ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ &
ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ
ΠΛΗΡΟΦΟΡΙΑΣ

ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ

SENTIMENT ANALYSIS AND PREDICTION USING TWITTER DATA

ΑΝΑΛΥΣΗ ΕΥΑΙΣΘΗΣΙΑΣ ΚΑΙ ΠΡΟΒΛΕΨΗΣ ΜΕ ΤΗ ΧΡΗΣΗ ΔΕΔΟΜΕΝΩΝ ΤΟΥ TWITTER

ΕΡΓΑΣΙΑ
Που υποβλήθηκε στο Τμήμα Στατιστικής
του Οικονομικού Πανεπιστημίου Αθηνών
ως μέρος των απαιτήσεων για την απόκτηση
Μεταπτυχιακού Διπλώματος
Συμπληρωματικής Ειδίκευσης στη Στατιστική
Μερικής Παρακολούθησης

Νικόλαος Φίλιππος Καμπουρίδης

Αθήνα
Οκτώβριος 2018

Acknowledgements

First I would like to thank my advisor Professor Nikolaos Demiris for his useful guidance and support. But mainly i have to thank him for the idea he has to give me such a hot topic to analyze and represent in my thesis.
Also i want to thank my parents, my brother and all of my friends for their continuous support.

Ευχαριστίες

Πρώτα θα ήθελα να ευχαριστήσω τον σύμβουλο και καθηγητή μου Νικόλαο Δεμίρη για τη χρήσιμη καθοδήγηση και υποστήριξη του. Αλλά κυρίως οφείλω να τον ευχαριστήσω για την ιδέα που μου έδωσε για να αναλύσω και να παρουσιάσω ένα τόσο σημαντικό θέμα στη διατριβή μου.
Επίσης, θέλω να ευχαριστήσω τους γονείς μου, τον αδελφό μου και όλους τους φίλους μου για τη συνεχή υποστήριξή τους.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Intro

In recent years, the social media and in particular Twitter, have invested heavily in the field of politics [book 1, ref1]. The association of politics with twitter as a campaign tool of politicians has been the subject of research and annotation. The research is not only confined to politicians and how they use the twitter but extends to the reactions of the citizens and the use of the twitter as a means of expression in political activities especially in election periods. In Greece, of course, the possibilities that it may have in this field have not yet been fully understood. Contrary to other countries, politicians support their campaign in Twitter, and citizens express their positive or negative attitude towards it.
Twitter emphasizes posting short messages (140 characters), making it ideal for constant quick comments. With over 300 million active users each month, around 500 million tweets a day are published worldwide. Due to these features and the vast amount of information, through Twitter we can find information about anything that happens in the world and concerns aspects of human activity.
The Twitter reflects society's opinion. Each user's tweet is a sign of what each person thinks and feels. Also with the Twitter we can see the current events that happen around us and how the world reacts through it. A common subject of discussion is politics.
This thesis, will analyze data from Twitter regarding the Greek elections and the Greek referendum and test how an election and referendum outcome could be predicted from this tweets [book 2, ref2]. In the next paragraph we will make a summary of what will follow in this thesis, as well as what we will be the subject of this.

1.2 Overview

In this thesis we examine the role of twitter from the politicians as well as the citizen's perspective. Specifically, we study its influence and contribution in the political process. The main objective of this thesis is to analyze the data relating to the elections and the referendum that took place in Greece during 2015.

1.3 Contents of the research

The work in this thesis includes, in addition to the introductory chapter where the study question is described, the following 4 chapters:
In chapter 2, "Data", describes the data collected from Twitter about Greece's elections and the Greece's referendum. This chapter also describes how a Python script is used to highlight the data and give a summary of their characteristics. The Greek Election dataset consists of the collection of tweets on the election. Greece's referendum dataset consists of a collection of tweets on the referendum.
In chapter 3, "Methodology", which was followed for selecting and studying articles, analyzing and selecting an appropriate collection and analysis tool for tweets and analyzing data relating to the elections and the referendum of Greece in 2015.
In chapter 4, "Experimental Results", we perform predictions for the test cases and we evaluate our predictions by comparing them to the actual results.
In the final chapter, "Discussion", are presented the conclusions that were exported and are made some suggestions for future research.

Chapter 2

Data Preparation

2.1 Data Collection

The data which collected for this thesis are related to the Greek Election and the Greek referendum. For the Greek elections we collected the tweets through a python script. Due to technical limitations, we forced to use the python script in order to achieve our goal for the test case of Greece's elections. We collected tweets that were posted from Greece during a year before the election's. Also, we gathered tweets from the Greece during one year before the Greece's referendum.
We now describe the political context used in this research. We used only the tweets of political leaders, roughly 11000 entries. We selected 8 political leaders as the most likely to join the House.The election took place on January 25th 2015, and the dataset covers about 1 year before the elections. We take the tweets that only refers to Greece and are written in Greek.

2.1.1 Political Context

Of the eight political leaders, the two most preeminent to get the prime minister's vote is Alexis Tsipras from Syriza party and Antonis Samaras from the New Democracy party. The first 2 parties received 36.34% for Syriza and 27.81% for New Democracy, respectively. However, even though the two candidates had a presence on television and interviews in Press, only Tsipras had a strong presence on the Twitter, while the tweets from the other candidate was very limited. This has the result that the data we have to study have even more interest in their interpretation. Before we continue we now give you an overview of the data we collected:

Data	Collection Period	Dates	File size	Dataset size
Greek elections	1 year	25/1/2014-25/1/2015	1,2Mb	tweets: 5018
Greek elections2	1 year	25/1/2014-25/1/2015	0,6Mb	tweets: 2641
Greek referendum	1 year	25/1/2014-25/1/2015	0,8Mb	tweets: 3268

Table 2.1: Summary of datasets

2.1.2 Greek elections

In the first parliamentary elections, on 25 January 2015, seven parties were elected to the House. We will deal with the first four which have taken the most votes to see some characteristics of them.
Syriza is a left party and its ideology comes from democratic socialism, social democrats, ecosocialism, Eurocommunism, anti-capitalism. It was founded in 2004 as an electoral alliance and since 2012 it has been a united party. The party generally did not get good votes in the parliamentary elections, but in 2012 it made a great reversal and from the last positions it took as a parliamentary party, it suddenly eased its rates to 16.78%. This is due to the fact that until then one of the two major parties, namely Pasok, when was a government, it forced to implement tough financial measures, due to a memorandum that had signed and it was dysfunctional for the people. So there are no left a margin for the people rather than vote for a new party to the power to negotiate better the lenders. Finally, in 2015 it managed to form a government with Alexis Tsipras as prime minister and a rate of 36.34%.
New Democracy is a center-right party and its ideology is defined by social liberalism and liberal conservatism. Since New Democrasy's establishment in 1974, it was always first or second to vote, with the result that it was either a government or a primary opposition. In 2015, however did not manage to take the first position and thus went second with a percentage of 27.81%.
Golden Dawn is a far-right party which ideologically lies to nazism/neonazism, anti-communism, ultra-nationalism, Euroscepticism and anti-globalization. It was established in 1980 and its activity began in 1993. While the party's percentages were almost zero by 2009, about 0.5%, in the national elections in 2012 experienced a rapid rise and has emerged sixth party in the House with 6.97%. The strengthening of Golden Dawn is due to the globalization of the economy, which creates economic problems, by strengthening nationalism and the collapse of the Greek political system [wiki 3, ref1].
Potami is a center, center-left party supporting neoliberalism, liberalism, social democrats. It was founded in 2014 and the first elections, that took place, was

in european elections, by doing to choose two Members of the European Parliament, with a rate of 6.6%. Finally, in the January elections, Potami, was elected a fourth party with 6.05% and elected 17 Members of Parliament.
In the second parliamentary elections that took place in the same year, and in particular on 20 September 2015, we had some differences in the ranking of the parties in the House. Now the fourth party in the House is Democratic Compensation and the House consists of 8 parties. The fourth position received by DC represents a center-left party which ideologically lies to social democrats. In the September 2015 elections, the alliance received 6.28% of the vote, thus electing 17 MPs in the Greek Parliament [wiki 4, ref2]. The new party that was baptized in the House is the union of centers. It was established in 1992, and supporting, liberalism, venizelism. In the elections of 20 September 2015, nine members were elected with the Union of Centers [wiki 5, ref3].

2.1.3 Greek referendum

The Greek referendum was carried out on July 5 with the question of whether to accept the draft agreement of three institutions, the European Commission, the European Central Bank (ECB) and the International Monetary Fund (IMF) which was proposed on 25 June. The result of the referendum was to reject the proposal of the draft agreement with 61.3% [wiki 6, ref4]. The subject of the disagreement was the kind of financial measures Greece had to take to complete the previous program of economic rescue against the Greek debt and a new rescue package. Finally, the government reached an agreement with the lenders under terms of a third memorandum. This has the result the partition of Syriza, and in the next elections of 2015, it pointed out winner, which formed a government with the Independent Greeks.

2.2 Data Preprocessing

The data are saved as csv files. Before we tell you what exactly contains its line, we will refer you that the elections1.csv, which is the results of the first election, contains the tweets of all the political leaders we are studying in our work. Each line of the csv files represents a tweet, which consists of 10 main attributes, such as username, date, retweets, favorites and text. We will use only those mentioned above, and may need it to use **hashtag** attribute in some points. This research focuses on the sentiment analysis toward the candidates in a particular time. The collected tweets are written in Greek. The preprocessed tweets are available upon request for future use.
We will do a data preprocessing before applying our analysis with the help of R,removing URL and pictures from the dataset. The rest of the attributes which we do not use are not removed in order to not change the original interpretation of a tweet. In closing, we point out that the variables we used to carry out our research were the **retweets** and the **favorites**, which are two quantitative variables. All the rest were applied to various chapters we analyzed, but mainly to the last two.

Chapter 3

Methodology

In this chapter we will analyze the methods which we have used in our research. The methods that one can analyze are many. Some of them are Data Visualization, Regression, Clustering, Time Series Analysis, Text Mining and Social Network Analysis which we will elaborate in the paragraphs that follows.

3.1 Data Exploration

In this section we will present some ways to look at data using some very well known statistical summaries and graphs. At first we will start with the structure of data. The function str() provides a method to display the structure of R data structures such as data frames, vectors, or lists. It helps us to understand how many variables we have and what kind of variables are these, if it is quantitative or qualitative. Also with dim() and names() function we can see the dimensionality of the data and the names of the variables respectively.

3.1.1 Exploring individual variables

In the beginning we will explore the numeric variables. To investigate these, we will make use of a usual set of measurements to describe values known as summary statistics. The summary() function displays several common summary statistics. The command **summary** can be divided into two types: measures of center and measures of spread. Measures of central tendency utilized to identify a value that falls in the middle of a dataset. One of the most usual measures of that type is the average. The average in statistics is the mean, which specified as the sum of all values divided by the number of values. We can calculate the mean for a vector in numbers in R by mean() function. Another measure of central tendency is the median, which is the middle number in a sorted list of numbers. In R the median() function finds the middle value.
In the other hand a measure of spread, is used to describe the variability in a sample or population. These measures is important because it gives us an idea for the representation of the data by using the mean. For example, if the values in the sample is large, the mean is not so representative as if the spread of sample is small. From the summary() function we get five statistics, which are all measures of spread. Firstly, we receive the minimum and the maximum, which are the smallest and largest values, respectively. R provides the min() and max() functions to calculate these values on a vector of data. Quartiles tell us about the spread of a data set by breaking the data set into quarters. The first quartile (Q1) lies between the 25th and 26th value, the second (Q2) is the median and the third (Q3) composed from the 75% of the observations.In R with function quantile() we can achieve to take the quartiles. One more measure of spread can be the very well known to all of us the **variance**. But what actually variance is? The definition of the variance is very simple, and actually is the amount of variation that occurs around the mean score. The variance can be found from this formula:

$$variance = \frac{\sum(X - \mu)^2}{N}$$

where μ = mean, X = score, $\sum$= the sum of, N= number of scores, $\sum X$= "add up all the scores"
We can calculate the variance in R by var() function.

3.1.2 Exploring categorical variables

Except from the numeric variables, there are also the categorical variables. Categorical variables are those variables that fall into a specific category. For instance, hair color, gender, or college major are all of them categorical variables.
For example:

- The category "hair color" could conclude the categorical variables "black," "brown," "brunette," and "white."
- The category "gender" could conclude the categorical variables "Male" and "Female."

The main difference of the category variables in relation to the quantitative ones is that the first ones are descriptive, that is, they describe an object while the second counts only one with its numerical value. The categorical data is investigated using **tables**. The table() function which used in R, lists the categories of the variable and count the values falling into this category. One more measure which i could use is a measure of central tendency and it is the mode. The mode basically indicate how often the value occurring. In this thesis we will not deal with this, but we mention it as a measure for a categorical variable.

3.2 Data Visualization

In this section we will describe with the help of graphs, the numeric and categorical variables. First, for the numeric variables we have the histogram, boxplot and so many others such as smoothscatter and qplot. We will involved mainly in our analysis with histogram and boxplot, rather than the other two. With the histogram we can see the width of a variable through a graph. The boxplot on the other, split the values of the variables into an ordered number of parts that act as boxes for values. We can create the histogram in R by using the hist() function, and the boxplot by using the boxplot() function. One more thing we forgot to say is that the histogram uses any number of portions of the same width, but allows the portions to contain different number of values. Also the series of the bars indicates the frequency of the values. In the boxplot, also known as a box and whiskers plot displayed the center and range of a numeric variable, which allows you to have a good optical of your data.
Now we will see some things about the visualization of the categorical data. In this data we have two graphs mainly, the barplot and the piechart. Bar graphs used to compare the sizes of categories. The values of that variable are qualitative indices, where the distribution of the variable shows the measurement or the percentage that each indicator has separately. Such a variable could be the political party that everyone supports, like "***SYRIZA***","***NEW DEMOCRACY***","***PASOK***" and many more. As opposed to barplot, piechart require all categories to be included in a graph. One difference between them is that the bargraph are more flexible than the piechart because in the first one you can compare selected categories, while in the other should either compare all categories or none. In R the functions are barplot() and pie() respectively.

3.2.1 Further exploration and visualization

In this subsection we will mention some other explorations and visualizations that exist. To begin with exploration we will refer to the relationship between two variables. Correlation test is used to estimate the association between two or more variables. In R we have cor() and cor.test() to calculate the correlation. We will talk about correlation more in detail in the next section. Visually we can see that with the plot() function, which in substance shows the relation between two variables. Also we have the aggregate() function ,which can, as the name suggests to aggregate the data. At last we have one more plot to discuss the qplot. The qplot has some differences from the normal plot. For instance, is not generic, in other words you cannot qualify any type of R object to qplot and anticipate to get some usual plot. Also it has some differences in the options, where allows to qplot to be far better visually from the default plot.

3.3 Regression

Regression is deal with definition the relationship between a single numeric dependent variable, which is the value to be predicted and one or more independent variables, the predictors. The predictors can be numeric variables mainly, but also it can be a categorical variable with the help of the dummy variable. Dummy variables are used to represent subgroups of the sample in our analysis. It is useful because they allow us to utilize a single regression equation to depict multiple groups. In this section we will intent to two major types of regression:the **linear** regression and the **logistic** regression.

3.3.1 Linear Regression

The simplest forms of regression supposed to be the connection between the independent and dependent variables follows a straight line [book 7, ref3]. The basic form for the linear equation is called the slope-intercept form and it has the look of $y = \alpha + \beta x$. The y indicates the dependent variable and the x the independent. The slope-term β defines how much the line changes for each increase in x. Positive values specify lines that slope uphill while negative values specify lines that slope downhill. The term α is known as the intercept because it determines the point where the line intercepts the y axis. It shows the value of y when $x = 0$. Essentially we try to find the best line that fit the data more well, with the help of statistical program R.
The best way to decide the optimal estimates for α and β is the **Ordinary Least Squares** (OLS) method. The OLS method attempts to minimize the sum of the squared errors(SSE) using the appropriate α and β. The SSE is the vertical distance between the predicted y value and actual y value. These errors are also noted as residuals.An example of this method could be represented in the following diagram 3.1:

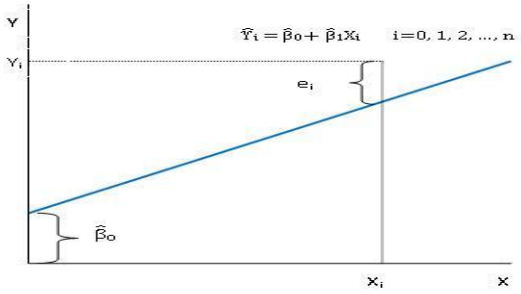


Figure 3.1: A linear regression

In mathematical terms, the aim of OLS can be reflected as the result of minimizing the following equation:

$$\sum (y_i - \hat{y_i})^2 = \sum e_i^2$$

This equation explains that the error is the difference between the actual and predicted value. To find though the total error in the regression we should square and sum all the error values across the data. We can be found α which depends of β solving the following formula:

$$\alpha = \bar{y} - \beta \bar{x}$$

The value of β resulting from calculus and can be shown with the help of the next type:

$$\beta = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

Very easily we can see that this equation is made up of two known parts:the covariance in the numerator multiplied by n and the variance in the denominator multiplied by n.
One more thing we would like to discuss before show the multiple linear regression is the correlation. The correlation between two variables is a value that points out how tightly their relationship follows a straight line. The correlation ranges between -1 and +1.The extreme values declare a perfectly linear relationship, while a correlation close to zero symbolizes the absence of a linear relationship. The following formula designate Pearson's correlation:

$$\rho_{x,y} = Corr(x, y) = \frac{Cov(x, y)}{\sigma_x \sigma_y}$$

where σ_x is the standard deviation of x and σ_y is the standard deviation of y.

3.3.2 Multiple Linear Regression

Now we will talk about the multiple linear regression, videlicet for the analyses that used mainly in the real world problems and generally from public. The strengths and weaknesses of multiple linear regression are showing next:

Strengths

1. It can be customized to almost any modeling tasks
2. It provides evaluation of both the strength and size of the relationships between attributes and the result

Weaknesses

1. Missing data need special attention, often using multiple imputation techniques
2. Naturally handles numeric attributes, so categorical data require extra processing

The aim of the multiple regression is exactly similar like linear regression's, and the major difference is that there are additional terms for additional independent variables. Multiple regression equations in general ensue the form of the following equation. The dependent variable y is decided as the sum of an intercept term α plus the product of the estimated value β and the x values for each of the i outcomes. An error has been added here as a prompt that this forecasts are not perfect. This displays the rest of the condition noted earlier:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_i x_i + \varepsilon$$

Let's check the interpretation of the estimated regression parameters. A coefficient is provided for each outcome,that means for each outcome allowed to have a separate estimated effect on the value of y. Videlicet, y changes by the amount β_i for each unit increase in x_i. The intercept α is then the expected value of y when the independent variables are all zero. For convenience the above equation is written:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_i x_i + \varepsilon$$

where β_0 is equal to α, from previous equation. Since the intercept is irrelevant to any of the other x variables, it assists to conceive β_0 s if it were being multiplied by a term x_0, which is a constant with the value 1:

$$y = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_i x_i + \varepsilon$$

The estimation of the values of the regression parameters, for each observed value of the variable y must be related to the observed values of the x variables utilizing the regression equation in the previous form. The following picture amplifies this structure 3.2:

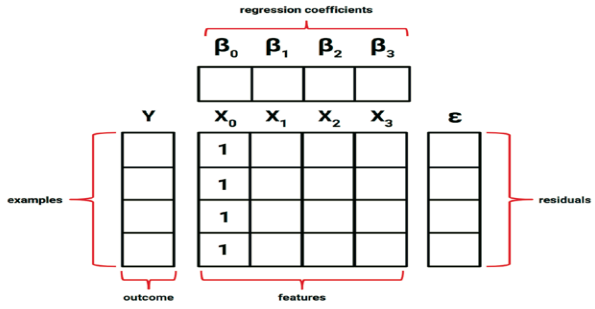


Figure 3.2: A multiple linear regression

This figure can be denoted in a form using bold font **matrix notation** to suggest that each of the terms presents multiple values:

$$\mathbf{Y} = \boldsymbol{\beta}\mathbf{X} + \boldsymbol{\varepsilon}$$

where $\mathbf{Y}$ is a vector, with a row for every example, the independent variables $\mathbf{X}$ are now a matrix, with a column for each feature plus a column of "1" values for the intercept and the regression coefficients $\boldsymbol{\beta}$ and errors $\boldsymbol{\varepsilon}$ are also vectors.
The aim is to solve for $\boldsymbol{\beta}$, the vector of regression coefficients that minimizes the sum of the squared errors between the predicted and actual $\mathbf{Y}$ values. From the bibliography we know that the best estimate of the vector $\boldsymbol{\beta}$ can be calculated as:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{Y}$$

where the $\boldsymbol{T}$ indicates the **transpose** of matrix $\mathbf{X}$, while the negative exponent indicates the **matrix inverse**. In R we will deal with the lm() function to encounter the multiple regression, and not with the function solve() and t(), which finds the inverse and transpose matrix respectively.

3.3.3 Logistic Regression

In this subsection we will discuss about logistic regression. The logistic regression can be applied as a special case of generalised linear models. **Logistic regression** analyzes a dataset in which there are one or more independent variables that decide an outcome. The outcome is a dichotomous variable, which have only two possible outcomes. The goal of the logistic regression is the same as of linear regression, to wit to find the best fit between the binary variable(dependent) and the explanatory variables(independent). Logistic regression is estimated by the following formula:

$$logit(p) = b_0 + b_1X_1 + b_2X_2 + \cdots + b_kX_k$$

where p is the probability of presence of the feature of attention. Now we must determine and explicate the logistic transformation. This transformation is often implemented to variables which are specified only on the open interval (0,1), such as percentages, to variables on (-∞,∞). The transformed variable is the natural logarithm of the odds responding to the quota. This is called the log-odds:

$$logit(p) = \ell = log\frac{p}{1-p}$$

where $\frac{p}{1-p} = odds = \frac{\text{probability of appearance of feature}}{\text{probability of disappearance of feature}}$

The estimation in logistic regression chooses parameters that maximize the likelihood of the observing values, known as MLE.

The logistic transformation is imprecise for proportions 0 and 1. It makes sense because a certain or impossible event cannot analyzed. We care about uncertain events. Also it is pointless to have infinite odds for or against an event, so the range of the logistic function does not include $\pm\infty$. The inverse transform, from log odds to proportion, is:

$$p = \frac{1}{1 + e^{-\ell}}$$

By multiplying numerator and denominator by e^{ℓ}:

$$p = \frac{e^{\ell}}{1 + e^{\ell}}$$

The domain for this function is (-∞,∞) and the range is (0,1).

3.3.4 Multinomial Logistic Regression

Except for the independent variables that may be more than 2, so the response variable can have more than two results. These response variables called polytomous("many outcome"). In our example we have eight possibilities. The concept behind this model is that there is one category which used as the base, and the probabilities of the others at the same time esteemed with respect to the base. The base will be the name of the variable **username**, which called **"atsipras"**.
The mathematical form of the multinomial logistic model is more complex than the binomial model. We want to model the probability π_{ij} that observation i is in each jth rank of the m response ranks $j = 1, ..., m$. The first response rank $j = 1$ is taken as the base rank. So the base probability π_{i1} is calculated as the residual probability after the other ranks $\pi_{i2}, ..., \pi_{im}$ have been modelled [article 8, ref1].
Thus the model has k + 1 coefficients for each of the $j = m - 1$ ranks (leaving out the base class). There are one intercept α_j and one "slope" for each predictor (continuous or each class of a classified predictor) β_{ij}, where $l = 1...k$ is a column in the model matrix [article 8, ref1].
The proper probabilities are then:

$$\pi_{ij} = \frac{e^{(\alpha_j + \beta_{1j}x_{i1} + ... \beta_{kj}xik)}}{1 + \sum_{l=2}^{m} e^{(\alpha_j + \beta_{1l}x_{i1} + ... \beta_{kl}x_{ik})}} \tag{3.1}$$

$$\pi_{i1} = 1 - \sum_{j=2}^{m} \pi_{ij} \tag{3.2}$$

This set of equations is fitted by maximum likelihood.
The fitted α and β can be treated to estimate the **log-odds** of an observation being sorted in each rank, **relative to the base class**. At this point, we must note that these are **not** the odds of being in that class, only the odds relative to the base class. The log-odds are calculated as:

$$ln\frac{\pi_{ij}}{\pi_{i1}} = \alpha_j + \beta_{1j}x_{i1} + ... + \beta_{kj}x_{ik}, j = 2, ..., m \tag{3.3}$$

Notice that if $m = 2$ this decreases to the binomial logit model.

