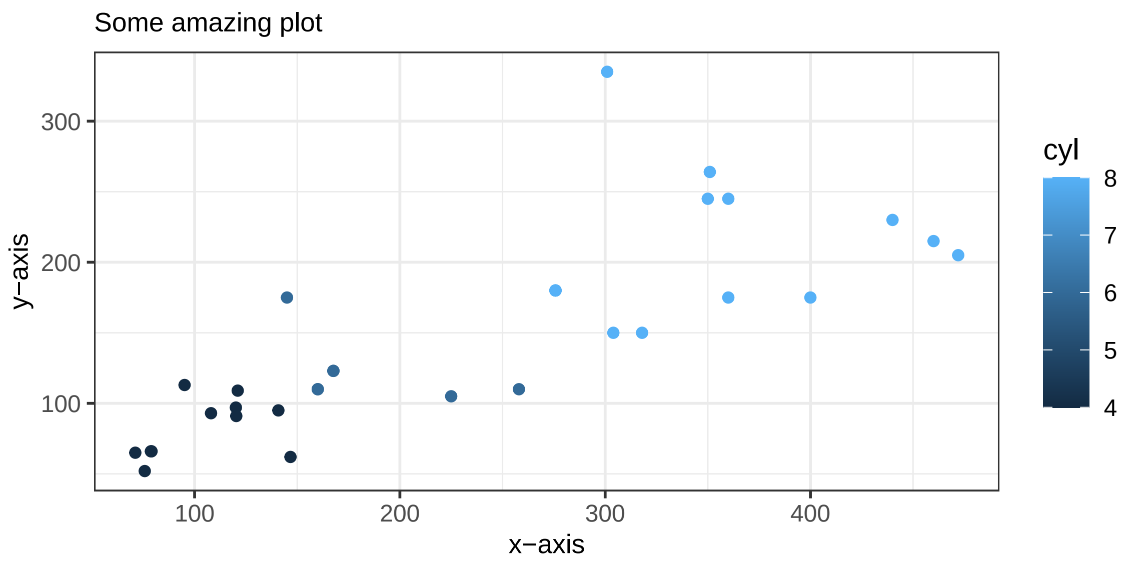

Figure 3.4: Caption Here

To make your graphs look extra nice in latex world, you could use Tikz device. Replace dev - ‘png’ with ‘tikz’ in the chunk below. Notice this makes the build time longer and produces extra tex files - so if you are comfortable with this, set your device to Tikz and try it out:

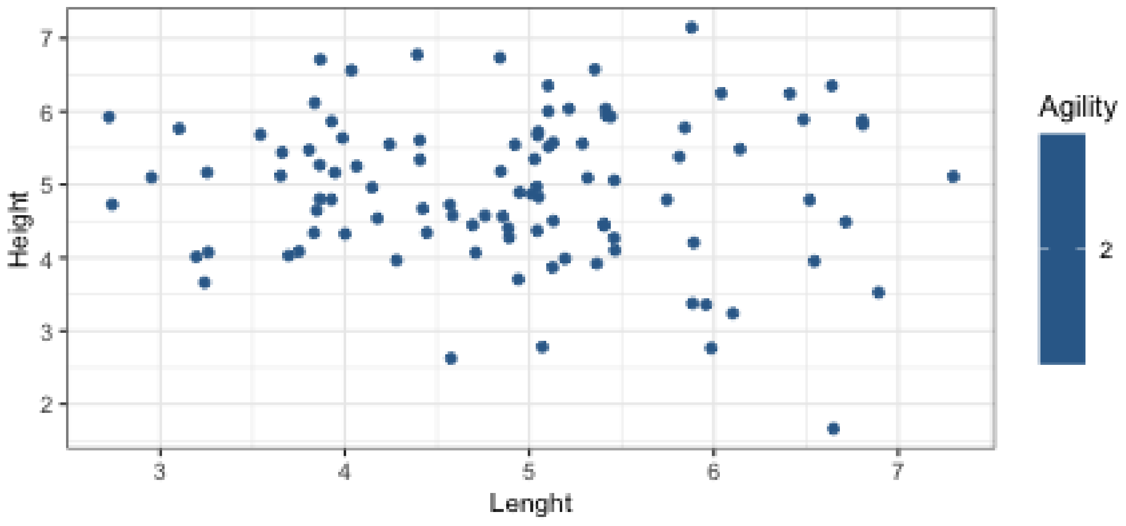

Figure 3.5: Caption Here

To reference the plot above, add a "\label'' after the caption in the chunk heading, as done above. Then reference the plot as such: As can be seen, Figures 3.4 and 3.5 are excellent, with Figure 3.5 being particularly aesthetically pleasing due to its device setting of Tikz. The nice thing now is that it correctly numbers all your figures (and sections or tables) and will update if it moves. The links are also dynamic.