So, once we fit the model, we can predict the odds of some rank, relative to the base. We can also predict the odds between any two ranks, by combining the above equation for the two ranks j and l [article 8, ref]:

$$ln\frac{\pi_{ij}}{\pi_{il}} = ln\Big(\frac{\pi_{ij}/\pi_{i1}}{\pi_{il}/\pi_{i1}}\Big) \tag{3.4}$$

$$= ln\frac{\pi_{ij}}{\pi_{i1}} - ln\frac{\pi_{il}}{\pi_{i1}} \tag{3.5}$$

$$= (\alpha_j - \alpha_l) + (\beta_{1j} - \beta_{1l})x_{i1} + ... + (\beta_{kj} - \beta_{kl})x_{ik} \tag{3.6}$$

In R we will use the multinom() function to deal with the multinomial logistic regression.

3.4 Clustering

Clustering is the technique that the population or data points can be divided into a number of groups such that data points in the same groups are more alike to other data points in the same group than those in other groups. In other words, the goal is to separate groups with much like characteristics and appoint them into clusters. In this section we will consider four of the clustering techniques. They will be the k-means clustering, k-medoids clustering, hierarchical clustering and density-based clustering.

3.4.1 K-Means Clustering

The K-means clustering method is a plain way for estimating the mean (vectors) of a set of K-groups. The plainness of the algorithm also can induce some bad solutions, for example the partial optimization of clusters of the objects. A classic approach of the K-means algorithm is the following:

1. Partial cluster points are chosen at random.
    - These display the "temporary" means of the clusters
2. The squared Euclidean distance from every object to every cluster is reckoned, and each object is assigned to the closest cluster.
3. For every cluster, the new centroid is calculated, and each point value is substituted by the relevant cluster centroid.
4. The squared Euclidean distance from an object to any cluster is calculated, and the object is assigned to the cluster with the smallest squared Euclidean distance.
5. The cluster centroids are recomputed based on the new membership assignment.
6. Steps 4 and 5 are repeated until no object moves clusters.

But for all these to apply, it is first necessary to apply some assumptions.These are the following:

1. K-means uses the squared Euclidean distance to issue objects to clusters.
    - There is a supposition that the data should have approximately the same scale to use such distances.
2. Because the squared Euclidean distance is utilized, the K-means algorithm prospects to attempt to find a minimum for the Error Sum of Squares (SSE).
    - The feeble assumption is each group has crudely the same SSE – meaning that the variance/covariance matrix between objects within a group is equal.

Before we finish with the K-means we will converse about the SSE.The SSE for a total set of object is computed by:

$$SSE = \sum_{j=1}^{p} \sum_{t=1}^{K} \sum_{i \in C_t} (x_{ij} - \bar{x}_j^{(t)})^2$$

The aim of K-means is to find out a solution such that there are no other solutions with lower SSE. K-means is a good method to swiftly classify your data into clusters. All you need to know are the number of clusters you request to find. In R the function that makes k-means clustering is kmeans().

3.4.2 K-Medoids Clustering

The K-medoids clustering method is affiliated to the K-means method because both of them algorithms are partitional, to wit dividing the dataset up into groups and both try to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster. In K-means, they conclude means as the centroids but in the K-medoids, data points are concluded to be the medoids. A medoid can be specified as that object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal. The basic idea of this algorithm is to firstly calculate the K vicarious objects which are called as medoids. After finding the set of medoids, every object of the data set is delegated to the nearest medoid. That means, object i is set into cluster v_i, when medoid mv_i is nearer than any other medoid m_w.
The algorithm succeeds in two steps:

- CREATE-step: This step consecutively chooses k "centrally located" objects, to be utilized as primary medoids.
- EXCHANGE-step: If the objective function can be restricted by trading(swapping) a concluded object with an excluded object, then the swap is carried out. This keep going till the objective function can no longer be lessened.

The following steps are the analysis of the two previous steps:

1. Firstly choose k random points as the medoids from the given n data points of the data set
2. Connect any data point to the closest medoid by using any of the most common distance metrics.
3. For any pair of non-selected object h and selected object i, compute the total exchanging cost TC_{ih}.
    - If $TC_{ih} < 0$, i is substituted by h
4. Repeat the steps 2-3 until there is no variation of the medoids.

But before we continue with the algorithm, we should consider four states:

(i) Shift-out membership: an object p_i maybe required to be displaced from presently assumed cluster of o_j to another cluster.

(ii) Update the present medoid: a new medoid o_c is found to relieve the current medoid o_j.

(iii) No change: objects in the present cluster outcome have the same or even smaller square error criterion(SEC) measure for all the possible redistributions considered.

(iv) Shift-in membership: an outside object pi is specified to the present cluster with the new(relieved) medoid o_c.

PAM works usefully for small data sets but does not scale well for large data sets. The graphic representation of the K-means and K-medoids clustering is done with the help of the clusplot and silhouette plot. In R the function that makes k-medoids clustering is pam().

3.4.3 Hierarchical Clustering

Hierarchical clustering includes clusters that have a prearranged ordering from top to bottom. There are two types of hierarchical clustering, Divisive and Agglomerative. In the Divisive method we define all of the observations to a single cluster and then segregate the cluster to at least two similar clusters. Lastly, we move on retrospectively on every cluster until there is one cluster for every observation. In the Agglomerative method we determine each observation to its own cluster. Then, calculate the similarity(distance) between each of the clusters and join the two most similar clusters. The related algorithm is shown below.

1. Place any data point into its own singleton group.
2. Try repetitively to join the two closest groups.
3. Repeat the step two until all the data are combined into a single cluster.

Prior to each grouping, it is required to specify the proximity table that contains the distance between each point using a distance function. The table is then updated to show the distance between each cluster. The following three methods differ in how the distance between each cluster is measured. In the single linkage, the distance between two clusters is designated as the shortest distance between two points in each cluster. In the complete linkage, the distance between two clusters is determined as the longest distance between two points in each cluster. In the average linkage, the distance between two clusters is assigned as the average distance between each point in one cluster to every point in the other cluster. Before we close this part we will see the differences between hierarchical and K-means clustering. In hierarchical clustering only needs a measure of similarity between groups of data points, in contrast with the K-means that needs a number of clusters K and a primal adjustment of data to clusters. The graphic representation of the hierarchical clustering is done with the help of the dendrogram. In R the function that makes hierarchical clustering is hclust().

3.4.4 Density-based Clustering

Density based clustering algorithm has played a crucial part to discovery non linear shapes structure based on the density. The most common density based algorithm is the Density-Based Spatial Clustering of Applications with Noise (DBSCAN). It uses the meaning of **density reachability** and **density connectivity**. A spot "p" is said to be density reachable from a spot "q" if spot "p" is within ε distance from spot "q" and "q" has enough number of spots in its neighbors which are within distance ε [google 9, ref1]. This is called density reachability. A sign "p" and "q" are said to be density coherent if there exist a sign "r" which has adequate number of signs in its neighbors and both the signs "p" and "q" are within the ε distance. This is chaining process. So, if "q" is neighbor of "r", "r" is neighbor of "s", "s" is neighbor of "t" which successively is neighbor of "p" implies that "q" is neighbor of "p". This is called density connectivity.
The steps of the **DBSCAN clustering** are the following:
Let $\boldsymbol{X} = \{x_1, x_2, x_3, ..., x_n\}$ be the set of data points. DBSCAN needs two parameters: ε (eps) and the minimum number of points that needed to form a cluster (minPts).

1. Start with a random initial point that has not been inspected.

2. Extract the neighborhood of this point using ε, because all points which are within the ε distance are neighborhood.

3. If there are ample neighborhood around this mark then clustering procedure starts and point is noted as visited else this point is labeled as noise, because later this point can become the part of the cluster.

4. If a spot is found to be a component of the cluster then its ε neighborhood is also the component of the cluster and the above process from step 2 is repeated for all ε neighborhood spots. This is repeated till all points in the cluster is defined.

5. A new unresolved point is recaptured and handled, guiding to the finding of a further cluster or noise.

6. This procedure continues until all signs are labeled as inspected.

The advantages of this algorithm are the following:

- We do not need to know the number of clusters a-priori.
- Can distinguish noise in data while clustering.
- Is able to discover randomly size and randomly shaped clusters.

The disadvantages of this algorithm are the following:

- It fails in case of variant density clusters.
- Fails in case of neck type of dataset.
- It does not perform well in case of high dimensional data.

In R the function that makes destiny based clustering is dbscan(). Before closing, we would like to mention to the outlier detection by clustering. To do that we compute the distances between objects and cluster centers, we pick the top 5 largest distances, we print the outliers and then we plotted by a simple scatterplot.

3.5 Time Series

Analyzing the data observed at different time points leads to peculiar problems. The apparent dependence imported by the sampling data over time reduces the feasibility of many usual statistical methods that demand random samples. The analysis of such data is usually mentioned as time series analysis.
Decomposition methods are established on an analysis of single elements of a time series. The power of each component is counted singly and then replaced in a model that explains the behavior of the time series. Two of the most important methods of decomposition are **multiplicative decomposition** and **additive decomposition**. The multiplicative decomposition model is enunciated as the product of the four components of a time series:

$$Y_t = TR_t S_t C_t I_t$$

where y_t is the value of the time series at time t, TR_t is the Trend at time t, S_t is the Seasonal component at time t, C_t is the Cyclical component at time t and I_t is the Irregular component at time t. Each item has a t mark to denote a specific time period. The time period can be counted in weeks, months, quarters, years and so on. The equivalent model for the additive decomposition is expressed as the sum of the four components of a time series:

$$Y_t = TR_t + S_t + C_t + I_t$$

The additive decomposition method is more suitable when the seasonal factors tend to be stationary from one year to the next. In contrast, multiplicative decomposition is more widely used, because it has seasonal factor that increases accordingly with the level of the time series.
Autoregressive Moving Average (ARMA) model composes of two parts, an autoregressive (AR) part and a moving average (MA) part. The AR part includes regressing the variable on its own lagged with the help of the past values. The MA part implicate modeling the error term as a linear combination of error terms occurring simultaneously and at diverse times in the past. The model is usually related to as the ARMA(p,q) model where p is the order of the autoregressive part and q is the order of the moving average part. An equation for the ARMA model could be the following:

$$Y_t = a_0 + a_1 y_{t-1} + ... + a_p y_{t-p} + b_1 e_{t-1} + ... + b_q e_{t-q} + e_t$$

where Y_t is the actual value of the time series at time t, y_{t-1} equals the actual value of the time series at time t – 1, $y_t - y_{t-1}$ equals the net change in the value of time

series between time t and time t – 1, e_{t-1} equals the error term at time t – 1, a_1 equals the autoregressive parameter for y_{t-1}, b_1 equals the moving average parameter for e_{t-1} and $e_t, e_{t-1}, e_{t-2}, ...,$ and e_{t-q} are uncorrelated. Also the a_0 is set to be the intercept of the model. This was the ARMA model. For the ARIMA model we have the following equation:

$$Y_t = a_0 + a_1 y_{d_t-1} + ... + a_p y_{d_t-p} + b_1 e_{t-1} + ... + b_q e_{t-q} + e_t$$

where y_d is Y differenced d times. For example, $Y_{d_t} = Y_t - Y_{t-1}$. The d deputizes the rate of differencing in the integrated (I(d)) component. Differencing a series implicates simply subtracting its current and previous values d times. Often, differencing is utilized to steady the series when the stationarity hypothesis is not met. These were the time-series models and we will deal with the ARIMA model mostly in this thesis. Before we close the section we will see the equation of the MA model and the AR model, simply for educational reasons. The MA equation is the following:

$$Y_t = a_0 + b_1 e_{t-1} + ... + b_q e_{t-q} + e_t$$

The AR equation is the next one:

$$Y_t = a_0 + a_1 y_{t-1} + ... + a_p y_{t-p} + e_t$$

3.6 Text Mining

Natural languages (English, French, German etc.) are different from programming languages. The significance or the meaning of a situation rely on the context, tone and a lot of other coefficients. Unlike programming languages, natural languages are vague. Text mining attend to helping computers comprehend the "meaning" of the text. Some of the ordinary text mining applications contain sentiment analysis e.g if a Tweet about a series says something positive or negative, text classification e.g classifying the calls you get as known or unknown numbers. In this thesis, we will learn about text mining and with the help of R we will represent some common text mining techniques, such as sentiment analysis, wordcloud and clustering. Before we analyze the text, we need to preprocess it.
Text data includes white spaces, punctuations, stop words etc. These characters do not attribute much information and are hard to process. For example, English stop words like "the" and "is" do not give you much information about the sentiment of the text. This will help insulate text mining in R on significant words [google 10, ref2]. Before we continue with the wordcloud we must first clean the text data. To do that we will need some process in R. We should convert the text to lower case, so that words like "read" and "Read" are considered the same word for analysis, remove numbers, remove English stopwords, remove punctuation, remove extra white spaces and stemming our text. Stemming is the procedure of decreasing inflected words to their word stem, base or root form. A very useful library for performing the above steps and extracting text in R is the package "tm". The main structure for operating documents in tm is called a Corpus, which presents a compilation of text documents. First we will clean the text in R, then with the help of "tm" library we process the text and for stemming the text, we will use the "SnowballC" library. Also we want to represent the document term as a matrix with the assistance of term-document matrix(tdm). Tdm is a matrix which is row is a document vector, with one column for every term in the entire corpus. Also the **frequent term** is the value in each cell of the matrix. With the tdm we can move on to create a wordcloud, which is an explanatory way to understand text data and to do text analysis.
We attempt to attain clusters of words with hierarchical clustering. Sparse terms are detached, so that the plot of clustering will not be serried with words. Then the distances between terms are computing with dist() after scaling. After that, we cluster the terms with hclust() and make the dendrogram to cut into 3 clusters. The method we used in agglomeration is set to be the ward method, which means the raise in variance when two clusters are combined. The tweets are clustered with the k-means and the k-medoids algorithms. The k-means clustering gets the values in

the matrix as numeric. We transpose the term-document matrix to a document-term one. We clustered the tweets with kmeans() and the number of clusters are set to ten. Subsequently, we control the popular words in every cluster and also the cluster centers. A fixed random seed is set with set.seed() before doing kmeans(), so that the clustering outcome can be repeated. The k-medoids clustering with the Partitioning Around Medoids (PAM) algorithm, which uses medoids instead of means to introduce clusters. It is more robust to noise and outliers than k-means clustering, and affords a report of the silhouette plot to show the quality of clustering.
Sentiment analysis is the operation of choosing whether a piece of writing is positive, negative or neutral. In this work we will use the package "syuzhet". "Syuzhet" employs NRC Emotion lexicon. The NRC emotion lexicon is a list of words and their relations with eight emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive) [google 10, ref2]. The get-nrc-sentiment function get back a data frame in which each row presents a sentence from the original file. The columns contain one for every emotion type was well as the positive or negative sentiment vigor. It lets us to get a body of text and return which emotions it shows. Also we can see whether an emotion is positive or negative.

3.7 Social Network Analysis

A social network is a social formation of people, connected (directly or indirectly) to each other through a usual connection or interest. Social network analysis (SNA) is the study of social networks to realize their formation and behavior [article2 11, ref2]. We gave an interpretation of the social network analysis, but we also have to say about what is a graph, and about its features. A graph is a binary connection, which are the **edges** between elements of a set, which are the **vertices**. The graphs according to the pairs are divided into two types, the directed and the undirected. If the pairs are unordered, then the graph is undirected, otherwise it is directed.
We will use the *igraph* package from R to make the network of terms, and the data will be from **text mining** section. First we create a network of terms based on their concomitant in the same tweets, and then make a network of tweets based on the terms shared by them. At last, we create a two-mode network constitute of both terms and tweets. For the network of terms we will use a matrix, specifically a **termDocMatrix**, which is a normal matrix. After that, it is changed into a term-term adjacency matrix, based on which a graph is built. Then we plot the graph to indicate the relation between frequent terms.
We will follow the same proceed for the network of terms but our graph of tweets will be based on the number of terms we have in common. Finally we have the two-mode network which composed of two types of vertices: **tweets** and **terms**.

Chapter 4

Results

In this chapter we will represent the results of the methods which we have used in our research.

4.1 Explore and Visualize

In this paragraph we will see the summary of the data, which are the following:

Table 4.1: Summary1

Statistic	N	Mean	St. Dev.	Min	Max
retweets	5,018	18.783	60.678	0	2,090
favorites	5,018	10.440	29.114	0	1,290

Table 4.2: Summary2

Statistic	N	Mean	St. Dev.	Min	Max
retweets	3,268	15.925	56.523	0	1,626
favorites	3,268	28.177	99.823	0	2,109

Table 4.3: Summary3

Statistic	N	Mean	St. Dev.	Min	Max
retweets	2,641	14.996	34.814	0	843
favorites	2,641	26.306	55.430	0	1,254

From the previous tables we look some statistics from all the data we have gathered for this thesis, for the **numeric** variables **retweets** and **favorites**. We have the number of the observations for each dataset, the mean, the standard deviation, the minimum and the maximum value.
Now we will have a view of the quantiles. Quantile divides a probability distribution into zones of equal probabilities.

(a) Retweets1

0%	25%	50%	75%	100%
0	0	2	12	2,090

(b) Favorites1

0%	25%	50%	75%	100%
0	0	3	13	1,280

(a) Retweets2

0%	25%	50%	75%	100%
0	0	3	10	1,626

(b) Favorites2

0%	25%	50%	75%	100%
0	1	5	17	2,109

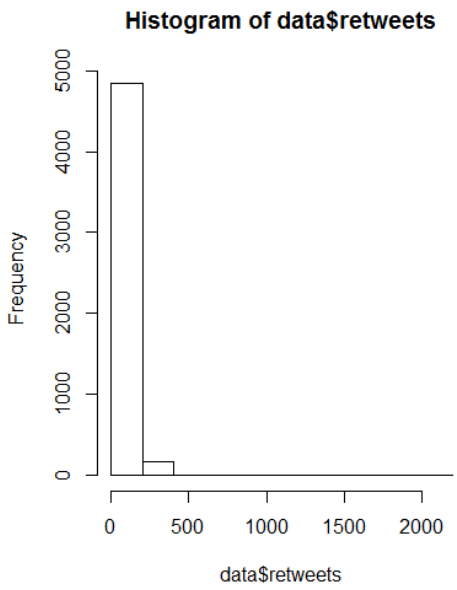
(a) Retweets3

0%	25%	50%	75%	100%
0	1	7	18	843

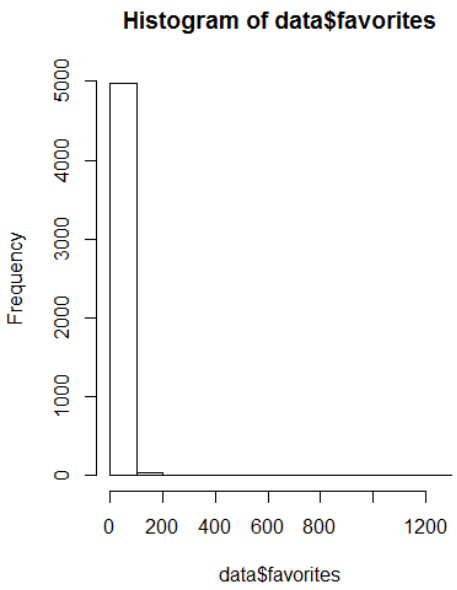
(b) Favorites3

0%	25%	50%	75%	100%
0	2	12	38	1,254

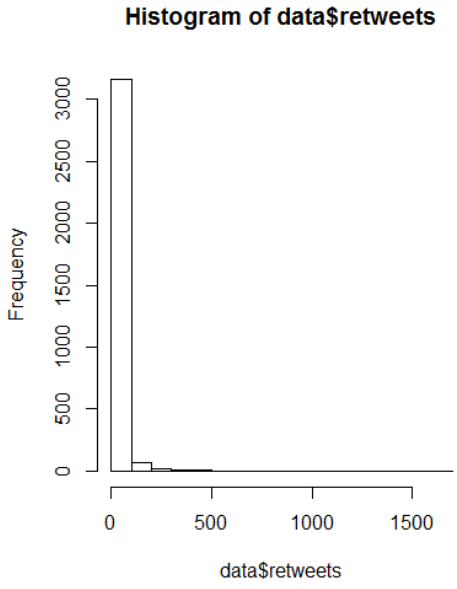
The retweets from the election dataset shows that the 50%, which is the median, that is the half of the retweets are below or equal to 2, while the other half, retweets more than 2. Three-fourth of the retweets are below or equal to 12, while the remaining one-fourth are above 12. The 25% are zero and the minimum value is zero. The maximum value is 2,090. The favorites for this dataset have some differences from the retweets. The 50% of the favorites are below or equal to 3, while the other half, favorites more than 3. Three-fourth of the favorites are below or equal to 13, while the remaining one-fourth are above 13. The 25% are zero and the minimum is zero. The maximum value is 1,280. The retweets from the referendum dataset shows that the 50%, which is the median, that is the half of the retweets are below or equal to 3, while the other half, retweets more than 3. Three-fourth of the retweets are below or equal to 10, while the remaining one-fourth are above 10. The 25% are zero and the minimum is zero. The maximum value is 1,626. The favorites for this dataset have differentiations from the retweets. The 50% of the favorites are below or equal to 5, while the other half, favorites more than 5. Three-fourth of the favorites are below or equal to 17, while the remaining one-fourth are above 1. The 25% are below or equal to 1, while the remaining three-fourth are above 17, and the minimum is zero. The maximum value is 2,109. The retweets from the election2 dataset shows that the 50%, which is the median, that is the half of the retweets are below or equal to 7, while the other half, retweets more than 7. Three-fourth of the retweets are below or equal to 18, while the remaining one-fourth are above 1. The 25% are below or equal to 1, while the remaining three-fourth are above 1, and the minimum is zero. The maximum value is 843. The favorites for this dataset have variations from the retweets. The 50% of the favorites are below or equal to 12, while the other half, favorites more than 12. Three-fourth of the favorites are below or equal to 38, while the remaining one-fourth are above 2. The 25% are below or equal to 2, while the remaining three-fourth are above 38, and the minimum is zero. The maximum value is 1,254. We notice that from the election dataset to the other two we have some increment in the 50% of the observations in both categories, while in retweets we have a fluctuation in the 75% between the three datasets, whilst the favorites only increased in the 75% of the observations. Next we give the histograms of the two variables for the three datasets.



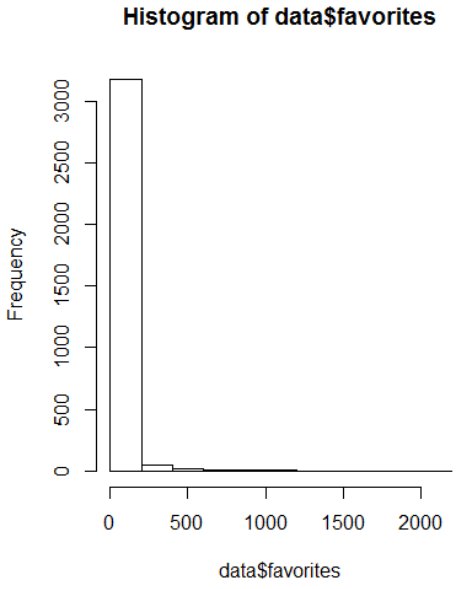
(a) histogram1



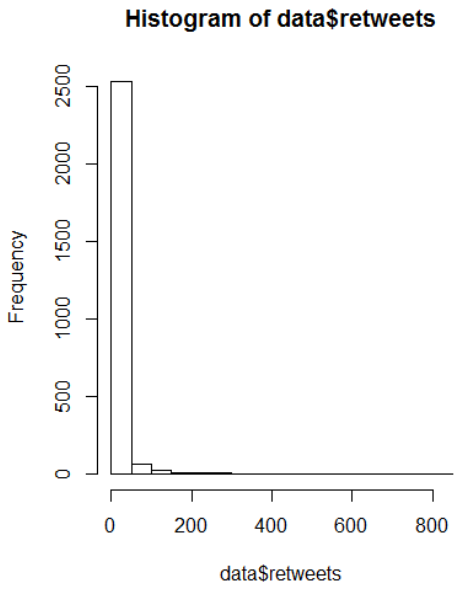
(b) hist2



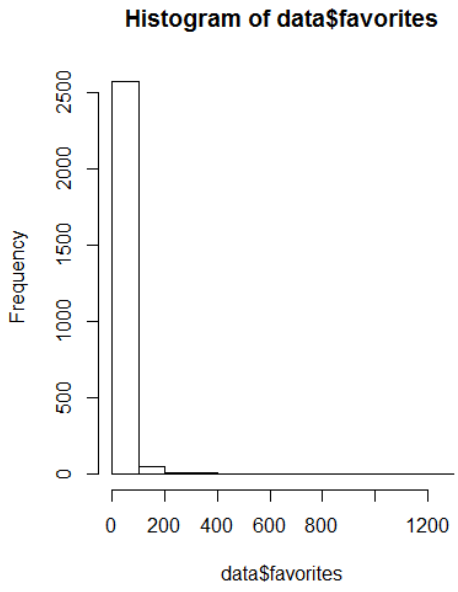
(a) hist3



(b) hist4



(a) hist5



(b) hist6

A histogram can show you the *shape* of the data, to wit the way that the data fall into groups. There are three shapes for the histogram: **symmetric, skewed-right, skewed-left**. A histogram is symmetric if you cut it down the middle and the left-side and right-side looks like exactly each other. A skewed right histogram have the look of a uneven mound, with a tail going off to the right. If a histogram is skewed left, it be like an uneven mound with a tail going off to the left. We notice that the retweets and the favorites from all the dataset are skewed-right. This means for the election dataset that all of the retweets were between 0-500 and favorites were between 0-200. For the referendum dataset we have the same skewness, but we see a few values from the retweets to have 200-500 and from favorites to have 400-1200. The election2 dataset has the same skewness, but we remark sundry values from the retweets to have 150-300 and from favorites to have 200-400.
We will then show only the charts of the **date** which is a date variable and **username** which is a factor variable, so we can see other more important things below in more detail.

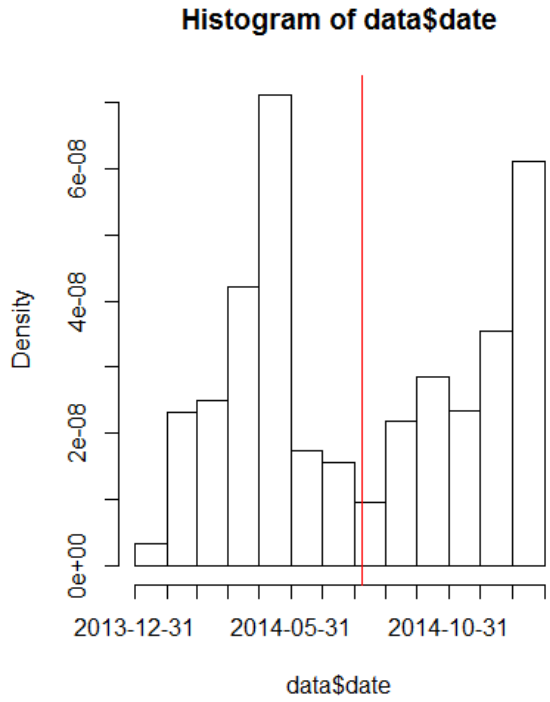


Figure 4.4: date1

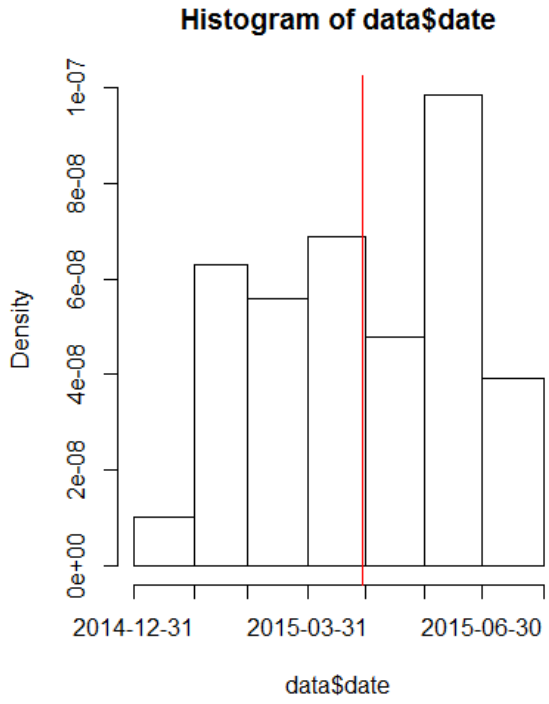


Figure 4.5: date2

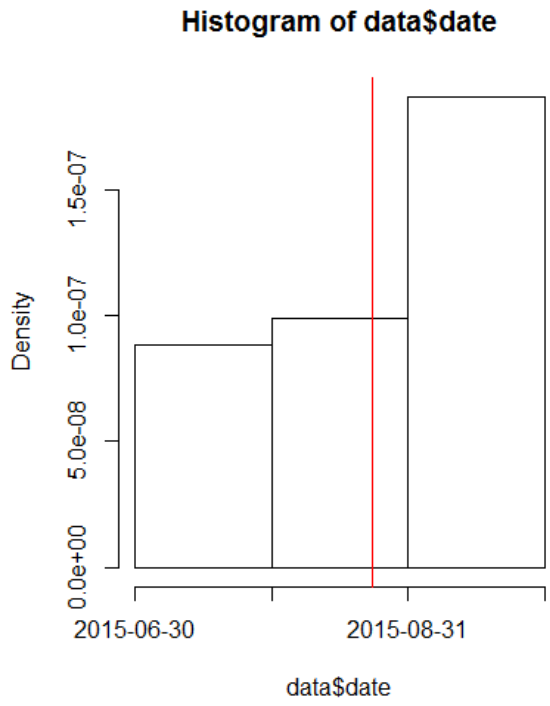


Figure 4.6: date3

We notice that in the first histogram, which includes observations of one year, we have an upward trend of the tweets up to 5 month, afterward down to 8 where its mean value is, and from 8 month and then it rises again. In the second histogram where the time period is 7 months, there is no upward or downward trend, as tweets seems to go up and down from month to month. In the third and last one which appears to contain 3 months, an upward trend is constantly observed.

For the factor variable **username** we have the following piecharts

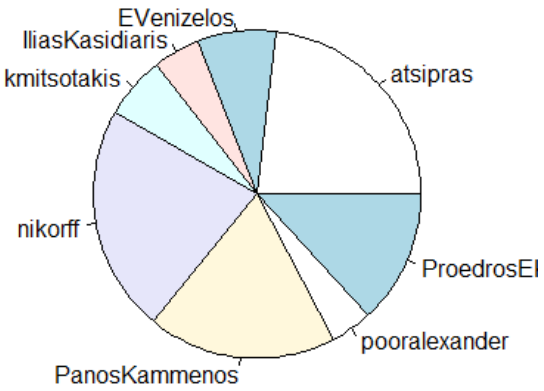


Figure 4.7: pie1

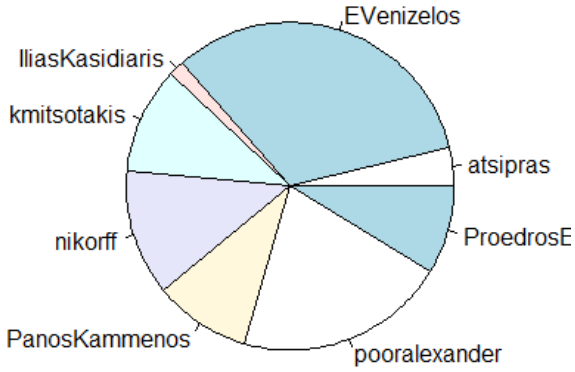


Figure 4.8: pie2

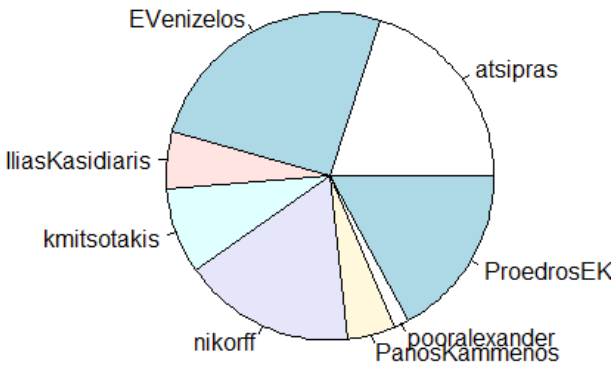


Figure 4.9: pie3

We notice that in the first piechart, which includes the dataset from the first elections, from the 8 levels of the username variable, the levels with the most observations are *"atsipras", "nikorff", "PanosKammenos"*. The second comprise the referendum data and gives for the username variable the highest values in *"EVenizelos", "pooralexander", "nikorff"* levels. In the third piechart we have the data from the second election and they will show us that the levels with the most observations are *"EVenizelos", "atsipras", "ProedrosEK"*. Because our variables are all factors, except from two numeric and one date, we will explore the relation of the two numeric variables only. We will show the relationship of the two variables, **retweets** and **favorites**, with the correlation coefficient, because not only it has a limited range from -1 to 1 which is more certain to make a decision, but also it does not have units, so as a result isn't affected by the different scales of the variables. For the first elections the correlation between the retweets and the favorites is ≈0.73. This means that we have positive relation between the two. Similarly, for the rest two datasets with correlation of ≈ 0.95 and ≈ 0.85 respectively. This indicates also a highly positive relation for the two variables. Also, we can compute the stats of **retweets** of every **username**

with aggregate(). We will show the results, because graphically with the help of the boxplot() function it doesn't look very good. Aggregate for the elections data:

	username	retweets.Min.	retweets.1st Qu.
1	atsipras	0.0000000	7.0000000
2	EVenizelos	0.0000000	0.0000000
3	IliasKasidiaris	2.0000000	5.0000000
4	kmitsotakis	0.0000000	2.0000000
5	nikorff	0.0000000	0.0000000
6	PanosKammenos	0.0000000	6.0000000
7	pooralexander	0.0000000	0.0000000
8	ProedrosEK	0.0000000	0.0000000

	retweets.Median	retweets.Mean	retweets.3rd Qu.
1	14.0000000	30.3852107	30.0000000
2	1.0000000	1.5684755	2.0000000
3	7.0000000	8.3090129	10.0000000
4	3.0000000	4.3483871	5.0000000
5	0.0000000	0.3425513	0.0000000
6	15.0000000	58.0586980	43.0000000
7	0.0000000	0.3365854	0.0000000
8	0.0000000	0.2583082	0.0000000

	retweets.Max.
1	2090.0000000
2	15.0000000
3	26.0000000
4	26.0000000
5	20.0000000
6	314.0000000
7	11.0000000
8	8.0000000

We can see that only the "atsipras" level has a value greater than 1000. This may be a sign of outliers. With the boxplot we will notice this evidence. Aggregate and boxplot for favorites will be left as an exercise for the reader. Aggregate for the referendum data:

	username	retweets.Min.	retweets.1st Qu.
1	atsipras	23.0000000	69.5000000
2	EVenizelos	0.0000000	3.0000000
3	IliasKasidiaris	7.0000000	11.0000000

4	kmitsotakis	0.0000000	6.0000000
5	nikorff	0.0000000	0.0000000
6	PanosKammenos	0.0000000	7.0000000
7	pooralexander	0.0000000	0.0000000
8	ProedrosEK	0.0000000	0.0000000

	retweets.Median	retweets.Mean	retweets.3rd Qu.
1	114.0000000	182.2913386	192.0000000
2	5.0000000	7.6035782	9.0000000
3	14.0000000	15.4150943	18.0000000
4	13.0000000	24.3797101	27.0000000
5	0.0000000	0.8960396	1.0000000
6	19.0000000	31.8419355	40.0000000
7	0.0000000	0.2961877	0.0000000
8	1.0000000	4.0456140	2.0000000

	retweets.Max.
1	1626.0000000
2	228.0000000
3	40.0000000
4	270.0000000
5	28.0000000
6	324.0000000
7	10.0000000
8	235.0000000

Aggregate for the elections2 data:

	username	retweets.Min.	retweets.1st Qu.
1	atsipras	8.0000000	18.0000000
2	EVenizelos	0.0000000	4.0000000
3	IliasKasidiaris	0.0000000	11.0000000
4	kmitsotakis	0.0000000	6.0000000
5	nikorff	0.0000000	0.0000000
6	PanosKammenos	0.0000000	5.0000000
7	pooralexander	0.0000000	0.0000000
8	ProedrosEK	0.0000000	0.0000000

	retweets.Median	retweets.Mean	retweets.3rd Qu.
1	23.0000000	31.5396226	33.0000000
2	7.0000000	9.7028232	11.0000000
3	15.0000000	15.6400000	18.0000000

```
4        13.0000000      19.4778761      23.0000000
5         0.0000000       0.7617978       1.0000000
6        21.0000000      64.2677165      89.0000000
7         0.0000000       0.6216216       1.0000000
8         1.0000000       2.3995585       3.0000000
   retweets.Max.
1    558.0000000
2     88.0000000
3     51.0000000
4    159.0000000
5     17.0000000
6    843.0000000
7      9.0000000
8     32.0000000
```

In the referendum data the same comments but in the elections2 data it seems to be out of bounds the 6 level which is the “PanosKammenos”.

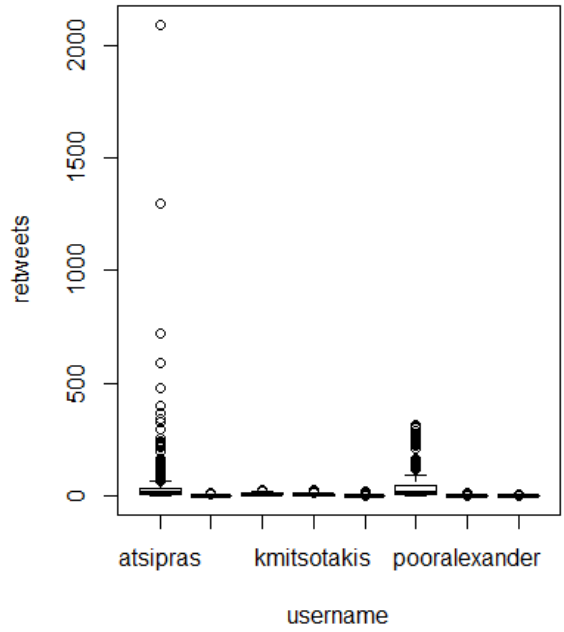


Figure 4.10: box1

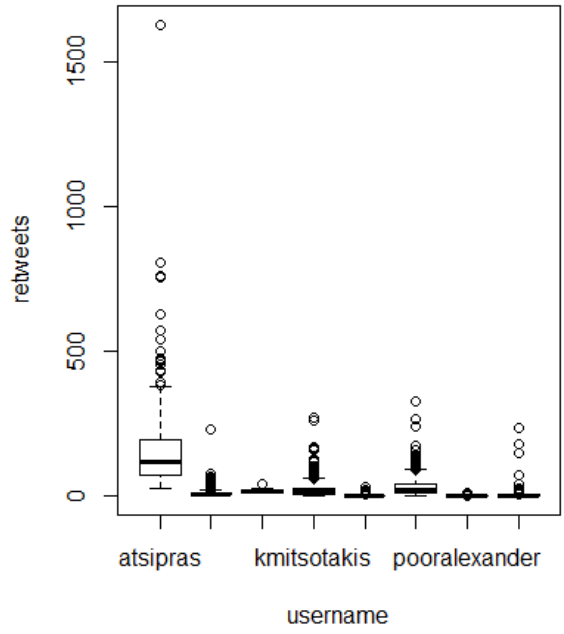


Figure 4.11: box2

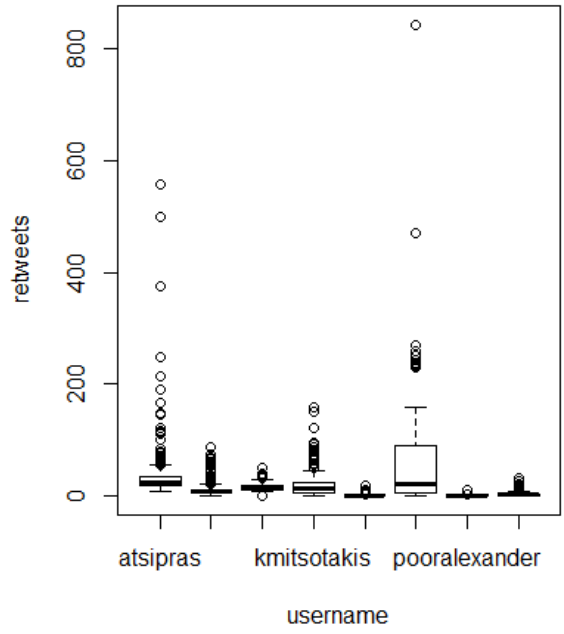


Figure 4.12: box3

It seems that the aggregate results verified and the third boxplot it does not have only extreme values in 1 level but in the 6 as well. Before we saw the correlation of the 2 variables and now we will see it graphically with the help of the scatterplot.

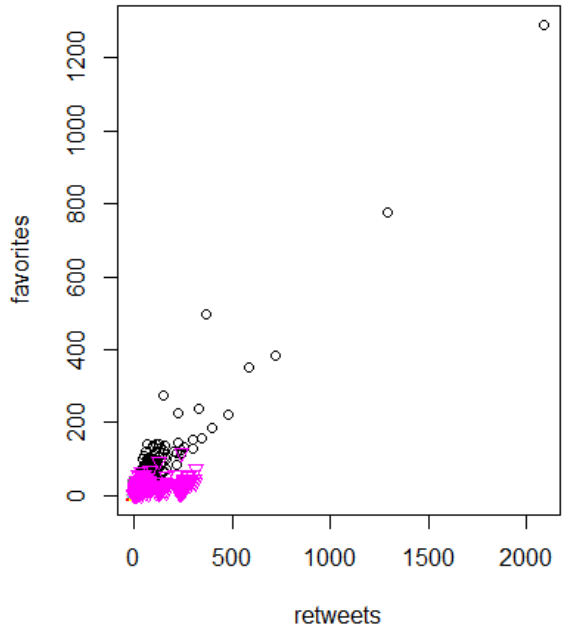


Figure 4.13: scatter1

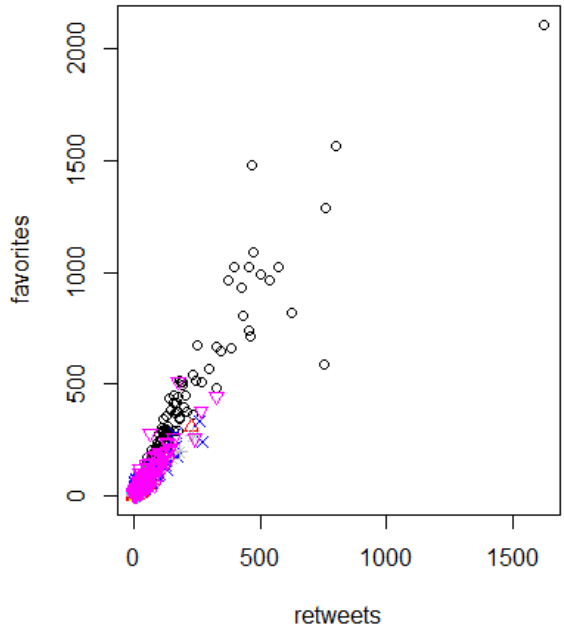


Figure 4.14: scatter2

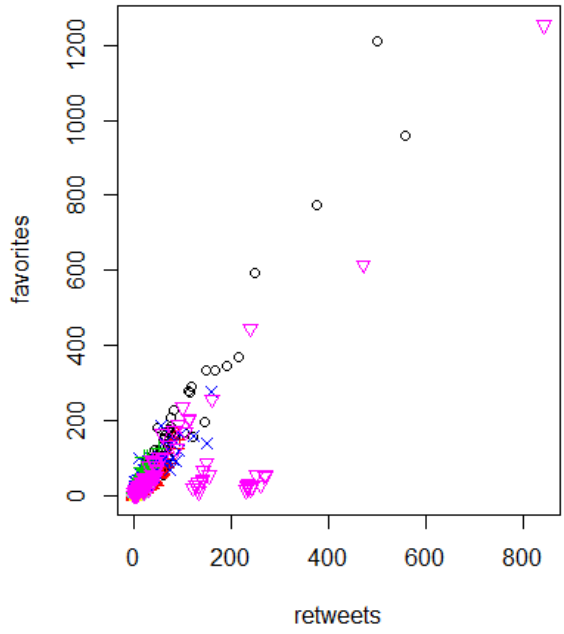


Figure 4.15: scatter3

Our scatterplot confirms the correlation coefficients we found earlier. Also the colors in the graph refers to the username levels. However, we can still see the correlation of the 2 variables for each username separately with the help of the qplot.

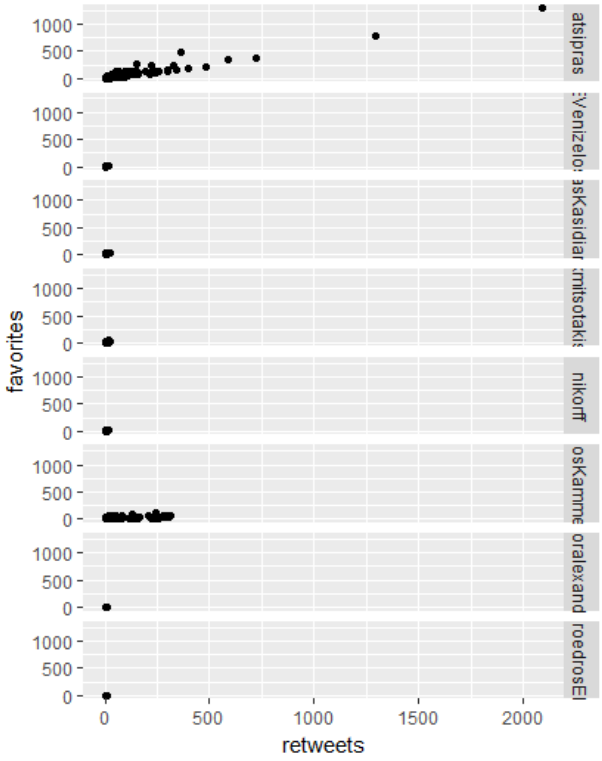


Figure 4.16: qplot1

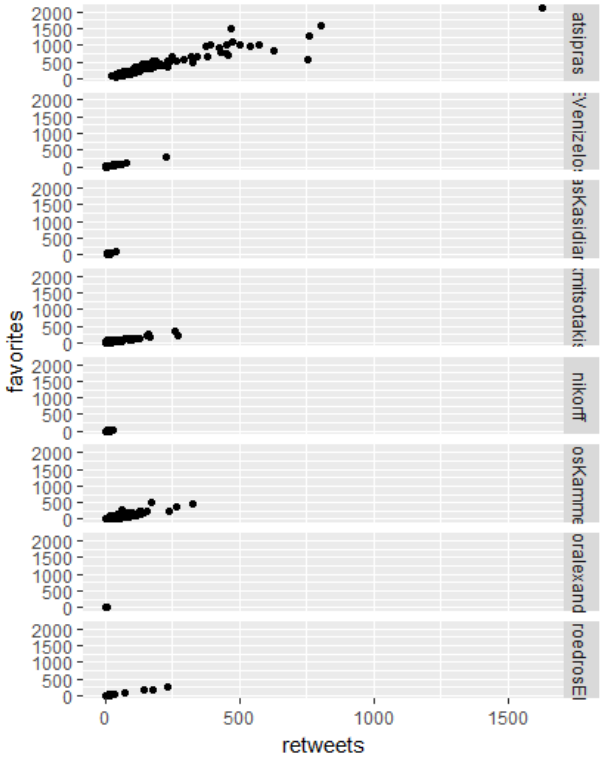


Figure 4.17: qplot2

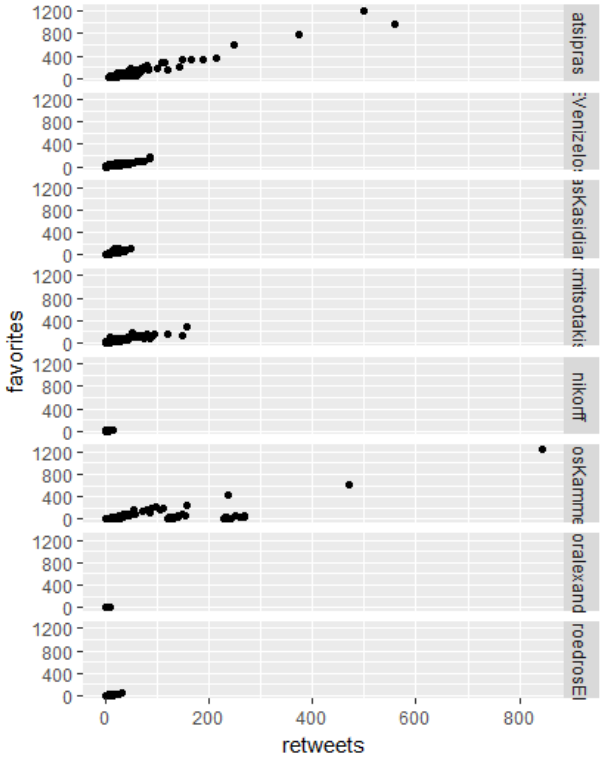


Figure 4.18: qplot3

We see outliers for the level "atsipras" and to the three data, and in the last data we notice that the level "PanosKammenos" has also extreme values.
In this paragraph we explored and visualized each variable individually depending on whether it was quantitative or qualitative. We saw the quantiles of the variables where it was feasible, and we made the graphs for all variables separately, depending on what they needed to do for each one of them. Inside the charts that made an impression was the histogram, the piechart and the qplot, with the last of them to showing the relationship between "retweets" and "favorites" based on "usernames".