I very strongly suggest using ggplot2 (ideally in combination with dplyr) using the ggtheme package to change the themes of your figures.

Also note the information that I have placed above the chunks in the code chunks for the figures. You can edit any of these easily - visit the Rmarkdown webpage for more information.

4. Splitting a page

You can also very easily split a page using built-in Pandoc formatting. I comment this out in the code (as this has caused issues building the pdf for some users - which I presume to be a Pandoc issue), but you are welcome to try it out yourself by commenting out the following section in your Rmd file.

5. Methodology

5.1. Subsection

Ideally do not overuse subsections. It equates to bad writing.[8]

5.2. Math section

Equations should be written as such:

$$\beta = \sum_{i=1}^{\infty} \frac{\alpha^2}{\sigma_{t-1}^2} \tag{5.1}$$
$$\int_{x=1}^{\infty} x_i = 1$$

[8]This is an example of a footnote by the way. Something that should also not be overused.

If you would like to see the equations as you type in Rmarkdown, use $ symbols instead (see this for yourself by adjusted the equation):

$$\beta = \sum_{i=1}^{\infty}\frac{\alpha^2}{\sigma_{t-1}^2} \int_{x=1}^{\infty} x_i = 1$$

Note again the reference to equation 5.1. Writing nice math requires practice. Note I used a forward slashes to make a space in the equations. I can also align equations using **&**, and set to numbering only the first line. Now I will have to type "begin equation'' which is a native LaTeXcommand. Here follows a more complicated equation:

$$\begin{aligned}
y_t &= c + B(L)y_{t-1} + e_t \qquad (5.2)\\
e_t &= H_t^{1/2} z_t; \quad z_t \sim N(0, I_N) \quad \& \quad H_t = D_t R_t D_t \\
D_t^2 &= \sigma_{1,t}, \dots, \sigma_{N,t} \\
\sigma_{i,t}^2 &= \gamma_i + \kappa_{i,t} v_{i,t-1}^2 + \eta_i \sigma_{i,t-1}^2, \quad \forall i \\
R_{t,i,j} &= diag(Q_{t,i,j}{}^{-1}).Q_{t,i,j}.diag(Q_{t,i,j}^{-1}) \\
Q_{t,i,j} &= (1 - \alpha - \beta)\bar{Q} + \alpha z_t z_t' + \beta Q_{t,i,j}
\end{aligned}$$

Note that in 5.2 I have aligned the equations by the equal signs. I also want only one tag, and I create spaces using "quads''.

See if you can figure out how to do complex math using the two examples provided in 5.1 and 5.2.

6. Results

Tables can be included as follows. Use the *xtable* (or kable) package for tables. Table placement = H implies Latex tries to place the table Here, and not on a new page (there are, however, very many ways to skin this cat. Luckily there are many forums online!).

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	21.00	6.00	160.00	110.00	3.90	2.62	16.46	0.00	1.00	4.00	4.00
2	21.00	6.00	160.00	110.00	3.90	2.88	17.02	0.00	1.00	4.00	4.00
3	22.80	4.00	108.00	93.00	3.85	2.32	18.61	1.00	1.00	4.00	1.00
4	21.40	6.00	258.00	110.00	3.08	3.21	19.44	1.00	0.00	3.00	1.00
5	18.70	8.00	360.00	175.00	3.15	3.44	17.02	0.00	0.00	3.00	2.00

Table 6.1: Short Table Example

To reference calculations **in text**, *do this:* From table 6.1 we see the average value of mpg is 20.98.

Including tables that span across pages, use the following (note that I add below the table: “continue on the next page’ ’). This is a neat way of splitting your table across a page.

Use the following default settings to build your own possibly long tables. Note that the following will fit on one page if it can, but cleanly spreads over multiple pages:

Table 6.2: Long Table Example

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21.00	6.00	160.00	110.00	3.90	2.62	16.46	0.00	1.00	4.00	4.00
21.00	6.00	160.00	110.00	3.90	2.88	17.02	0.00	1.00	4.00	4.00
22.80	4.00	108.00	93.00	3.85	2.32	18.61	1.00	1.00	4.00	1.00
21.40	6.00	258.00	110.00	3.08	3.21	19.44	1.00	0.00	3.00	1.00
18.70	8.00	360.00	175.00	3.15	3.44	17.02	0.00	0.00	3.00	2.00
18.10	6.00	225.00	105.00	2.76	3.46	20.22	1.00	0.00	3.00	1.00
14.30	8.00	360.00	245.00	3.21	3.57	15.84	0.00	0.00	3.00	4.00
24.40	4.00	146.70	62.00	3.69	3.19	20.00	1.00	0.00	4.00	2.00
22.80	4.00	140.80	95.00	3.92	3.15	22.90	1.00	0.00	4.00	2.00
19.20	6.00	167.60	123.00	3.92	3.44	18.30	1.00	0.00	4.00	4.00
17.80	6.00	167.60	123.00	3.92	3.44	18.90	1.00	0.00	4.00	4.00
16.40	8.00	275.80	180.00	3.07	4.07	17.40	0.00	0.00	3.00	3.00
17.30	8.00	275.80	180.00	3.07	3.73	17.60	0.00	0.00	3.00	3.00
15.20	8.00	275.80	180.00	3.07	3.78	18.00	0.00	0.00	3.00	3.00
10.40	8.00	472.00	205.00	2.93	5.25	17.98	0.00	0.00	3.00	4.00
10.40	8.00	460.00	215.00	3.00	5.42	17.82	0.00	0.00	3.00	4.00
14.70	8.00	440.00	230.00	3.23	5.34	17.42	0.00	0.00	3.00	4.00
32.40	4.00	78.70	66.00	4.08	2.20	19.47	1.00	1.00	4.00	1.00
30.40	4.00	75.70	52.00	4.93	1.61	18.52	1.00	1.00	4.00	2.00
33.90	4.00	71.10	65.00	4.22	1.83	19.90	1.00	1.00	4.00	1.00

Continued on next page

Table 6.2: Long Table Example

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	21.50	4.00	120.10	97.00	3.70	2.46	20.01	1.00	0.00	3.00	1.00
	15.50	8.00	318.00	150.00	2.76	3.52	16.87	0.00	0.00	3.00	2.00
	15.20	8.00	304.00	150.00	3.15	3.44	17.30	0.00	0.00	3.00	2.00
	13.30	8.00	350.00	245.00	3.73	3.84	15.41	0.00	0.00	3.00	4.00
	19.20	8.00	400.00	175.00	3.08	3.85	17.05	0.00	0.00	3.00	2.00
	27.30	4.00	79.00	66.00	4.08	1.94	18.90	1.00	1.00	4.00	1.00
	26.00	4.00	120.30	91.00	4.43	2.14	16.70	0.00	1.00	5.00	2.00
	30.40	4.00	95.10	113.00	3.77	1.51	16.90	1.00	1.00	5.00	2.00
	15.80	8.00	351.00	264.00	4.22	3.17	14.50	0.00	1.00	5.00	4.00
	19.70	6.00	145.00	175.00	3.62	2.77	15.50	0.00	1.00	5.00	6.00
	15.00	8.00	301.00	335.00	3.54	3.57	14.60	0.00	1.00	5.00	8.00
	21.40	4.00	121.00	109.00	4.11	2.78	18.60	1.00	1.00	4.00	2.00

6.1. Huxtable

Huxtable is a very nice package for making working with tables between Rmarkdown and Tex easier.

This cost some adjustment to the Tex templates to make it work, but it now works nicely.

See documentation for this package here. A particularly nice addition of this package is for making the printing of regression results a joy (see here). Here follows an example:

If you are eager to use huxtable, comment out the Huxtable table in the Rmd template, and uncomment the colortbl package in your Rmd's root.

Note that I do not include this in the ordinary template, as some latex users have complained it breaks when they build their Rmds (especially those using tidytex - I don't have this problem as I have the full Miktex installed on mine). Up to you, but I strongly recommend installing the package manually and using huxtable. To make this work, uncomment the *Adding additional latex packages* part in yaml at the top of the Rmd file. Then comment out the huxtable example in the template below this line. Reknit, and enjoy.

Table 6.3: Regression Output

	Reg1	Reg2	Reg3
(Intercept)	-2256.361 ***	5763.668 ***	4045.333 ***
	(13.055)	(740.556)	(286.205)
carat	7756.426 ***		7765.141 ***
	(14.067)		(14.009)
depth		-29.650 *	-102.165 ***
		(11.990)	(4.635)
N	53940	53940	53940
R2	0.849	0.000	0.851

*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

FYI - R also recently introduced the gt package, which is worthwhile exploring too.

7. Lists

To add lists, simply using the following notation

- This is really simple
  - Just note the spaces here - writing in R you have to sometimes be pedantic about spaces...
- Note that Rmarkdown notation removes the pain of defining LaTeXenvironments!

8. Conclusion

I hope you find this template useful. Remember, stackoverflow is your friend - use it to find answers to questions. Feel free to write me a mail if you have any questions regarding the use of this package. To cite this package, simply type citation("Texevier") in Rstudio to get the citation for Katzke (2017) (Note that uncited references in your bibtex file will not be included in References).

References

10 Fama, E.F. & French, K.R. 1997. Industry costs of equity. *Journal of financial economics*. 43(2):153–193.

Grinold, R.C. & Kahn, R.N. 2000. Active portfolio management.

Katzke, N.F. 2017. *Texevier: Package to create elsevier templates for rmarkdown*. Stellenbosch, South Africa: Bureau for Economic Research.

Appendix

Appendix A

Some appendix information here

Appendix B