4.2 Regression

In this section we will represent some of the most well-known regression techniques. First we will take a look at the results of linear regression and after that we will also have a view in logistic regression. Let's have a look in the summary of linear regression.

```
Call:
lm(formula = retweets ~ username + date + favorites,
data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-404.15    -9.32    -2.27     6.81   212.99

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.260e+03  7.677e+01  16.410  < 2e-16 ***
EVenizelos       1.038e+00  2.242e+00   0.463  0.64357
IliasKasidiaris -2.519e+01  2.660e+00  -9.469  < 2e-16 ***
kmitsotakis     -4.545e-01  2.374e+00  -0.191  0.84822
nikorff          6.427e+00  1.609e+00   3.994  6.58e-05 ***
PanosKammenos    4.200e+01  1.626e+00  25.834  < 2e-16 ***
pooralexander    1.221e+01  2.831e+00   4.315  1.63e-05 ***
ProedrosEK       5.392e+00  1.858e+00   2.902  0.00373 **
date            -9.002e-07  5.448e-08 -16.524  < 2e-16 ***
favorites        1.594e+00  1.926e-02  82.731  < 2e-16 ***
---

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

Residual standard error: 36.7 on 5008 degrees of freedom
Multiple R-squared:  0.6349,    Adjusted R-squared:  0.6342
F-statistic: 967.6 on 9 and 5008 DF,  p-value: < 2.2e-16
```

Summary for the referendum dataset

```
Call:
```

```
lm(formula = retweets ~ username + date + favorites,
data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-355.47    -1.71    -0.25     0.73   441.57

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)      -6.628e+01  1.018e+02  -0.651  0.515
EVenizelos        2.910e+01  2.203e+00  13.206  < 2e-16 ***
IliasKasidiaris   2.072e+01  3.125e+00   6.630  3.91e-11 ***
kmitsotakis       2.893e+01  2.249e+00  12.863  < 2e-16 ***
nikorff           2.791e+01  2.331e+00  11.974  < 2e-16 ***
PanosKammenos     2.981e+01  2.238e+00  13.319  < 2e-16 ***
pooralexander     2.759e+01  2.260e+00  12.208  < 2e-16 ***
ProedrosEK        2.889e+01  2.378e+00  12.151  < 2e-16 ***
date              2.699e-08  7.125e-08   0.379  0.705
favorites         5.749e-01  4.213e-03 136.435  < 2e-16 ***

---

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

Residual standard error: 17.16 on 3258 degrees of freedom
Multiple R-squared:  0.9081,    Adjusted R-squared:  0.9079
F-statistic:  3577 on 9 and 3258 DF,  p-value: < 2.2e-16
```

Summary for the elections2 dataset

```
Call:
lm(formula = retweets ~ username + date + favorites,
data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-147.178   -3.309   -0.743    2.783  206.830
```

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -2.186e+03  2.517e+02  -8.688  < 2e-16 ***
EVenizelos       7.616e+00  1.075e+00   7.087  1.76e-12 ***
IliasKasidiaris -1.338e+00  1.541e+00  -0.868  0.385302
kmitsotakis      4.524e+00  1.354e+00   3.341  0.000846 ***
nikorff          6.367e+00  1.199e+00   5.309  1.20e-07 ***
PanosKammenos    4.062e+01  1.670e+00  24.324  < 2e-16 ***
pooralexander    5.455e+00  2.808e+00   1.943  0.052182
ProedrosEK       7.361e+00  1.183e+00   6.221  5.73e-10 ***
date             1.514e-06  1.745e-07   8.677  < 2e-16 ***
favorites        5.444e-01  6.548e-03  83.148  < 2e-16 ***

---

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

Residual standard error: 16.26 on 2631 degrees of freedom
Multiple R-squared:  0.7827,    Adjusted R-squared:  0.782
F-statistic:  1053 on 9 and 2631 DF,  p-value: < 2.2e-16
```

The first object in our summary is the formula R used to fit the dataset. The next thing in our output is the residuals. Residuals are the difference between the observed values and the predicted values. The residuals output cuts it down into 5 summary points. You should have a zero mean value for a symmetrical distribution. It seems that the residuals do not appear to be symmetrical. Next we have the coefficients. The most important things in the coefficients output are the **estimate** and the **p-value** with which we will deal in our analysis. For the elections dataset we see that the residuals are not symmetrical. The estimate for the intercept is 1.260e+03, which means that the average value for the retweets is that number when all the explanatory variables are set to zero. For the **favorites** variable we have that if we increase it by one unit, then the expected value of the retweets should be increased by 1.594e+00. Similarly for the **date** variable.
For the **username** variable, because it is a factor we have other interpretation. This means that one of the levels must be the indicator. In our example, the level "atsipras" is the indicator that's why it take on value of zero and this level will not appear in our output, to wit a model without that level. For p-value we have that the levels "EVenizelos" and "kmitsotakis" from the username variable are not important

to be in our model. From the last three we will focus only in the R-squared parameter. The R-square explains the percentage of variation in the response variable that is explained by variation in the explanatory varibles. In our example it explained $\approx 63\%$, which is good enough.
Similar interpretation for the referendum and elections2 datasets only that in estimate of the coefficients the variable **date** is not significant to included in the model, and the R-squared is almost $\approx 0.91\%$. It means that the model fit to the data extremely well. The form of the R-squared is the following:

$$R^2 = \frac{SSM}{SST}$$

where SST is the total sum of squares, and SSM is the sum of squares from the model. For the third dataset the levels "IliasKasidiaris" and "pooralexander" are not significant. Also the R-squared is about $\approx 0.78\%$ which means very good fit to the data.

To verify the residuals summary we will make the following plot:

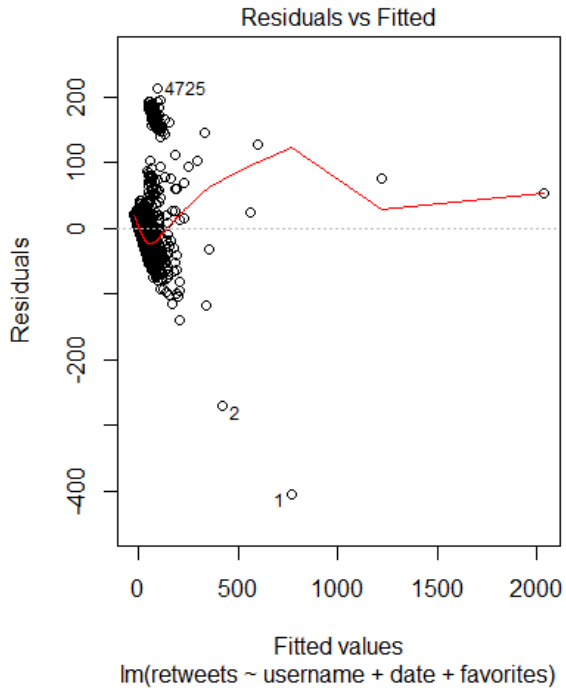


Figure 4.19: resid1

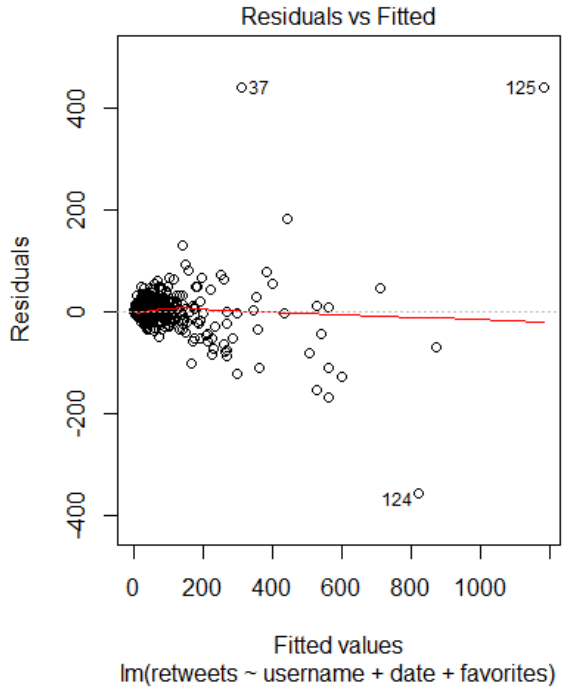


Figure 4.20: resid2

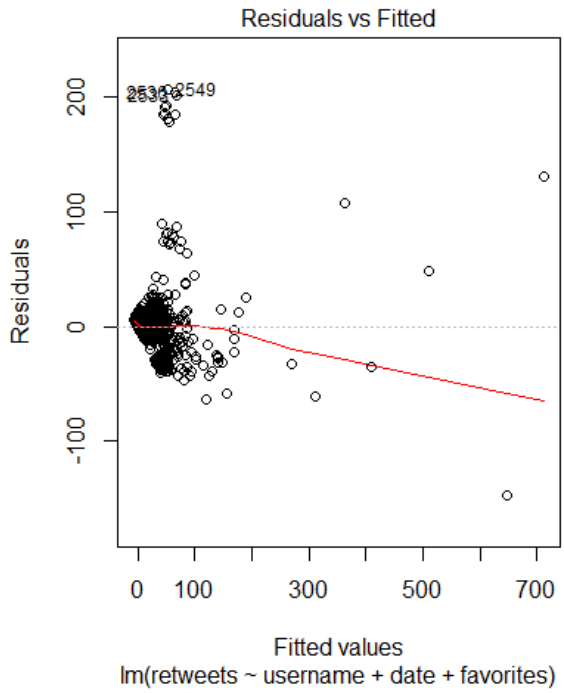


Figure 4.21: resid3

We observe that indeed the second plot is symmetrically distributed as we found earlier, and the other two they are not so symmetrical. Also with the help of the QQplot we can take a look in the normality of the data. We did not do it to save time for other graphs, and if you want to check it out there is in the code in the appendix.
Subsequently we will make the plot between the predicted values of our model and the **retweets** variable to see how well the predicted values fitted with the observed one's.

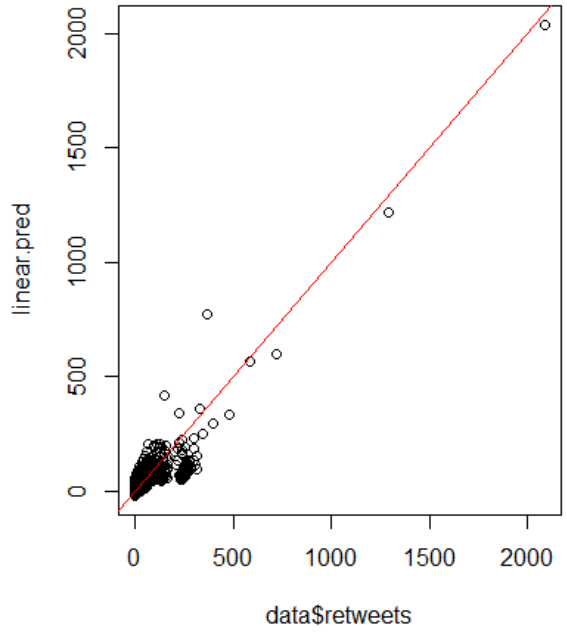


Figure 4.22: pred1

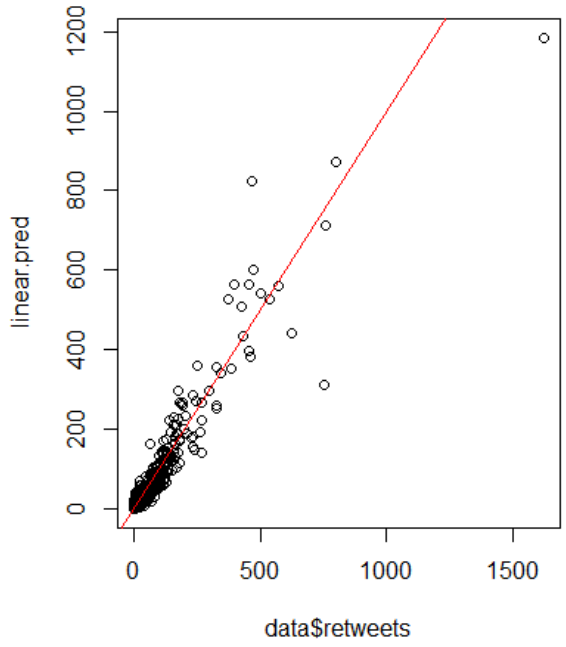


Figure 4.23: pred2

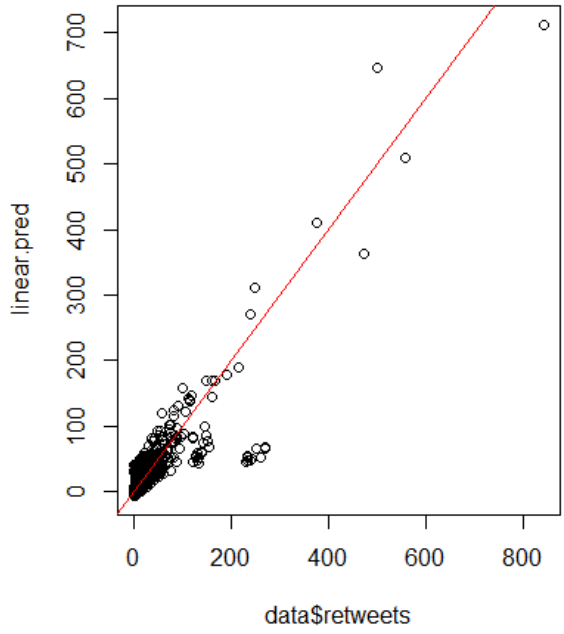


Figure 4.24: pred3

From the first and second plot, which are the election and referendum datasets respectively, we observe that our data fitted good enough in the regression line, with some outliers. In the elections2 dataset we see that the values are at a distance from the regression line, so as a result we have more extreme values than the other two datasets.
To check the **username** variable as a response one we will make a miltinomial logistic regression.I have already said some things in theory and now I will just show the boxplots that have the levels of this variable in relation to the predicted values of this model.

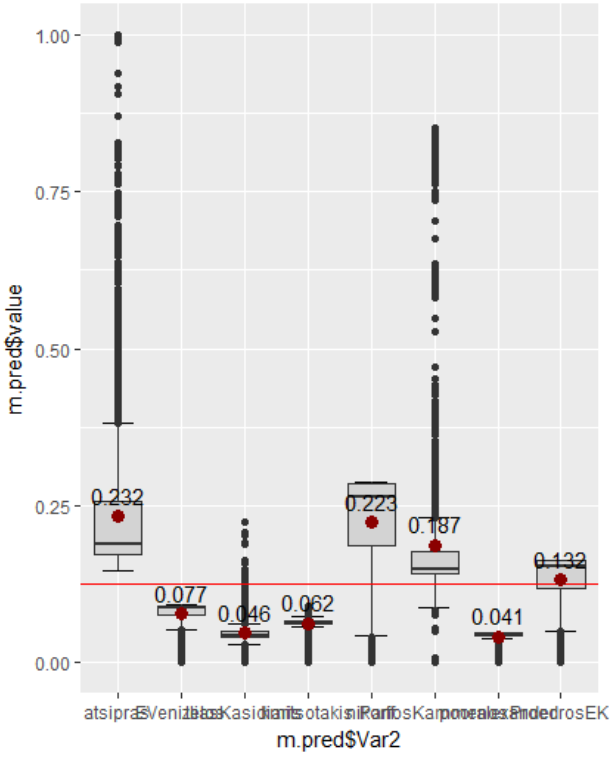


Figure 4.25: mpred1

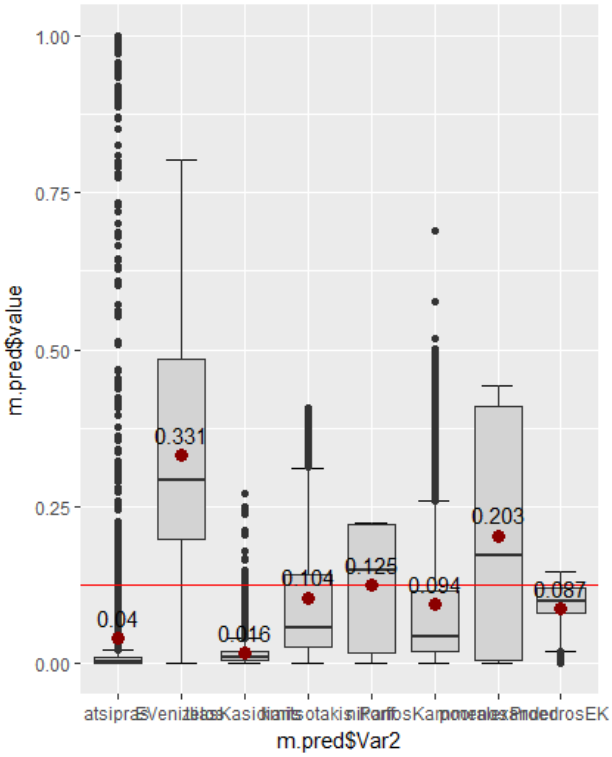


Figure 4.26: mpred2

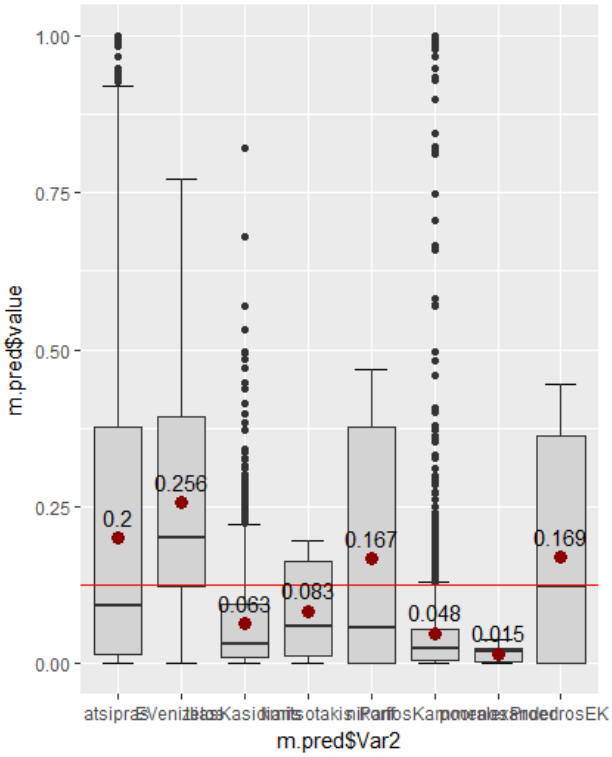


Figure 4.27: mpred3

In these boxplots we can distinguish two major things except from the quantiles. The first is the mean value for each level of the variable username which denoted by red dots, and the mean value of the predicted values in a red horizontal line. This will help to understand how close or far is every level, from the average of all the predicted prices. We can notice that and the three datasets have outliers, but in the referendum data we have 5 levels to have extreme values and in the elections2 data we drop into 3 levels only. This maybe means that these people who said the tweets were more careful about what to say in public.
In this paragraph we can see that our model has good fit in our data for all 3 datasets. This we can understand it by the value of R^2, which is bigger than 0.6 and in the 3 datasets. Also the predicted values with the retweets variable seems to have a good fit also, except maybe from the elections2 dataset which has more extreme values than the other 2. In the end we make a very interesting boxplots with the multinomial logistic and we found something very important. We observe that all the datasets have outliers but as we go on from the first elections to referendum data the outliers drop to 5 levels to have extreme values and in the elections2 data drop

even more into 3 levels only.

4.3 Clustering

In this paragraph we will talk about clustering techniques. The techniques that we will discuss are the *k-means clustering, k-medoids clustering, hierarchical clustering and the density-based clustering.* For all clustering techniques we have split our dataset into 2 subsets for better results. For k-means we have the cluster means which actually is the mean of the prices for every variable for those samples delegated to that cluster, which matches the center for each cluster. In our election data we take the following results:

```
K–means clustering with 2 clusters of sizes 8, 249

Cluster means:
   retweets favorites
1 230.87500 22.625000
2   9.39759  9.939759
```

The 75.1% is a measure of the total variance in your dataset that is explicated by the clustering. By recommending the samples to k clusters rather than n, where n are the number of clusters, we managed a decrease in sums of squares of 75.1%. Now let us compare the clusters with the variable **username**.

	1	2
atsipras	0	67
EVenizelos	0	13
IliasKasidiaris	0	10
kmitsotakis	0	15
nikorff	0	46
PanosKammenos	8	51
pooralexander	0	15
ProedrosEK	0	32

As we can observe, the data befalling to the "PanosKammenos" username got grouped into cluster 1, and all the other levels into cluster 2. Similar interpretation for the other two datasets, which can be found in the excel files, but in the elections2 dataset we have some differences which we will see from the plots right now.

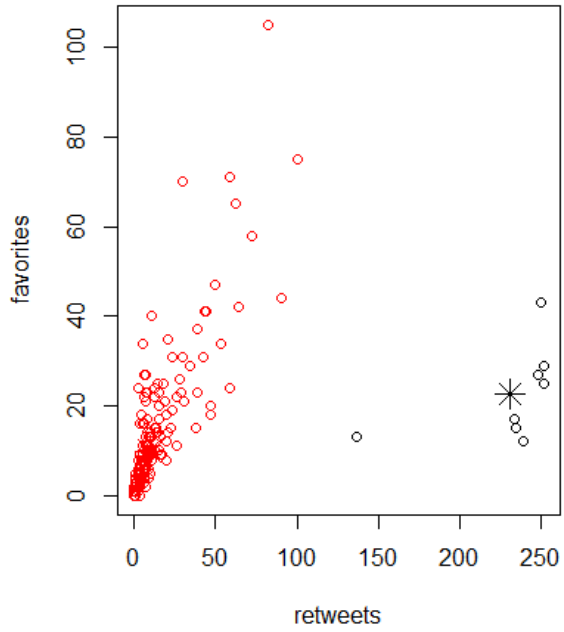


Figure 4.28: kmeans1

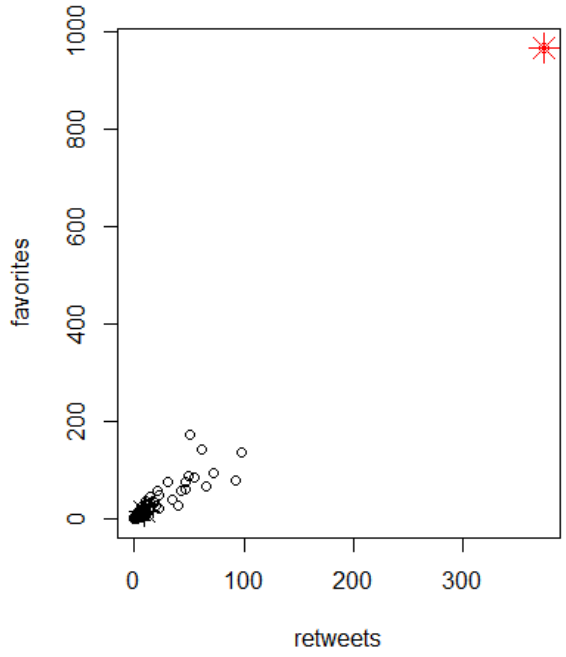


Figure 4.29: kmeans2

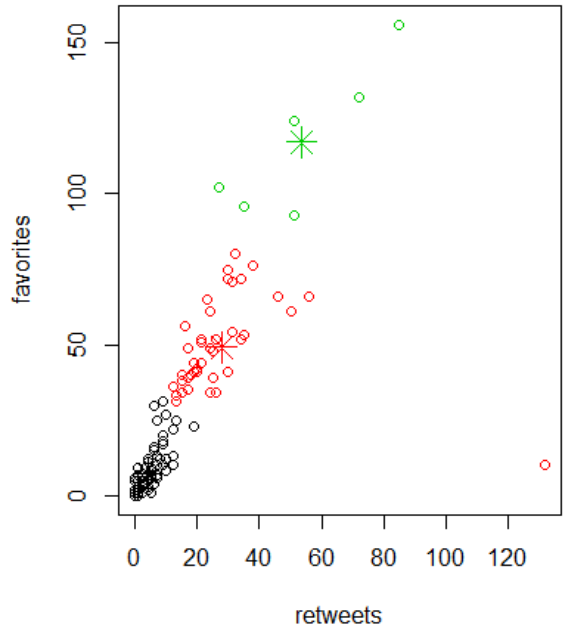


Figure 4.30: kmeans3

We notice from the scatterplots that the data in the elections2 plot have 3 clusters, while the other two datasets have 2. This maybe means that because of the short period of time some candidates could not express their opinions with logical arguments and they ended up in extreme positions. Perhaps can explain the 3 clusters and the outliers we have in the elections2 dataset.
For the k-medoids we have the same interpretations for the three datasets, but we have different plots. We have two plots, the clusplot and the silhouette plot. We take the following results:

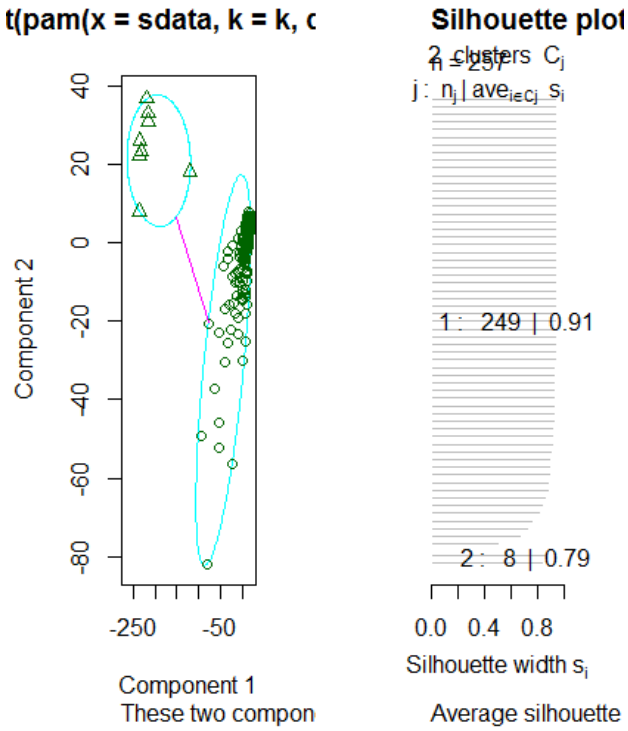


Figure 4.31: clus1

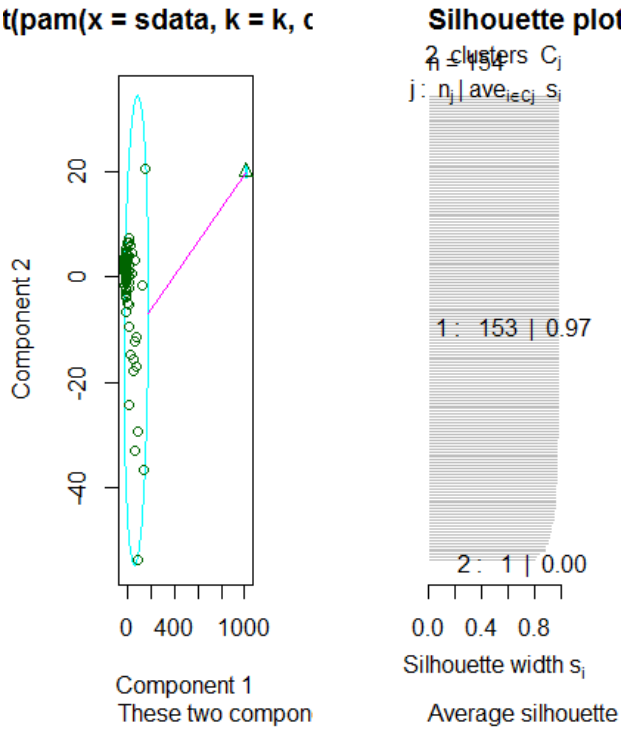


Figure 4.32: clus2

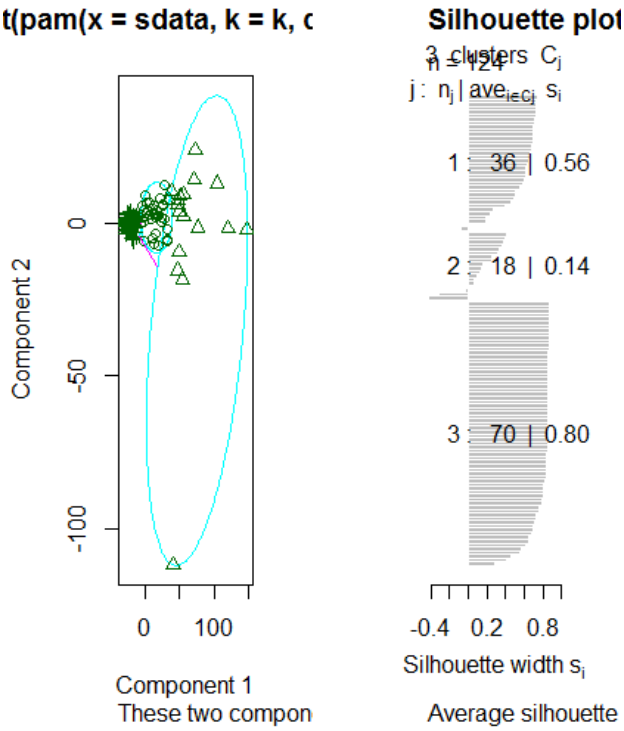


Figure 4.33: clus3

The lines between the two and the three clusters respectively showing the distance between them. For the silhouette plot we want to have large s_i almost to 1 for better clustering. In the election data the 1st cluster has 0.91 and the 2nd 0.79. That means that corresponding observations are very well clustered. For the referendum data we have the second cluster at 0 value. This means that the observations in that cluster lies between the two clusters. For the elections2 data we have that the second cluster lies between the other two. Since the average S_i are greater than 0.5 for all cases, it means that and the three datasets are well clustered.
For the hierarchical clustering we have the following plots. These plots called dendrograms and they have the following form:

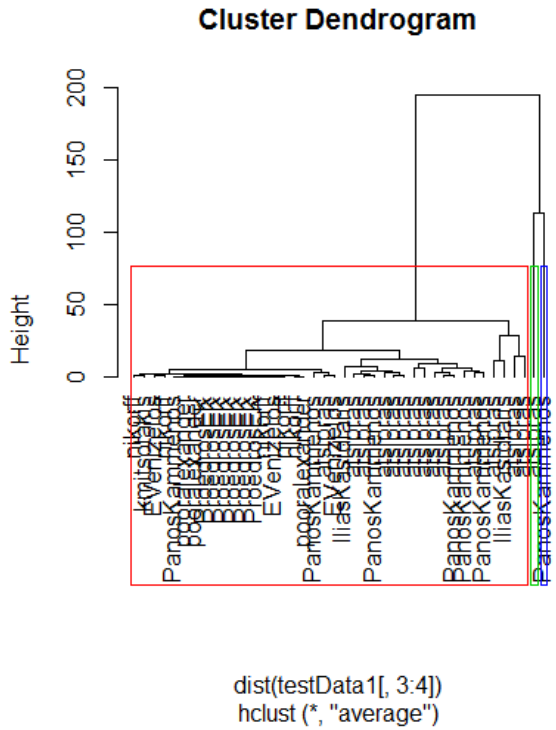


Figure 4.34: hclust1

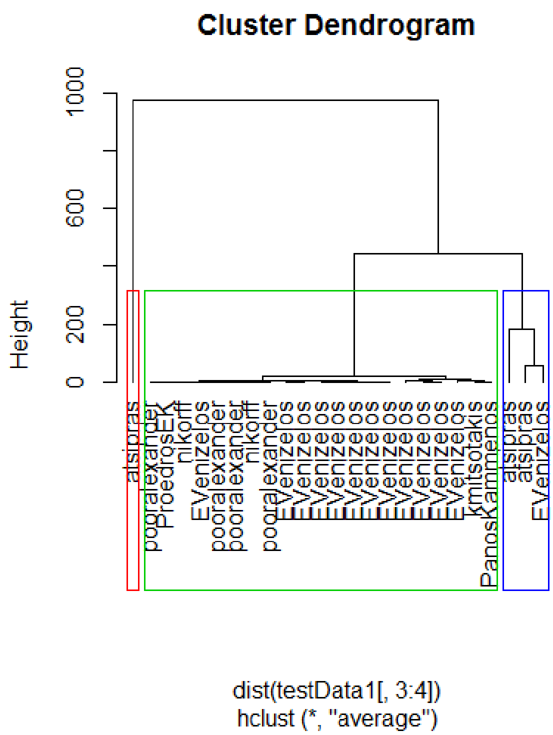


Figure 4.35: hclust2

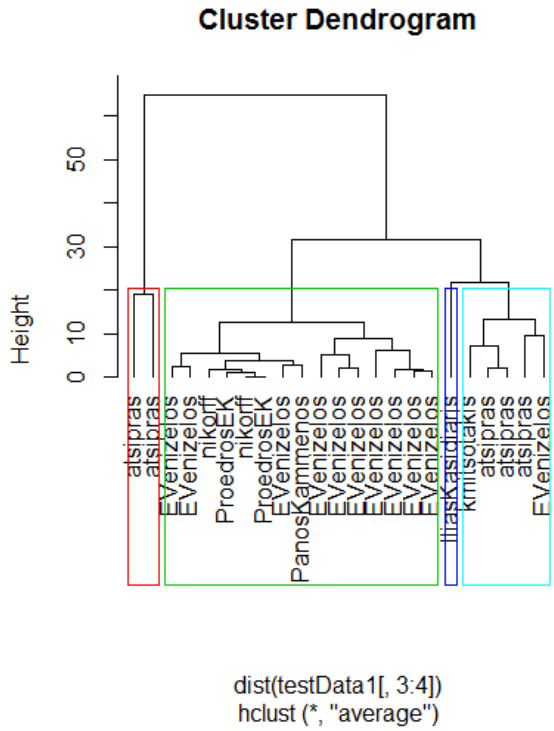


Figure 4.36: hclust3

From the previous dendrograms we perceive that the election and the referendum datasets have 3 clusters, while the third has 4 clusters. We observe that the two methods, to wit the k-means and the hierarchical clustering differ to each other by one cluster. For example, the election data with the k-means technique has 2 clusters while with the hierarchical has 3 clusters. The density-based method left to the reader for practice.

At this point we will raise the issue of outlier detection by clustering with the help of the scatterplot. For this detection we will use all the data for each one dataset separately, and not the two split datasets.

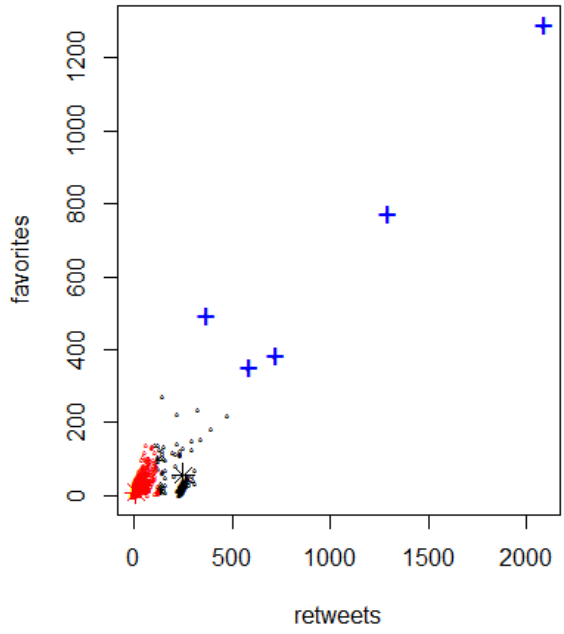


Figure 4.37: out1

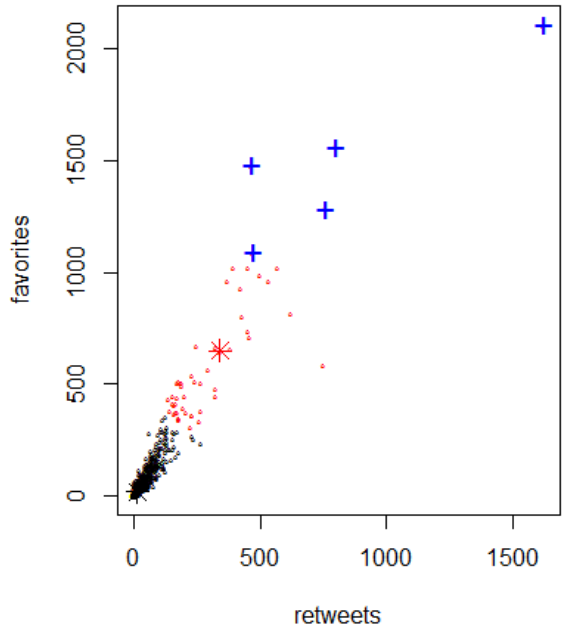


Figure 4.38: out2

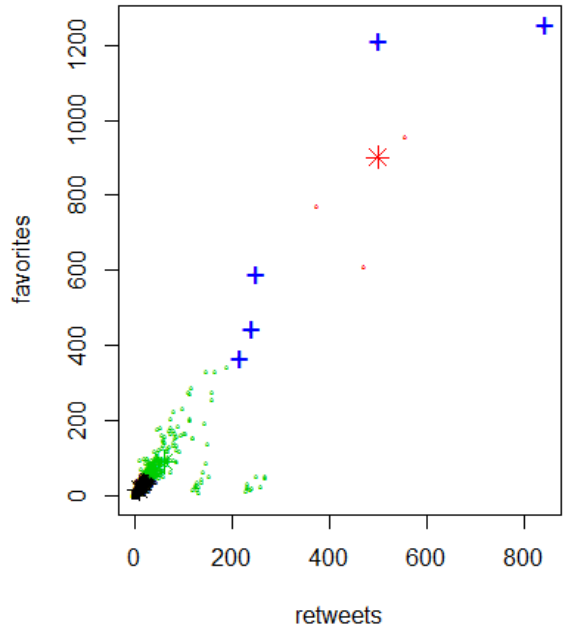


Figure 4.39: out3

We observe that the first 5 outliers for the election dataset fall into the 1st cluster, while for the referendum dataset fall into the 2nd cluster. The elections2 dataset has 3 of 5 extreme values to fall into the the 3rd cluster and the remaining 2 to fall into the 2nd cluster.
As a conclusion we can say that the number of clusters ranges from 2 to 4 depending on each method we use at a time. This means that our observations are quite grouped together, and there is virtually no need to do all of these methods to draw our conclusions. Only one of these methods is enough to understand how many groups our data have.

4.4 Time Series

In this section we will talk about time series analysis and mining. First of all because our datasets is not a time series data we will converted to it with the function *xts()*. The converted data frame concerns the variables **date** and **retweets**. Now we will see the time series plots, which are composed by monthly observations of the elections, referendum and elections2 data from 2014-2015.They have 12(=1*12) values.

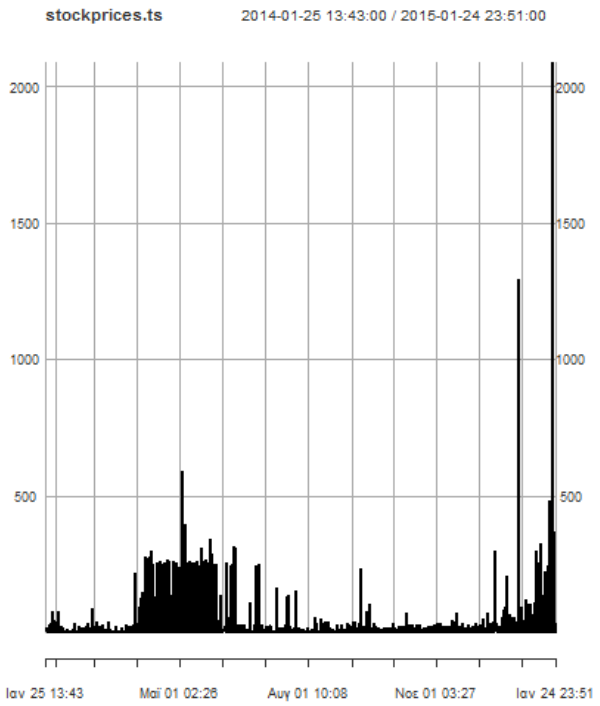


Figure 4.40: time1

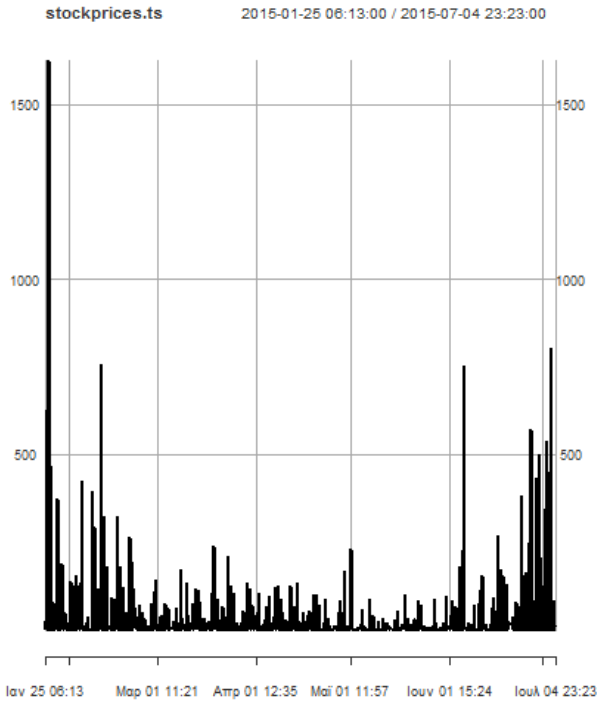


Figure 4.41: time2

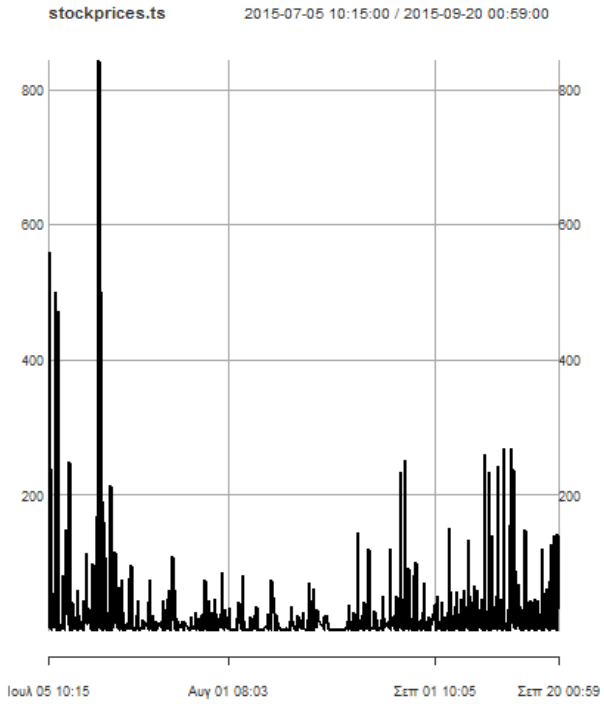


Figure 4.42: time3

From these three datasets we can notice that the extreme values for the referendum and the elections2 datasets are in the first observations, while for the election dataset is in the last observations. An interpretation could be that from April to June of 2014 we have approximately 200-600 retweets for the election dataset. For the same period, but in 2015 in the referendum dataset, we have approximately 150-250 retweets, and for the elections2 dataset from August to September in 2015 year we have a range from 100-250 retweets. We showed this chart because it is easier to interpret it. There is the code for the original plot in time series in the appendix.

Let's see what information we can take from the next plots.

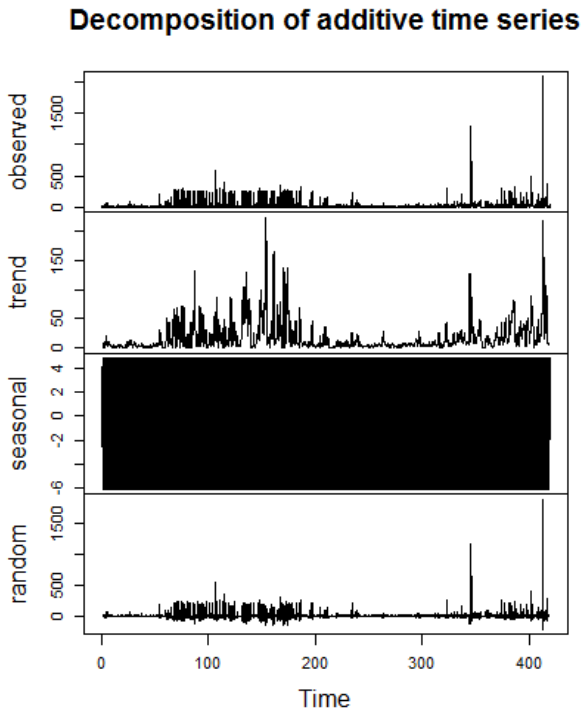


Figure 4.43: deco1

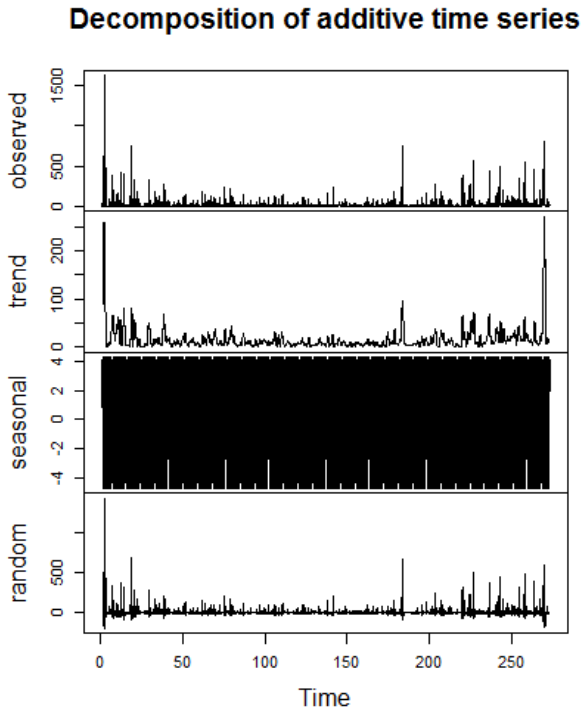


Figure 4.44: deco2

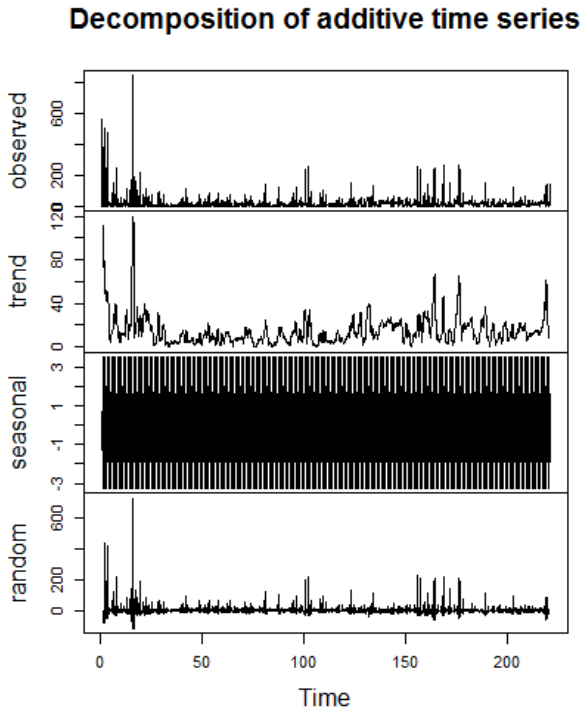


Figure 4.45: deco3

From these plots except from the original time series chart we can perceive a trend plot which shows the trend of the observations, a seasonal plot which shows the seasonal factors and the random shows the remaining components after removing trend and seasonal factors. We can distinguish that the observed and trend plots have similar behavior, while it seems to not have any seasonal factors because our data concern only one year of observations. As we saw earlier with the help of the time-series plot, we can also see that the trend graph shows an upward trend in the months of April to June of 2014 in the election data. On the trend graph of the referendum data, we mainly have an upward trend from June to July of 2015, and on the trend graph from the elections2 data we have an upward trend from August to September of 2015.
Before we closing this section we will talk about the time series forecasting. We will achieve this forecast with the help of the following plots:

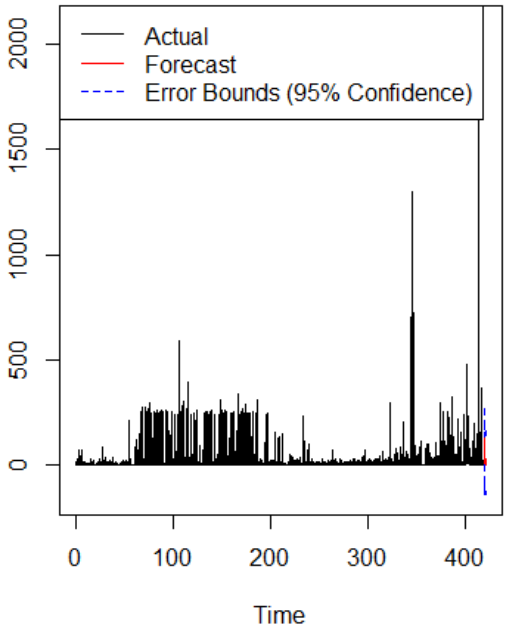


Figure 4.46: fore1

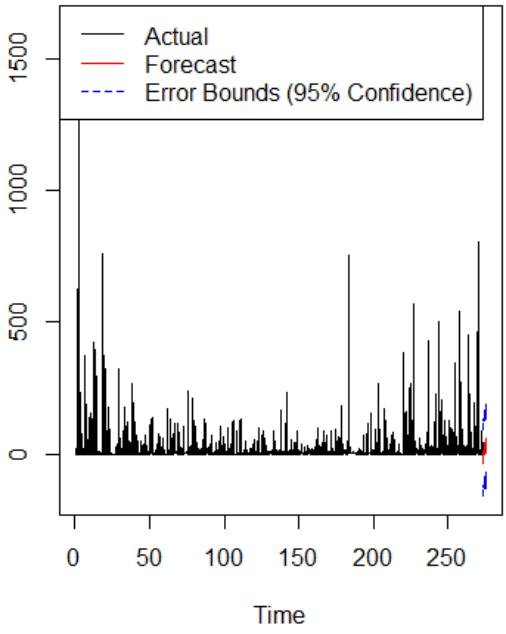


Figure 4.47: fore2

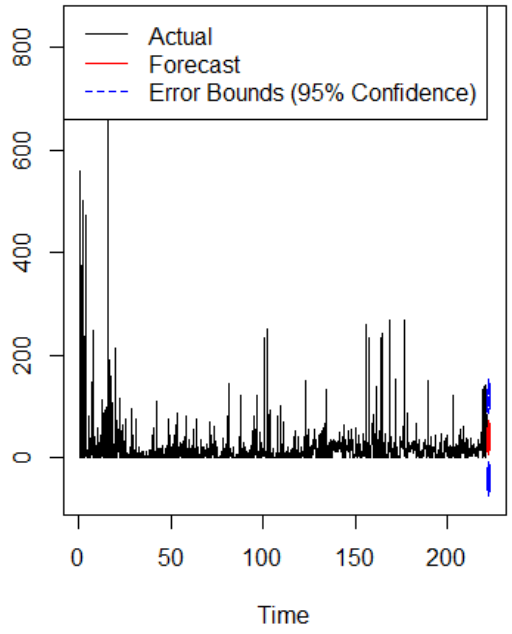


Figure 4.48: fore3

We discern a red solid line which are the forecasted values, and the two blue dotted lines which are the error bounds at a cofidence interval of 95%.The forecasts in all three datasets show that they will continue roughly the course they have with a probability of error of 5%.
As a general conclusion, we conclude that in some months we have some increase in retweets because those periods are either pre-election periods or a major event occurred. Also, the predictions we made in the three figures seem to be of no great effect.

4.5 Text Mining

In this section we will discuss about text mining in R. We have describe the process in chapter 3 and now we will show you only the results. Also we have change the data from Greek to English for R, because R can only understand English with the Natural Language Process that supplies, and yet we have split the dataset into two for faster and better results. In this section and the next one we will use mainly the **text** variable and almost none of the remaining variables. Because we have already described the process of how to make a term-document matrix, we will move on to the Frequent Terms and associations. To show the most frequent words visually we make a barplot for them.

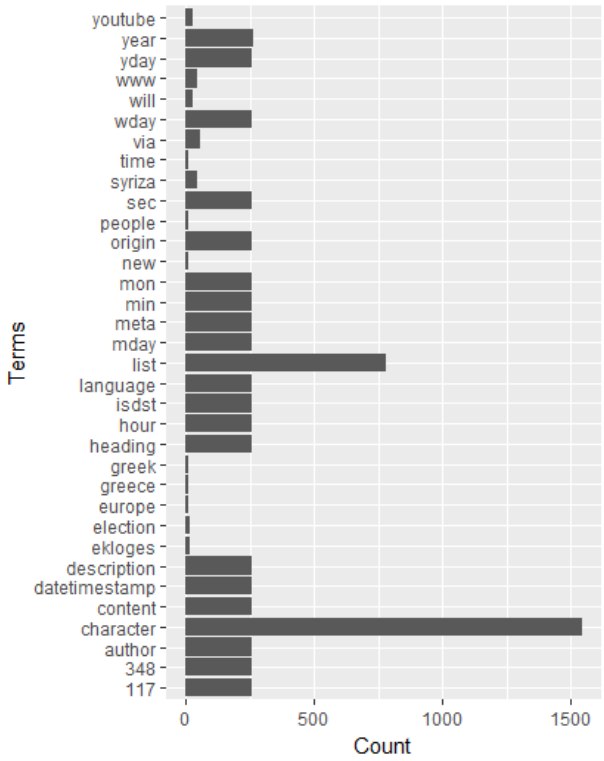


Figure 4.49: freq1

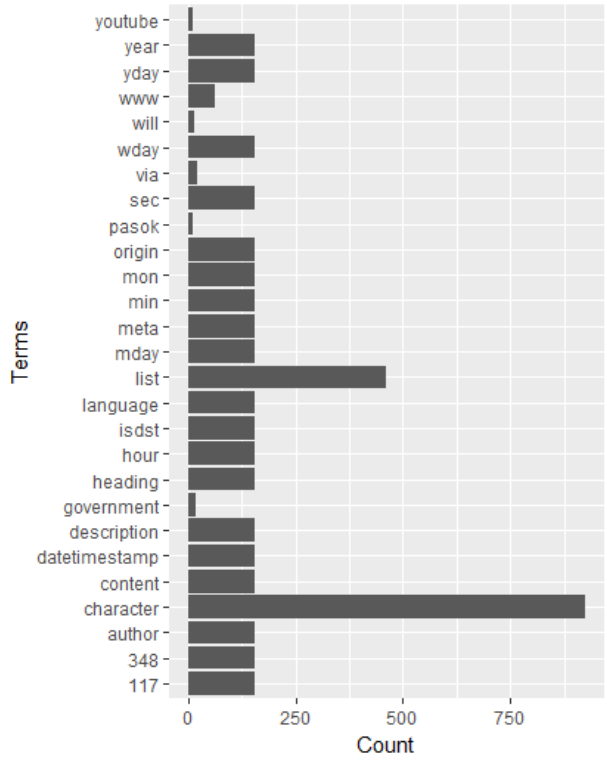


Figure 4.50: freq2

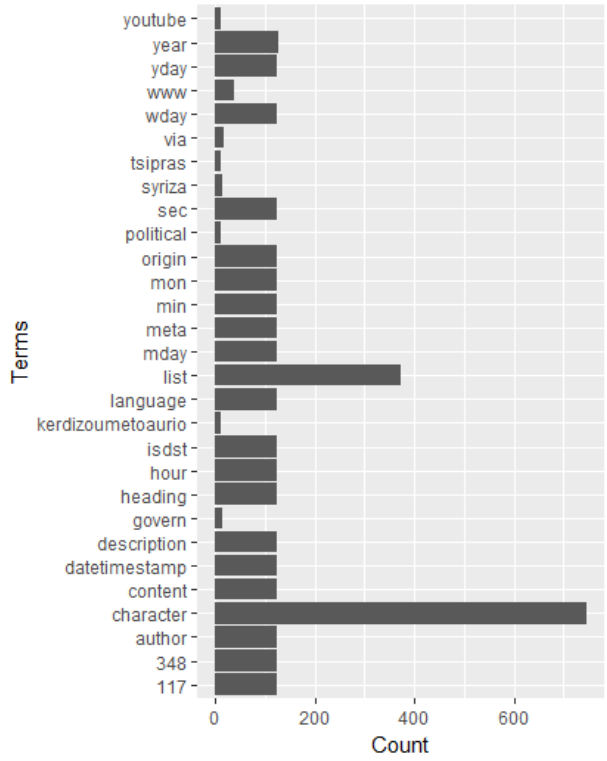


Figure 4.51: freq3

We see 3 barplots with termFrequency>= 10. Each barplot contains different number of terms. For example in the first barplot, which contains the election data, we see 34 terms, in the referendum barplot 27 terms and in the elections2 barplot 29 terms. The 2 most frequent terms are the "character" and the "list" terms.
After this we can see the importance of words with a word cloud. In the function wordcloud() the first 2 parameters are the words and their frequencies. Also we can put in it a parameter for mininmum frequency and for order. Let's see the wordclouds visually.

Figure 4.52: cloud1

Figure 4.53: cloud2

Figure 4.54: cloud3

From the wordclouds above we see that the top 2 words are the "character" and the "list", which confirm that the election, referendum and elections2 tweets represent information on R and data mining.
Now we will converse about how to clustering the words in R with the hierarchical clustering. We have described the procedure in the previous chapter and now we will see the graphs visually.

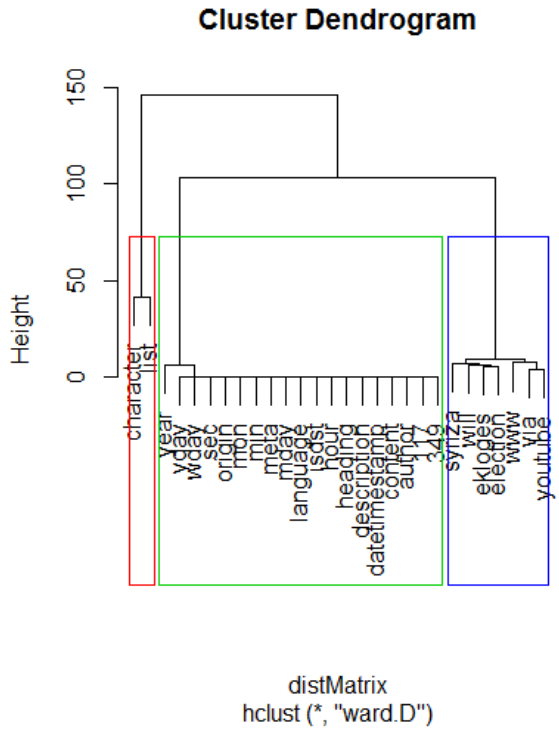


Figure 4.55: clden1

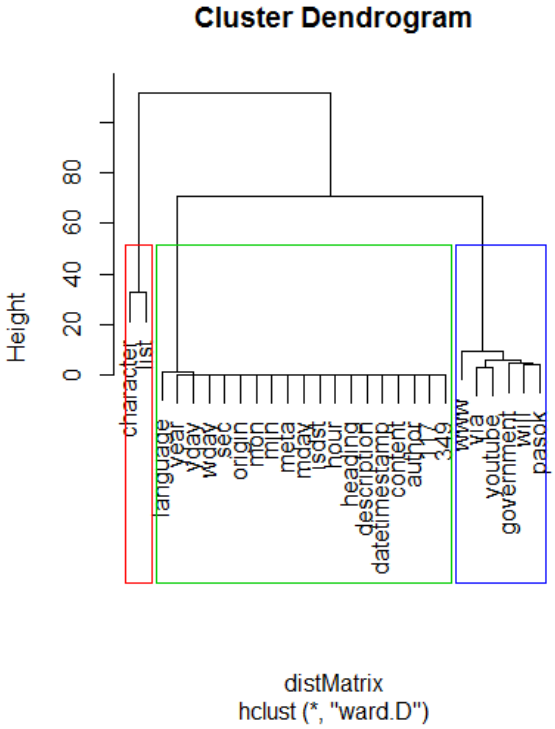


Figure 4.56: clden2

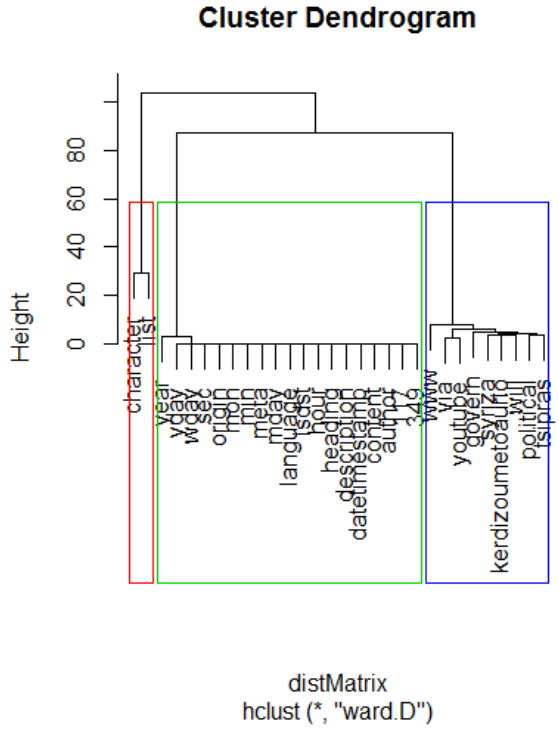


Figure 4.57: clden3

In the previous dendrograms we can see 3 clusters. In the election dataset we can notice that the terms “character” and “list” are always together into one cluster, and the next words are clustered by time mainly in a second cluster, coloured by green, and the remaining words are clustered by voting terms in a third cluster, coloured by red. In the referendum dataset we have again 3 clusters, one for the “character” and “list” terms, one for terms with timing meaning and one with what the “usernames” which include in this thesis said about the parliament. In the elections2 dataset we have 3 clusters once again with exactly the same meaning as the other two.

Before closing we will confer about how to cluster the tweets with the k-means and k-medoids techniques. For clustering the tweets with the k-means algorithm we take the prices in the matrix as numeric. We transpose the term-document matrix to a document-term one. The tweets are then clustered with kmeans() with the number of clusters set to eight [book 12, ref4]. Also we control the famous words in each cluster and also the cluster centers. The cluster centers can be found in the R code and we will not analyze it in this chapter. To find out what the clusters contain, we can examine the top three words in each cluster. For the election dataset

```
cluster 1: character list 117
cluster 2: character list syriza
cluster 3: character list year
cluster 4: character list 117
cluster 5: character year list
cluster 6: character list syriza
cluster 7: character list 117
cluster 8: character list 117
cluster 9: character list will
cluster 10: character list 117
```

For the referendum dataset

```
cluster 1: character list language
cluster 2: character list www
cluster 3: character list pasok
cluster 4: character list 117
cluster 5: character list 117
cluster 6: character list 117
cluster 7: character list 117
cluster 8: character list 117
cluster 9: character list 117
cluster 10: character list 117
```

For the elections2 dataset

```
cluster 1: character list 117
cluster 2: character list year
cluster 3: character list 117
cluster 4: character list govern
cluster 5: character list 117
cluster 6: character list 117
```

```
cluster 7: character list 117
cluster 8: character list 117
cluster 9: character list govern
cluster 10: character list www
```

From the previous top three words for each of the 10 clusters we observe that all the clusters from all the datasets refer to three mainly terms, “character”, “list” and “117”. Also there are some words where are unique in every dataset. For example in the election data the special terms are “syriza” and “will”, in the referendum data “pasok” and “language”, and in the elections2 data the word “govern”.
Now we will use the k-medoids algorithm for clustering the tweets because it is more robust to noise and extreme values than k-means clustering, and also gives a figure of the silhouette plot to show the quality of clustering. In the meantime let’s see the plots for this method.

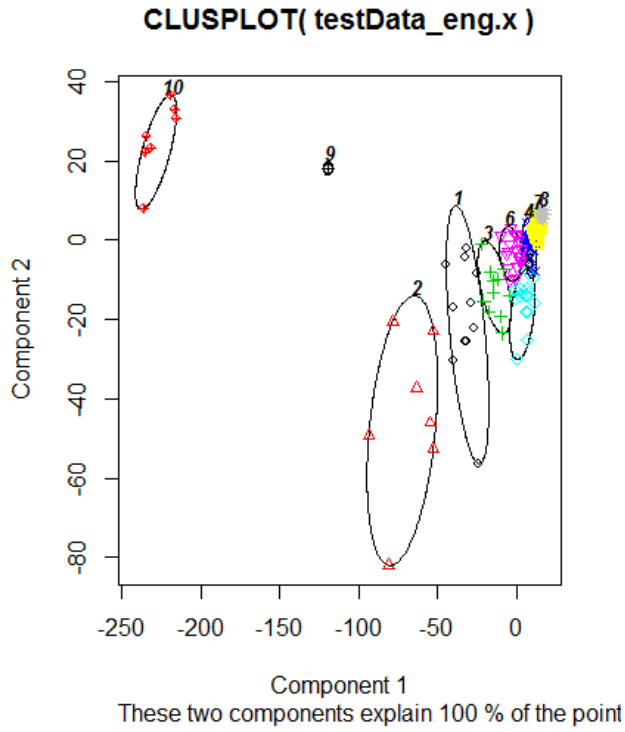


Figure 4.58: clus10

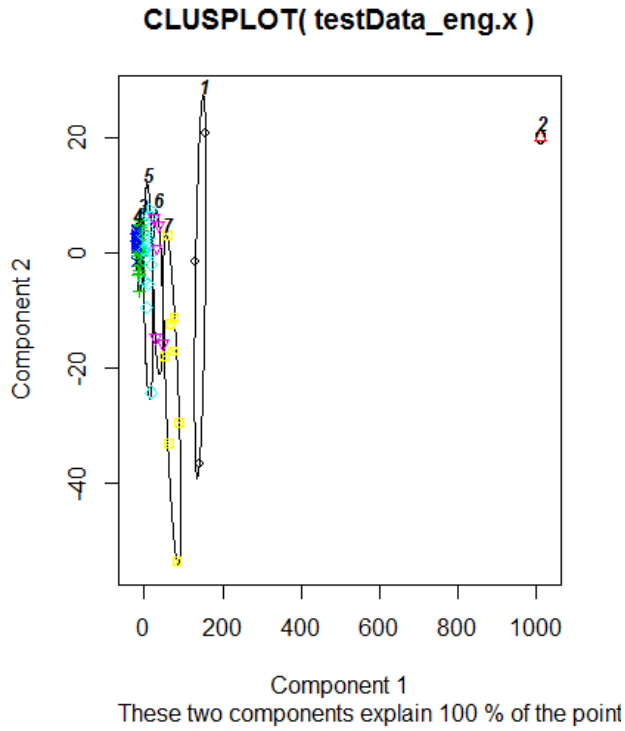


Figure 4.59: clus11

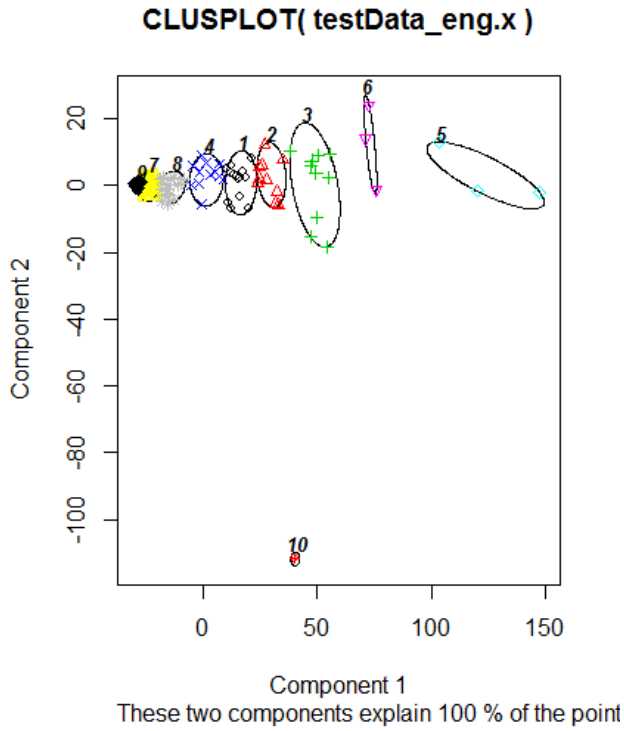


Figure 4.60: clus12

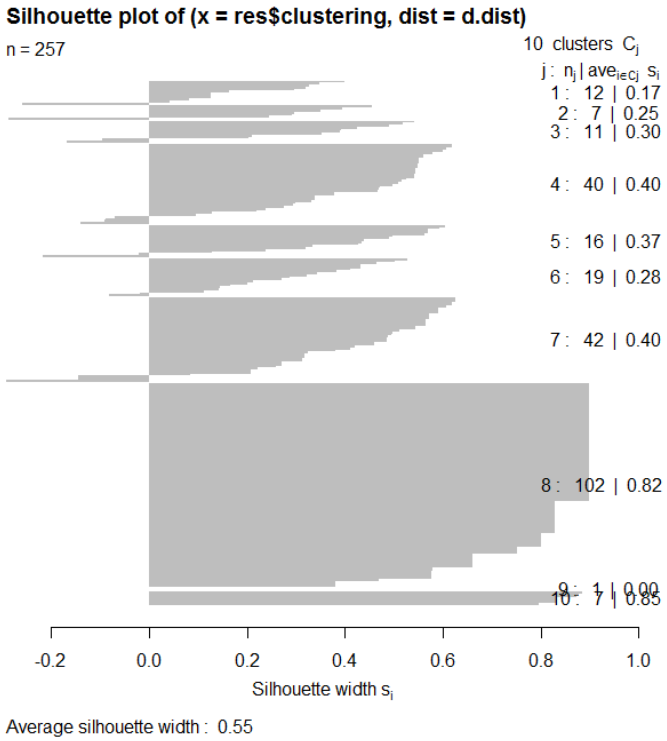


Figure 4.61: silh10

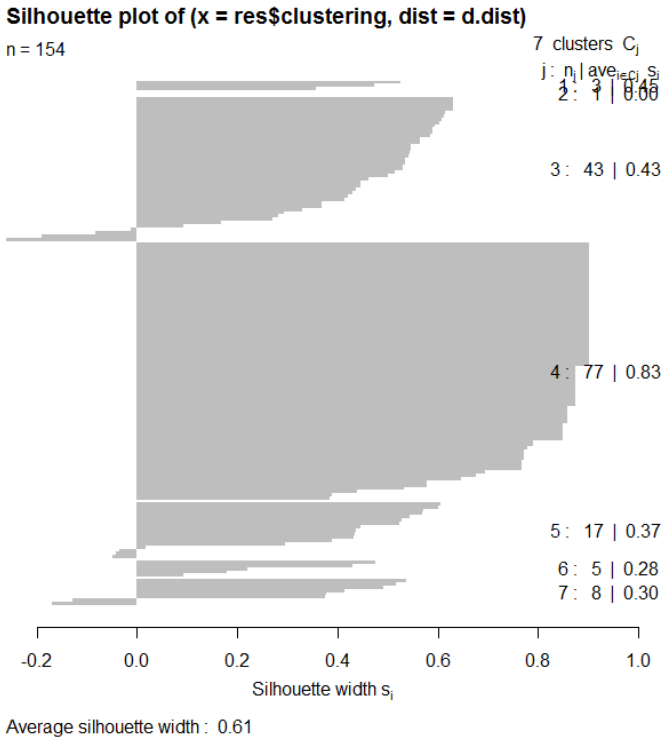


Figure 4.62: silh11

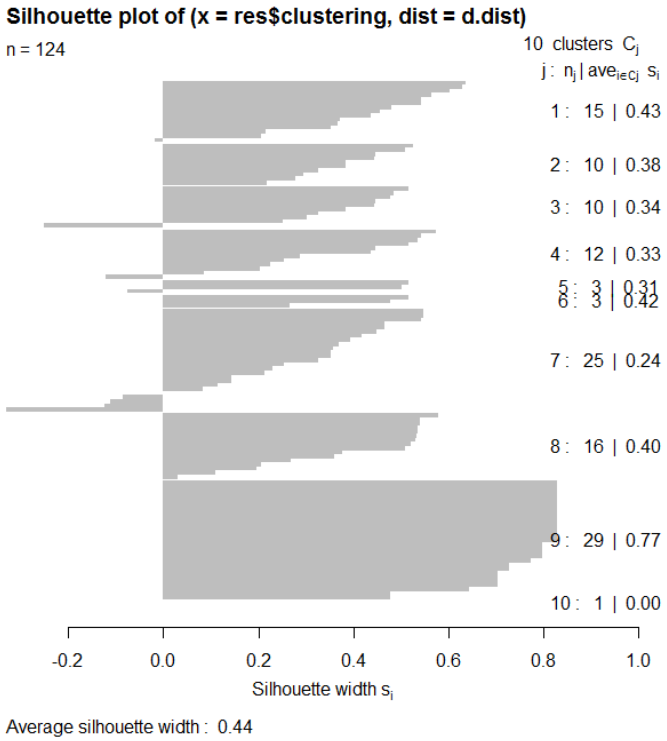


Figure 4.63: silh12

We have the clusplots which show us how the tweets are clustered, and also we have the silhouette plots which they analyze more the content of the clusters. From the average silhouette width and for the three datasets we can notice that the values are between 0.44-0.61. This means that clusters are moderate separated from one another. This looks even better in the silhouette width of each cluster separately.
In closing this section we will yet explore the scores of the sentiment analysis. The plots below shows that sentiment scores.

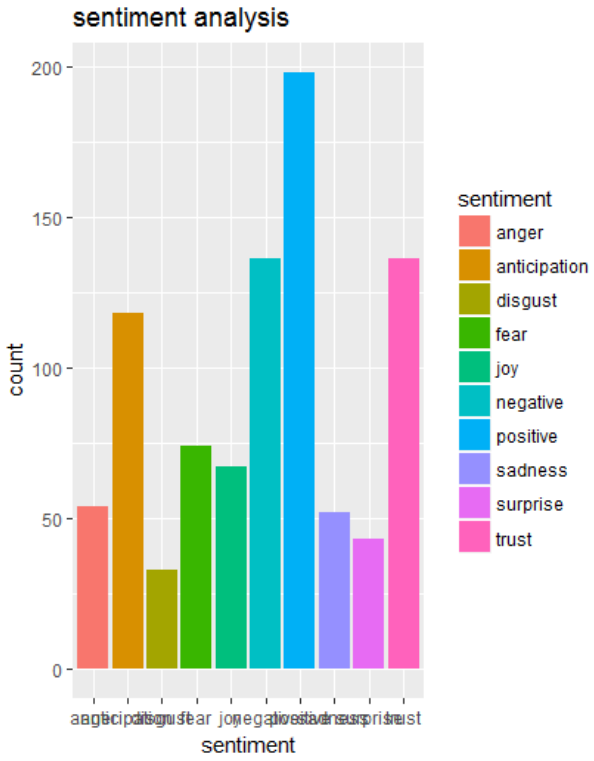


Figure 4.64: sent1

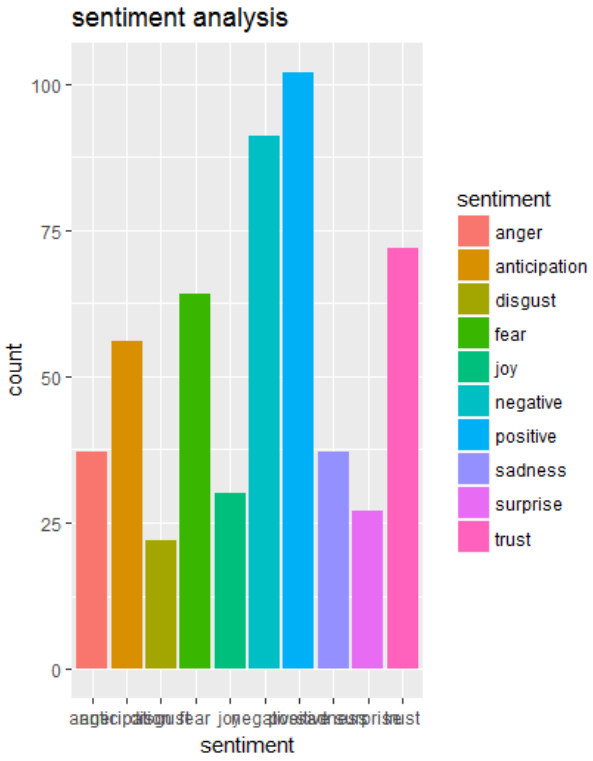


Figure 4.65: sent2

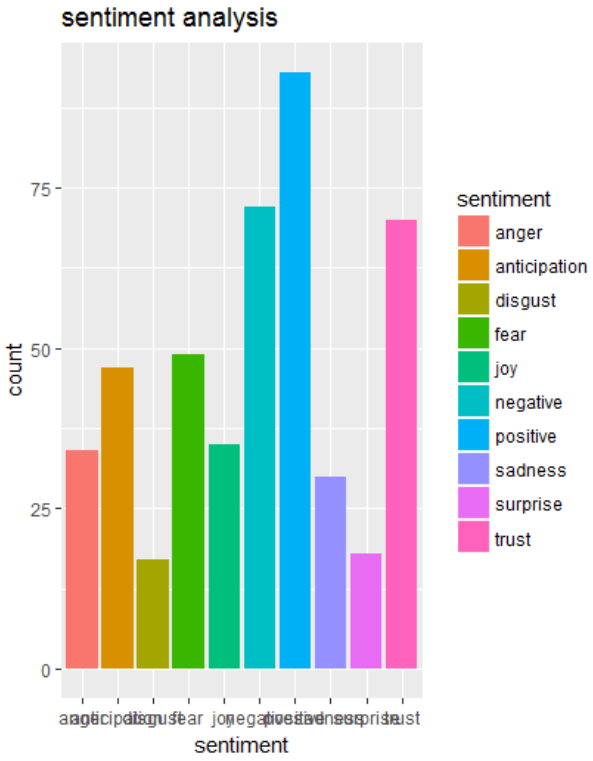


Figure 4.66: sent3

We notice that the 3 highest sentiment scores and for the three datasets are at the levels "negative", "positive", and "trust" of sentiment. For the remaining levels is not observed great variation, except from the "fear" level that seems to increasing in the referendum and elections2 datasets.
In this section, we have seen what are the most common terms presented in these data, and how many terms each has been given separately. We then saw how these terms can be grouped using the hierarchical analysis, which showed us that the terms are divided into 3 groups depending on their content. We also talked about how to group these terms using the k-means and k-medoids clustering. With the help of the k-medoids algorithm, we saw that 10, 7 and 10 groups respectively are spaced apart with each other. Finally, the sensitivity analysis showed us that the feelings that predominantly dominate our data are "negative", "positive", and "trust".

4.6 Social Network Analysis

In this section we will deal with the subject of the social network analysis and we will see some interesting plots in it. The data which we will use are from the previous chapter and we have do some conversions in them, which had described in precedent chapter, for better results. First we convert the tdm matrix to a normal matrix and after that in a term-term adjacency matrix for more prominent results. Now let's see the first plots of networks.

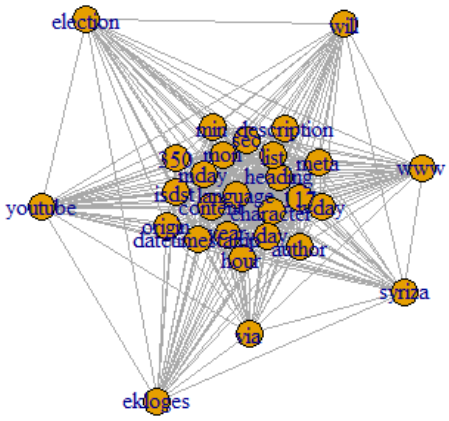


Figure 4.67: graph1

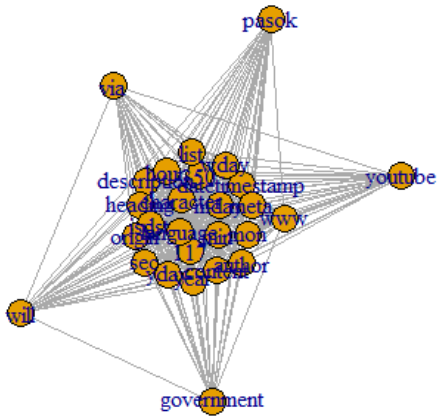


Figure 4.68: graph2

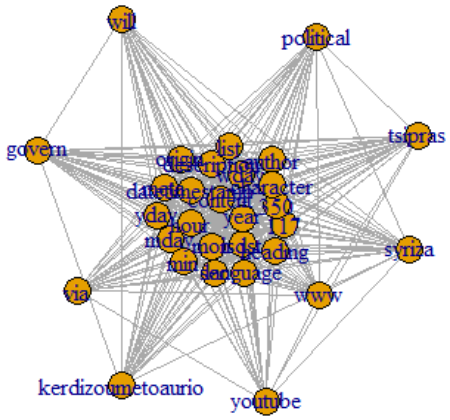


Figure 4.69: graph3

We can discern that the graphs have different shapes because of the terms that contain every dataset. For example it seems that the election and the elections2 datasets have the shape of a polygon because the terms have more interactions between them. Next we set the label size of vertices based on their degrees, to make significant terms come out [book 12, ref4]. Also we define the width and transparency of edges based on their weights.

Figure 4.70: graph4

Figure 4.71: graph5

Figure 4.72: graph6

Next we have the plots for the cohesive blocks.

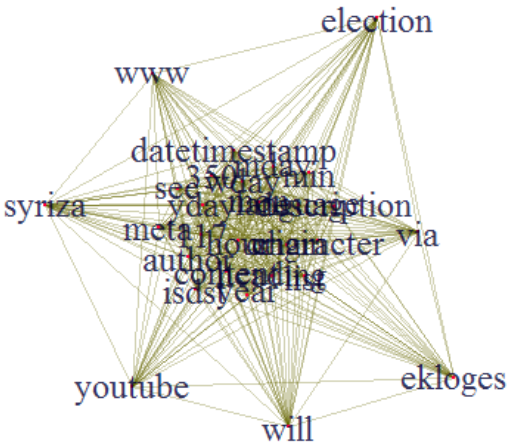


Figure 4.73: cliq1

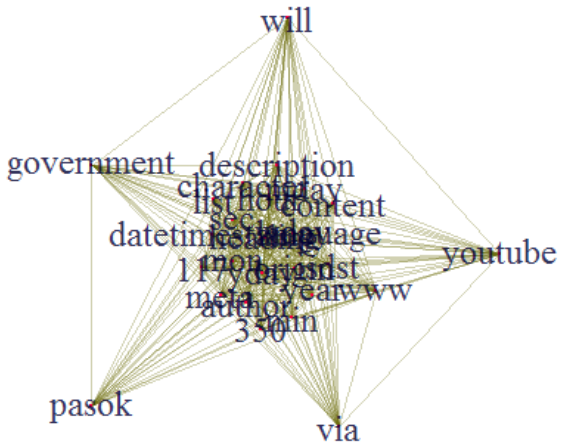


Figure 4.74: cliq2

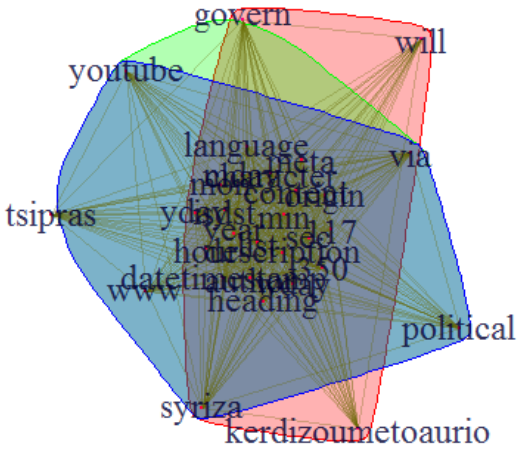


Figure 4.75: cliq3

From this plots we can detect the communities for the network of terms. In the election and referendum plots does not seems to exist any community between the network terms. In the elections2 though, it seems that there are three cohesive blocks for the terms. They still exist some other cliques, which they explored only in the appendix.

The last part of this thesis will be the two-mode network, which consisting of two types of vertices: tweets and terms. At first, we create a graph g directly from termDocMatrix. After that, dissimilar colors and sizes are specified to term vertices and tweet vertices. We also put the width and color of edges.

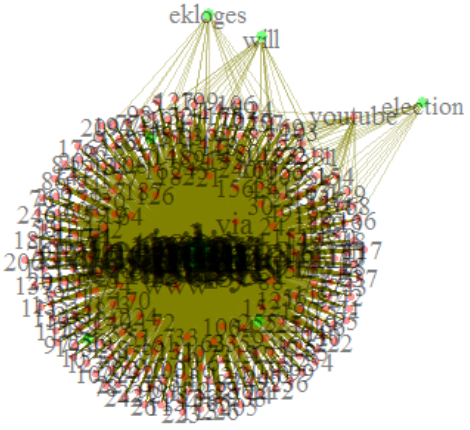


Figure 4.76: tnet1

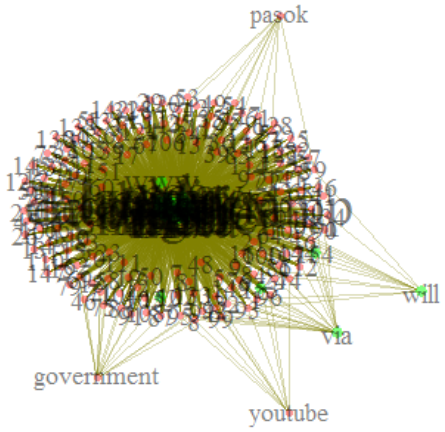


Figure 4.77: tnet2

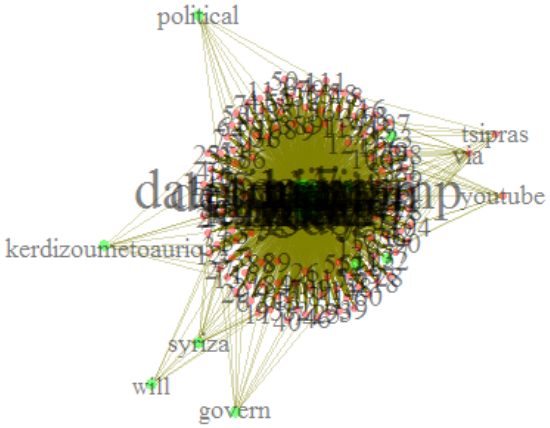


Figure 4.78: tnet3

We can perceive that the most tweets in the election dataset are around 4 centers, “ekloges”, “will”, “youtube” and “election”. There are many other centers in this graph, but because the graph is too overcrowded we can not detect those terms and we only say about what we see. In the referendum dataset are around 5 centers, “pasok”, “will”, “via” and “government”, and for the elections2 dataset are around 8 centers, “political”, “will”, “kerdizoumetoaurio”, “govern”, “syriza”, “tsipras”, “via”, “youtube”.

As for a general result we can say that first we do a plot of networks for the terms we have in our data with the help of the graphs. With it we see the connection between these terms and how close is one to the other term. Lastly we have a look at a two-mode network graph, which give to us how the terms and tweets connected to each other and in what extent achieve that.

Chapter 5

Conclusion

We gathered data from Twitter for the Greek Elections of 2015 and the referendum of the same year. Data contained one year of tweets for the first election, almost six months of tweets for the referendum and approximately 2 and a half months for the second election. The data were gathered through an appropriate python script. We used a number of statistical methods to explore and investigate these data, such as Data Exploration and Visualization techniques, Regression analysis, Clustering, Time Series analysis, Text Mining and Social Network Analysis.
We mostly confirmed the literature in the limited predictive ability of such data for accurate forecasting of election results but we gathered considerable expertise in a number of contemporary data science methods. Broadly, the techniques of Data Exploration and Visualization, Regression and Clustering gave reasonable results. The time series methods gave moderate results while Text Mining and Social Network Analysis seem to need more elaborate approaches. It appears that when we investigate a network, the most crucial factor relates to the people following this party. We also noticed that a number of followers are not particularly satisfied with New Democracy and Syriza parties, but it seems that the public has even less trust in other parties, resulting in this bipartisation by the ruling parties of power.
In conclusion, more work is required in order to improve the predictive ability of data from the Social Media. The problem of predicting election results with Twitter is largely considered to be a hard one and more detailed data, including on potential confounders, would be a useful addition in tackling this problem.

Appendix A

R Script

```
#chapter 1,2
#Read file
#install.packages("rJava")
#install.packages("xlsxjars")
#install.packages("xlsx")
library(rJava)
library(xlsxjars)
library(xlsx)
dev.off()
#me book1.xlsx
data <-read.xlsx(file.choose(),sheetIndex = 1,
encoding = "UTF-8",colIndex = c(1:10))

#chapter3
#have a look at the data
#the first 6 obs
head(data)
#returns the dimensionality of data
dim(data)
#The name of variables
names(data)
#return the structure and attributes of data
str(data)
attributes(data)
#first 5 rows of data
```

```
data[1:5,]
#explore individual variables
#install.packages("stargazer")
library(stargazer)
summary(data[,3:4])
stargazer(data[,3:4])
#explore data$retweets variable,numeric variable
quantile(data$retweets)
quantile(data$favorites)
#variance
var(data$retweets)
var(data$favorites)
#histogram
hist(data$retweets)
hist(data$favorites)
#plot
plot(density(data$retweets))
plot(density(data$favorites))

#date variable
summary.Date(data$date)
mdate <- mean(data$date)
hist(data$date,breaks ='months')
abline(v=mdate,col='red')


#factor variable
table(data$username)
#tha kratisw to piechart
pie(table(data$username))
#barplot(table(data$username))

#explore multiple variables
cov(data$retweets,data$favorites)
#positive relation,0.73
cor(data$retweets,data$favorites)

#compute the stats of retweets of every username
```

```
with aggregate()
aggregate(retweets~username,summary,data=data)
aggregate(favorites~username,summary,data=data)
#box-and-whisker plot, to show the median,
first and third quartile
of a distribution
data_sub <- subset(data,data$retweets<=1000)
datauser <- subset(data,data$username=='atsipras')
boxplot(retweets~username,data=data,xlab='username',
ylab='retweets')
boxplot(retweets~username,data=data_sub,xlab='username',
ylab='retweets')
#data_sub1 <- subset(data,data$favorites<=600)
boxplot(favorites~username,data=data,xlab='username'
,ylab='favorites')
boxplot(favorites~username,data=data_sub,xlab='username'
,ylab='favorites')
#scatterplot
with(data,plot(retweets,favorites,col=username
,pch=as.numeric(username)))
with(data_sub,plot(retweets,favorites,col=username
,pch=as.numeric(username)))

#add a small amount of noise to the data
plot(jitter(data$retweets),jitter(data$favorites))
plot(jitter(data_sub$retweets), jitter(data_sub$favorites))

#A smooth scatter plot
smoothScatter(data$retweets,data$favorites)
smoothScatter(data_sub$retweets,data_sub$favorites)


#A 3D scatter plot
library(scatterplot3d)
#scatterplot3d(data$retweets,data$favorites)
#scatterplot3d(data_sub$retweets,data_sub$favorites)

#interactive 3D scatter plot
```

```
library(rgl)
plot3d(data$retweets,data$favorites)
plot3d(data_sub$retweets,data_sub$favorites)



library(ggplot2)
qplot(retweets,favorites,data=data,facets = username ~.)
qplot(retweets,favorites,data=data_sub,facets = username ~.)



#chapter5
#linear regression is to predict response with a
linear function
of predictors as follows: y = c0 + c1x1 + c2x2 + ... + ckxk;
#where x1; x2; ... ; xk are predictors and y is the
response to predict

#Then a linear regression model is built with function lm()

fit <- lm(retweets~username+date+favorites,data = data)
summary(fit)
#fit$coefficients
#The differences between observed values and fitted values
#residuals(fit)
#par(mfrow = c(2, 2))
plot(fit)
#par(mfrow = c(1, 1))
#p3d1 <- plot3d(data_sub$date, data_sub$username,
data_sub$retweets,highlight.3d=T, type="h", lab=c(2,3))
#s3d1 <- scatterplot3d(data_sub$date, data_sub$username
,data_sub$retweets, highlight.3d=T, type="h", lab=c(2,3))
linear.pred <- predict(fit, type = 'response',data=data)
plot(data$retweets, linear.pred)
abline(0,1,col='red')
```

```
#plot for lm for better visualization
d1.plot <- ggplot(data = data,
                  aes(x = data$retweets,
                      y = linear.pred,
                      color = linear.pred)) +
  geom_point()+geom_abline(intercept =0,slope =1,col='red')
print(d1.plot)



#similar with favorites
#doesn't work well because i have factors with more than
2 levels and essentially this will not have any meaning
in my analysis.

#so i will do the analysis with logistic regression
#Logistic regression is used to predict the probability
of occurrence of an event by fitting data to a
#logistic curve. A logistic regression model is built as the
following equation:
#logit(y) = c0 + c1x1 + c2x2 + ... + ckxk;
#where x1; x2; .. ; xk are predictors, y is a
response to predict,
and logit(y) = ln(y/1-y).


#first way
library(nnet)
data$out <- relevel(data$username, ref = 'atsipras')
mymodel <- multinom(out~retweets+favorites+date,data = data)
summary(mymodel)
pred <- predict(mymodel,data,type = 'probs')
logLik(mymodel)
#prediction,type indicates the type of prediction required.
The default is on the scale of the linear predictors,
and the alternative
"response" is on the scale of the response variable
library(reshape2)
```

```
m.pred <- melt(pred)
plot(m.pred$Var2, m.pred$value, col='lightgray')
abline(h=mean(m.pred$value),col='red')
means <- tapply(m.pred$value,m.pred$Var2,mean)
means_round <- round(means, digits = 3)
points(means_round,col='red',pch=18)
text(means_round+0.02,labels = means_round)

fun_mean <- function(x){
  return(data.frame(y=round(mean(x),digits = 3),
  label=round(mean(x,na.rm=T),digits = 3)))}

d.plot <- ggplot(data = m.pred,
                 aes(x = m.pred$Var2,
                     y = m.pred$value,
                     color = m.pred$value)) +
  stat_boxplot(geom = "errorbar", width = 0.5)+
  geom_boxplot(fill='lightgray')+
  geom_hline(yintercept = mean(m.pred$value),col='red')
  + stat_summary(fun.y=mean, colour='darkred'
  ,geom="point", size=3)
  +stat_summary(fun.data = fun_mean, geom="text", vjust=-0.7)
print(d.plot)


#model fit to observations
require(ade4)
fits <- data.frame(fitted(mymodel))
str(fits)
pts <- triangle.plot(fits[,c(1:3)], scale = F,
show.position = F)
points(pts, col=c("green","magenta","red")
[as.numeric(data_sub$out)])
levels(data_sub$out)[1:3]


#second way, without date variable
library(mlogit)
```

```
data1 <- mlogit.data(data_sub, choice = 'username',
shape = 'wide'
,drop.index = TRUE)
result <- mlogit(username~0|retweets+favorites,data = data1)
summary(result)



#chapter6
#This chapter presents examples of various clustering
techniques in R, including k-means clustering,
#k-medoids clustering, hierarchical clustering
and density-based
clustering
#The k-Means Clustering
#split your data into 2 subsets for better results
set.seed(1234)
ind <- sample(2, nrow(data), replace=TRUE,
prob=c(0.95, 0.05))
trainData <- data[ind==1,]
testData <- data[ind==2,]

(kmeans.result <- kmeans(testData[,3:4],2))
table(testData$username, kmeans.result$cluster)
plot(testData[,3:4][c("retweets", "favorites")],
col = kmeans.result$cluster)
#plot cluster centers
points(kmeans.result$centers[,c("retweets", "favorites")],
col = 1:3,pch = 8, cex=2)

#from the real data except the 2 largest variables
#plot(data_sub[,3:4][c("retweets", "favorites")],
col = kmeans.result$cluster)
#plot cluster centers
#points(kmeans.result$centers[,c("retweets", "favorites")],
col = 1:3,pch = 8, cex=2)

#The k-Medoids Clustering
```

```
#it is represented with the object closest to
#the center of the cluster in the k-medoids clustering

library(fpc)
library(cluster)
library(vegan)
library(HSAUR)
pamk.result <- pamk(testData[,3:4])
# number of clusters
pamk.result$nc
# check clustering against actual species
dev.off()
table(pamk.result$pamobject$clustering, testData$username)
layout(matrix(c(1,2),1,2)) # 2 graphs per page
plot(pamk.result$pamobject)
layout(matrix(1)) # change back to one graph per page

#Next, we try pam() with k = 3 or k=4.
pam.result <- pam(testData[,3:4], 4)
table(pam.result$clustering, testData$username)
layout(matrix(c(1,2),1,2)) # 2 graphs per page
plot(pam.result)
layout(matrix(1)) # change back to one graph per page


#hierarchical clustering to the data

set.seed(5467)
data1=data
ind1 <- sample(2, nrow(data1), replace=TRUE,
prob=c(0.99, 0.01))
trainData1 <- data1[ind1==1,]
testData1 <- data1[ind1==2,]

hc <- hclust(dist(testData1[,3:4]), method="ave")
plot(hc, hang = -1, labels=testData1$username)
# cut tree into 3 clusters
rect.hclust(hc, k=3, border = 2:4)
```

```
groups <- cutree(hc, k=3)

#Density-based Clustering
#provides a densitybased clustering for numeric data
ds <- dbscan(data[,3:4], eps=10, MinPts=2)
# compare clusters with original class labels
table(ds$cluster, data$username)
plot(ds, data[,3:4])

#in the above method we make it with real data.Let's see
#it with the testData

ds1 <- dbscan(testData[,3:4], eps=10, MinPts=2)
# compare clusters with original class labels
table(ds1$cluster, testData$username)
plot(ds1, testData[,3:4])
#the data are projected to distinguish classes
plotcluster(testData[,3:4], ds1$cluster)

#The clustering model can be used to label new data,
based on the similarity between new
#data and the clusters. The following example draws
a sample of
10 objects from iris and adds
#small noises to them to make a new dataset for labeling.
The random noises are generated with
#a uniform distribution using function runif().
newData <- testData1[,3:4]
newData <- newData + matrix(runif(42*2, min=0, max=0.2),
nrow=42, ncol=2)
# label new data
myPred <- predict(ds1, testData[,3:4], newData)
# plot result
plot(testData[c(3,4)], col=1+ds1$cluster)
points(newData, pch="*", col=1+myPred, cex=3)
# check cluster labels
table(myPred, testData1$username)
```

```
#euresi eps gia to density-cluster
library(dbscan)
dbscan::kNNdistplot(testData[,3:4], k =  2)
abline(h = 10, lty = 2,col='red')



#chapter7
#outlier detection by clustering
kmeans1.result <- kmeans(data[,3:4], centers=2)
# cluster centers
kmeans1.result$centers
# cluster IDs
kmeans1.result$cluster

# calculate distances between objects and cluster centers
centers <- kmeans1.result$centers[kmeans1.result$cluster, ]
distances <- sqrt(rowSums((data[,3:4] - centers)^2))
# pick top 5 largest distances
outliers <- order(distances, decreasing=T)[1:5]
# who are outliers
print(outliers)
print(data[,3:4][outliers,])
# plot clusters
plot(data[,3:4][,c("retweets", "favorites")], pch="o"
,col=kmeans1.result$cluster, cex=0.3)
# plot cluster centers
points(kmeans1.result$centers[,c("retweets", "favorites")]
, col=1:3,pch=8, cex=1.5)
# plot outliers
points(data[,3:4][outliers, c("retweets", "favorites")]
, pch="+", col=4, cex=1.5)



#chapter8
stockprices <- data.frame(data$date,data$retweets)
```

```
require(xts)
require(lubridate)
stockprices.ts <- xts(stockprices$data.retweets,
order.by=as.POSIXct(stockprices$data.date))
print(stockprices.ts)
str(stockprices.ts)
attributes(stockprices.ts)
plot(stockprices.ts)

#decomposition
stockprices1.ts <- ts(stockprices.ts,frequency = 12)
plot(stockprices1.ts)
#
detach("package:igraph")
f <- decompose(stockprices1.ts)
# seasonal figures
f$figure
plot(f$figure, type="b", xaxt="n", xlab="")
# get names of 12 months in English words
monthNames <- months(ISOdate(2014,1:12,1))
# label x-axis with month names
# las is set to 2 for vertical label orientation
axis(1, at=1:12, labels=monthNames, las=2)
plot(f)

#Here are a few more operations you can do
cycle(stockprices1.ts)
#This will print the cycle across months.
plot(aggregate(stockprices1.ts,FUN=mean))
#This will aggregate the cycles and display a month
on year trend
boxplot(stockprices1.ts~cycle(stockprices1.ts))
#Box plot across months will give us a sense on
seasonal effect

#forecasting
fit1 <- arima(stockprices1.ts, order=c(1,0,0),
list(order=c(2,1,0), period=12))
```

```
fore <- predict(fit1, n.ahead=24)
# error bounds at 95% confidence level
U <- fore$pred + 2*fore$se
L <- fore$pred - 2*fore$se
ts.plot(stockprices1.ts, fore$pred, U, L, col=c(1,2,4,4),
lty = c(1,1,2,2))
legend("topleft", c("Actual", "Forecast",
"Error Bounds (95% Confidence)"),col=c(1,2,4), lty=c(1,1,2))



#chapter10
#transforming text
#me book 2
data_eng <- read.xlsx(file.choose(),sheetIndex = 1,
encoding = "UTF-8",colIndex = c(1:10))
library(tm)
#make new dateset for better results
set.seed(1234)
ind_eng <- sample(2, nrow(data_eng), replace=TRUE,
prob=c(0.95, 0.05))
trainData_eng <- data_eng[ind_eng==1,]
testData_eng <- data_eng[ind_eng==2,]

# build a corpus, and specify the source to be
character vectors
myCorpus <- Corpus(VectorSource(testData_eng$text))
# convert to lower case
myCorpus <- tm_map(myCorpus, content_transformer(tolower))
# remove URLs
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
# tm v0.6
myCorpus <- tm_map(myCorpus, content_transformer(removeURL))
# remove anything other than English letters or space
removeNumPunct <- function(x)
gsub("[^[:alpha:][:space:]]*", "", x)
myCorpus <- tm_map(myCorpus,
content_transformer(removeNumPunct))
```

```
removeURL1 <- function(x) gsub("http[[:alnum:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeURL1))
# remove numbers
myCorpus <- tm_map(myCorpus, removeNumbers)
myCorpus <- tm_map(myCorpus, removeWords,
stopwords("english"))
# Remove punctuations
myCorpus <- tm_map(myCorpus, removePunctuation)
# Eliminate extra white spaces
myCorpus <- tm_map(myCorpus, stripWhitespace)


#stemming words
# keep a copy of corpus to use later as a dictionary
for stem completion
myCorpusCopy <- myCorpus
# stem words
myCorpus <- tm_map(myCorpus, stemDocument)
# inspect documents (tweets) numbered 11 to 15
inspect(myCorpus[11:15])
# The code below is used for to make text fit for paper width
   for (i in 11:15) {
     cat(paste("[[", i, "]] ", sep=""))
     writeLines(strwrap(myCorpus[[i]], width=73))
     }

# inspect the first 5 documents (tweets)
inspect(myCorpus[1:5])
# The code below is used for to make text fit for paper width
   for (i in 11:15) {
     cat(paste0("[", i, "] "))
     writeLines(strwrap(as.character(myCorpus[[i]]), 60))
     }


stemCompletion2 <- function(x, dictionary) {
   x <- unlist(strsplit(as.character(x), " "))
   # Unexpectedly, stemCompletion completes an
```

```
    empty string to
      # a word in dictionary. Remove empty string
      to avoid above issue.
      x <- x[x != ""]
      x <- stemCompletion(x, dictionary=dictionary)
      x <- paste(x, sep="", collapse=" ")
      PlainTextDocument(stripWhitespace(x))
      }

myCorpus <- lapply(myCorpus, stemCompletion2,
dictionary=myCorpusCopy)
myCorpus <- Corpus(VectorSource(myCorpus))
inspect(myCorpus[11:15])

# Text stemming (reduces words to their root form)
library("SnowballC")
tdm <- TermDocumentMatrix(myCorpus)
m <- as.matrix(tdm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 10)


#Frequent Terms and Associations
# inspect frequent words
findFreqTerms(tdm, lowfreq=10)
#finds frequent terms with frequency no less than ten.

#frequent term visually
termFrequency <- rowSums(as.matrix(tdm))
termFrequency <- subset(termFrequency, termFrequency>=10)
library(ggplot2)
df <- data.frame(term=names(termFrequency),
freq=termFrequency)
ggplot(df, aes(x=term, y=freq)) + geom_bar(stat="identity") +
   xlab("Terms") + ylab("Count") + coord_flip()
#second ggplot,shows the freq
ggplot(df, aes(x=term, y=freq, fill=freq)) +
```

```
geom_bar(stat="identity") +
  xlab("Terms") + ylab("Count") + coord_flip()
#third ggplot, shows the freq from highest to lowest
ggplot(df, aes(x=reorder(term, -freq), y=freq, fill=freq)) +
geom_bar(stat="identity") +
  xlab("Terms") + ylab("Count") +
  theme(axis.text.x=element_text(angle=45, hjust=1))


#highly associated with a word
#terms associated with "elector" (or "eklog")
with correlation
no less than 0.25, and the
#words are ordered by their correlation with
"elector" (or 'eklog")
# which words are associated with "elector"?
findAssocs(tdm, "election", 0.25)

# which words are associated with "eklog"?
findAssocs(tdm, 'ekloges', 0.25)

#word cloud
library(wordcloud)
m <- as.matrix(tdm)
# calculate the frequency of words and sort it descendingly
by frequency
wordFreq <- sort(rowSums(m), decreasing=TRUE)
# colors
pal <- brewer.pal(9, "BuGn")
pal <- pal[-(1:4)]
# word cloud
set.seed(375) # to make it reproducible
grayLevels <- gray( (wordFreq+10) / (max(wordFreq)+10) )
#first wordcloud
wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=3,
random.order=F, colors=pal, max.words=150)
#another library wordcloud2, ti sixnotita exei i kathe leji
library(wordcloud2)
```

```
v <- sort(rowSums(m), decreasing=TRUE)
d <- data.frame(word = names(v), freq=v)
wordcloud2(d, size = 0.8, shape = 'circle')


#Clustering Words
#find clusters of words with hierarchical clustering
#Sparse terms are removed, so
#that the plot of clustering will not be crowded with words

# remove sparse terms
tdm2 <- removeSparseTerms(tdm, sparse=0.95)
m2 <- as.matrix(tdm2)
# cluster terms
distMatrix <- dist(scale(m2))
fit2 <- hclust(distMatrix, method="ward.D")
plot(fit2)
# cut tree into 3 clusters
rect.hclust(fit2, k=3, border = 2:4)
(groups <- cutree(fit2, k=3))

#Clustering Tweets
#Clustering Tweets with the k-means Algorithm
#k-means clustering, which takes the values in the
matrix as numeric
#We transpose the term-document matrix to a document-term one
# transpose the matrix to cluster documents (tweets)
m3 <- t(m2)
# set a fixed random seed
set.seed(122)
# k-means clustering of tweets
k <- 10
kmeansResult <- kmeans(m3, k)
# cluster centers
round(kmeansResult$centers, digits=3)

#To make it easy to find what the clusters are about,
we then check the top three words in every cluster.
```

```
for (i in 1:k) {
   cat(paste("cluster ", i , ": ", sep=""))
   s <- sort(kmeansResult$centers[i ,], decreasing=T)
   cat(names(s)[1:3], "\n")
   #print the tweets of every cluster
   #print(rdmTweets[which(kmeansResult$cluster==i )])
}

#From the above top words and centers of clusters , we can see
that the clusters are on the same topics.

#Clustering Tweets with the k-medoids Algorithm
#k-medoids clustering with the Partitioning
Around Medoids (PAM)
algorithm , which
#uses medoids (representative objects) instead of means
to represent clusters
#It is more robust to
#noise and outliers than k-means clustering , and provides
a display
of the silhouette plot to show
#the quality of clustering

library(fpc)
# partitioning around medoids with estimation of number
of clusters
pamResult <- pamk(m3, metric="manhattan")
# number of clusters identified
(k1 <- pamResult$nc)

pamResult <- pamResult$pamobject
#print cluster medoids
   for (j in 1:k1) {
      cat(paste("cluster", j , ": "))
      cat(colnames(pamResult$medoids)
      [which(pamResult$medoids[j ,]==1)],
      "\n")
      #print tweets in cluster j
```

```
        #print(rdmTweets[pamResult$clustering==j])
    }

#clusplot
testData_eng.x<-testData_eng[,3:4]
cl3<-pam(testData_eng.x,10)$clustering
clusplot(testData_eng.x,cl3,color=F,labels=4,lines=0,
cex=.8,col.clus=1,col.p=cl3)


#different kind of plots with ggfortify
library(ggfortify)
#kmeans
autoplot(kmeans(testData_eng.x,10),data=testData_eng.x,
label=TRUE,label.size=3,frame=TRUE)
#pam
autoplot(pam(testData_eng.x,10),frame=TRUE,
frame.type='norm',label=TRUE,label.size=3)


#silhouette plot
d.dist=daisy(testData_eng.x)
res=pam(d.dist,10)#or whatever your choice of
clustering algorithm is
sil=silhouette(res$clustering,d.dist)#or use
your cluster vector
windows()#RStudio sometimes does not display
silhouette plots correctly
plot(sil)
library(factoextra)
fviz_silhouette(sil)

#sentiment analysis
library(syuzhet)
library(lubridate)
library(ggplot2)
library(scales)
library(reshape2)
```

```
library(dplyr)
tweets <-iconv(testData_eng$text,from = "UTF-8",to="UTF-8")
#obtain sentiment scores
s <- get_nrc_sentiment(tweets)
head(s)

ts<-data.frame(t(s))

ts_new <- data.frame(rowSums(ts[2:257]))
#The function rowSums computes column sums across
rows for each level
of a grouping variable.

#Transformation and cleaning
names(ts_new)[1] <- "count"
ts_new <- cbind("sentiment" = rownames(ts_new), ts_new)
rownames(ts_new) <- NULL

#better Visualisation
library("ggplot2")
qplot(sentiment, data=ts_new, weight=count, geom="bar",
fill=sentiment)
+ggtitle("sentiment analysis")
#more explanatory
qplot(sentiment, data=ts_new, weight=count, geom="bar",
fill=count)
+ggtitle("sentiment analysis")


#chapter11
termDocMatrix <- as.matrix(tdm2)
# inspect part of the matrix
termDocMatrix[5:10,1:20]
# change it to a Boolean matrix
termDocMatrix[termDocMatrix>=1] <- 1
# transform into a term-term adjacency matrix
termMatrix <- termDocMatrix %*% t(termDocMatrix)
# inspect terms numbered 5 to 10
```

```
termMatrix[5:10,5:10]

#In the above code, %*% is an operator for the product
of two matrices,
and t() transposes a
#matrix. Now we have built a term-term adjacency matrix,
where the rows and columns represent
#terms, and every entry is the number of concurrences
of two terms.
Next we can build a graph
#with graph.adjacency() from package igraph.

library(igraph)
# build a graph from the above matrix
g <- graph.adjacency(termMatrix, weighted=T,
mode="undirected")
# remove loops
g <- simplify(g)
# set labels and degrees of vertices
V(g)$label <- V(g)$name
V(g)$degree <- degree(g)
#After that, we plot the network with
layout.fruchterman.reingold
# set seed to make the layout reproducible
set.seed(3952)
layout1 <- layout.fruchterman.reingold(g)
plot(g, layout=layout1)

#In the above code, the layout is kept as layout1,
so that we can plot
the graph in the same
#layout later.

#A different layout can be generated with the first
line of code below.
The second line produces
#an interactive plot, which allows us to manually
rearrange the layout
```

```
plot(g, layout=layout.kamada.kawai)
tkplot(g, layout=layout.kamada.kawai)
#A 3D plot of the graph can be shown rglplot()
library(rgl)
coords <- layout.kamada.kawai(g, dim=3)
open3d()
rglplot(g, vertex.size=3, vertex.label=NA,
edge.arrow.size=0.6, layout=coords)


#Next, we set the label size of vertices based
on their degrees,
to make important terms stand
#out.
V(g)$label.cex <- 2.2 * V(g)$degree / max(V(g)$degree)+ .2
V(g)$label.color <- rgb(0, 0, .2, .8)
V(g)$frame.color <- NA
egam <- (log(E(g)$weight)+.4) / max(log(E(g)$weight)+.4)
E(g)$color <- rgb(.5, .5, 0, egam)
E(g)$width <- egam
# plot the graph in layout1
plot(g, layout=layout1)
#Next, we try to detection communities from the graph.
#cohesive blocks
blocks <- cohesive.blocks(g)
blocks
#windows()
plot(blocks, g, vertex.size=.3, vertex.label.cex=1.5,
edge.color=rgb(.4,.4,0,.3))
#maximal cliques
cl <- maximal.cliques(g)
length(cl)
colbar <- rainbow(length(cl) + 1)
for (i in 1:length(cl)) {
  V(g)[cl[[i]]]$color <- colbar[i+1]
}
plot(g, mark.groups=cl, vertex.size=.3, vertex.label.cex=1.5,
edge.color=rgb(.4,.4,0,.3))
```

```
#largest cliques
cl <- largest.cliques(g)
length(cl)
colbar <- rainbow(length(cl) + 1)
for (i in 1:length(cl)) {
  V(g)[cl[[i]]]$color <- colbar[i+1]
}
plot(g, mark.groups=cl, vertex.size=.3, vertex.label.cex=1.5,
edge.color=rgb(.4,.4,0,.3))


#11.2
# remove common terms to simplify graph and find
# relationships between tweets beyond keywords
tdm.m <- m2
idx <- which(dimnames(tdm.m)$Terms %in%
c("117", "datetimestamp","isdst"))
M <- tdm.m[-idx,]
# build tweet-tweet adjacency matrix
tdm.tweetm <- t(M) %*% M
tdm.tweetm[5:10,5:10]
# configure plot
library(igraph)
tdm.g <- graph.adjacency(tdm.tweetm, weighted=TRUE,
mode="undirected")
V(tdm.g)$degree <- degree(tdm.g)
tdm.g <- simplify(tdm.g)

# set labels of vertices to tweet IDs
V(tdm.g)$label <- V(tdm.g)$name
V(tdm.g)$label.cex <- 1
V(tdm.g)$color <- rgb(.4, 0, 0, .7)
V(tdm.g)$size <- 2
V(tdm.g)$frame.color <- NA

# barplot of connections
barplot(table(V(tdm.g)$degree),
```

```
main="Number of Adjacent Edges")

# set vertex colors based on degree
idx <- V(tdm.g)$degree == 258
V(tdm.g)$label.color[idx] <- rgb(0,0,.3,.7)
# convert tweets to data.frame
tdm.df <- do.call("rbind",
lapply(testData_eng$text, as.data.frame))
# set labels to the IDs and the first 20 characters of tweets
V(tdm.g)$label[idx] <- paste(V(tdm.g)$name[idx],
                             substr(tdm.df$X[2:21], 1, 20),
                             sep=": ")
egam <- (log(E(tdm.g)$weight)+.2) / max(log(E(tdm.g)$weight)
+.2)
E(tdm.g)$color <- rgb(.5, .5, 0, egam)
E(tdm.g)$width <- egam
set.seed(3152)
layout2 <- layout.fruchterman.reingold(tdm.g)
plot(tdm.g, layout2)

# delete vertices in crescent with no degrees
# remove from graph using delete.vertices()
tdm.g2 <- delete.vertices(tdm.g,
                          V(tdm.g)[degree(tdm.g)==0])
plot(tdm.g2, layout=layout.fruchterman.reingold)

# remove edges with low degrees
tdm.g3 <- delete.edges(tdm.g,
                       E(tdm.g)[E(tdm.g)$weights <= 1])
tdm.g3 <- delete.vertices(tdm.g3,
                          V(tdm.g3)[degree(tdm.g3)==0])
plot(tdm.g3, layout=layout.fruchterman.reingold)
tdm.df$X[c(7,12,6,9,8,3,4)]


#11.3
# create a graph
g <- graph.incidence(termDocMatrix, mode=c("all"))
```

```
# get index for term vertices and tweet vertices
nTerms <- nrow(M)
nDocs <- ncol(M)
idx.terms <- 1:length(nTerms)
idx.docs <- (nTerms+1):(nTerms+nDocs)
# set colors and sizes for vertices
V(g)$degree <- degree(g)
V(g)$color[idx.terms] <- rgb(0, 1, 0, .5)
V(g)$size[idx.terms] <- 6
V(g)$color[idx.docs] <- rgb(1, 0, 0, .4)
V(g)$size[idx.docs] <- 4
V(g)$frame.color <- NA
# set vertex labels and their colors and sizes
V(g)$label <- V(g)$name
V(g)$label.color <- rgb(0, 0, 0, 0.5)
V(g)$label.cex <- 1.4*V(g)$degree/max(V(g)$degree) + 1
# set edge width and color
E(g)$width <- .3
E(g)$color <- rgb(.5, .5, 0, .3)
set.seed(958)
plot(g, layout=layout.fruchterman.reingold)


#Next, we remove "ekloges", "117" and "election" to show
the relationship
between tweets with other
#words. Isolated vertices are also deleted from graph.

idx <- which(V(g)$name %in% c("ekloges", "117", "election"))
g2 <- delete.vertices(g, V(g)[idx-1])
g2 <- delete.vertices(g2, V(g2)[degree(g2)==0])
set.seed(209)
plot(g2, layout=layout.fruchterman.reingold)
```

For the Python Script we will not represent it in this thesis because it is not just a script but a lot and it will take up a space in our thesis unfair.

Bibliography

[1] Maidou. Η χρήση του Twitter ως μέσο συμμετοχής στην πολιτική. 2016.

[2] Tsintzou. Election Analysis and Prediction of Election Results with Twitter. 2016.

[3] wikipedia. Χρυσή Αυγή. URL: https://el.wikipedia.org/wiki/%CE%A7%CF%81%CF%85%CF%83%CE%AE_%CE%91%CF%85%CE%B3%CE%AE.

[4] wikipedia. Δημοκρατική Συμπαράταξη. URL: https://el.wikipedia.org/wiki/%CE%94%CE%B7%CE%BC%CE%BF%CE%BA%CF%81%CE%B1%CF%84%CE%B9%CE%BA%CE%AE_%CE%A3%CF%85%CE%BC%CF%80%CE%B1%CF%81%CE%AC%CF%84%CE%B1%CE%BE%CE%B7.

[5] wikipedia. Ένωση Κεντρώων. URL: https://el.wikipedia.org/wiki/%CE%88%CE%BD%CF%89%CF%83%CE%B7_%CE%9A%CE%B5%CE%BD%CF%84%CF%81%CF%8E%CF%89%CE%BD.

[6] wikipedia. Δημοψήφισμα. URL: https://el.wikipedia.org/wiki/%CE%95%CE%BB%CE%BB%CE%B7%CE%BD%CE%B9%CE%BA%CF%8C_%CE%B4%CE%B7%CE%BC%CE%BF%CF%88%CE%AE%CF%86%CE%B9%CF%83%CE%BC%CE%B1_%CF%84%CE%BF%CF%85_2015.

[7] Brett Lantz. Machine Learning with R. 2015.

[8] D G Rossiter. "Analyzing land cover change with logistic regression in R". In: (2016). URL: http://www.css.cornell.edu/faculty/dgr2/teach/R/R_lcc.pdf.

[9] sites.google.com. Density based clustering algorithm. URL: https://sites.google.com/site/dataclusteringalgorithms/density-based-clustering-algorithm.

[10] www.springboard.com. text-mining-in-r. URL: https://www.springboard.com/blog/text-mining-in-r/.

[11] Srivasta. "Data Mining based social network analysis". In: (2008). URL: https://www.siam.org/meetings/sdm08/TS1.pdf.

[12] Yanchang Zhao. R and Data Mining: Examples and Case Studies. 2015.