LUT
Lappeenranta
University of Technology

LUT School of Business and Management

B.Sc. programme in Business Administration

Financial Management

Forecasting univariate time series - comparison of statistical models and software resources available to undergraduate students

BACHELOR'S THESIS

14.5.2017
Written by: Sami Kohvakka
Supervisor: Jan Stoklasa

TIIVISTELMÄ

Tekijä:	Sami Kohvakka
Tutkielman nimi:	Kandiopiskelijoille tarjolla olevien tilastollisten mallien ja ohjelmistojen vertailu yhden muuttujan aikasarjojen ennustamisessa
Akateeminen yksikkö:	School of Business and Management
Koulutusohjelma:	Kauppatieteiden kandidaatti / Talousjohtaminen
Ohjaaja:	Jan Stoklasa
Hakusanat	Aikasarjat, Ennustaminen, ARIMA, MATLAB, Python, R, SAS

Tässä kandidaatintyössä vertailtiin neljää tilastollista ohjelmaa suorituskyvyn ja käytettävyyden osalta. Vertailu keskittyi ohjelmistojen käytettävyyteen yhden muuttujan aikasarjojen ennustamisessa eikä ota huomioon hyödyllisyyttä muilla ekonometrian, tilastotieteen tai soveltavan matematiikan aloilla. SAS on yleisesti käytetty yhteiskuntatieteissä sen tarjoamien ANOVA mahdollisuuksien vuoksi. MATLAB on insinööritieteissä yleisesti käytössä matriisilaskennan ja koneoppimisen sovellutuksien ansiosta. Python ei varsinaisesti ole tilastollinen ohjelma, vaan pikemminkin ohjelmointikieli. Monipuolisuuden ja yksinkertaisuuden ansiosta Python on usein ensimmäinen ohjelmointikieli, jonka tietotekniikan opiskelijat oppivat. Ennustamisen kannalta R kuitenkin osoittautui parhaaksi vaihtoehdoksi useimmille kandidaattiopiskelijoille.

Tässä työssä vertailtiin myös eri ennustusmenetelmien tarkkuutta kahdessa aikasarjassa. Ensimmäinen aikasarjoista edustaa henkilöautojen kysyntää Yhdysvalloissa vuosina 1980-2016 ja siinä voidaan havaita kausittainen vaihtelu ja trendi. Toinen aikasarjoista on S&P 500 indeksi, joka edustaa osakemarkkinoita. Tässä sarjassa ei ole selvää kuviota vaan se vaikuttaa noudattavan satunnaiskävelyä. Ennustamiseen käytettiin yksinkertaisten menetelmien lisäksi ARIMA-malleja. Tukivektorikoneet, neuroverkot ja GARCH-mallit jätettiin vertailun ulkopuolelle, koska ne eivät kuulu kandidaatinopintoihin ekonometriassa vaan maisterikurssille syvennetyssä liiketoiminta-analytiikassa.

Yleisesti ARIMA-mallit suoriutuivat hyvin ja onnistuivat ennustamaan tarkasti kausittaisen vaihtelun kysynnässä. ARIMA(0,1,0) malli poimi lineaarisen trendin S&P 500 indeksistä, mutta ei kyennyt ennustamaan äkkinäistä pudotusta indeksin arvossa. Tutkielman löydökset ovat linjassa aikaisemman tutkimuksen kanssa ja väittävät, että: a) Menetelmän tarkkuus riippuu käytetystä virheen mittaamisen menetelmästä. b) Kandidaatin opinnot mahdollistavat kausittaisen kysynnän ennustamisen. c) Monimutkaisempia menetelmiä tarvitaan osakemarkkinoiden ennustamiseen. d) Yksinkertaiset menetelmät ovat yllättävän hyviä. e) Mallin määritys ja oppimisdatan valinta voi vaikuttaa ennustustarkkuuteen. f) Menetelmien yhdistäminen voi parantaa ennustustarkkuutta.

ABSTRACT

Author:	Sami Kohvakka
Title:	Forecasting univariate time series – comparison of statistical methods and software resources available to undergraduate students
School:	School of Business and Management
Degree programme:	B.Sc. in Business Administration / Financial Management
Supervisor:	Jan Stoklasa
Keywords:	Time series, Forecasting, ARMA, MATLAB, Python, R, SAS

Four statistical software were compared in this thesis in terms of performance and usability. Comparison focuses on forecasting capabilities of the selected software in univariate time series and does not consider usefulness of the software in other fields of econometrics, statistical analysis or applied mathematics. SAS is widely used in social sciences due to ANOVA capabilities and MATLAB is popular in engineering sciences because of capabilities in matrix calculus and machine learning. Python is technically not a software, because it is more of a generic programming language. Because of the ease of learning and wide applicability, it is often the first programming language modern IT students take. In terms of forecasting, R was selected the best choice for most undergraduate students.

This thesis also compared performance of different forecasting methods in two econometric time series. First of the series represents demand of new passenger cars in the United States from 1980 to 2016. It has a seasonal pattern and a time trend. Second of the series is S&P 500 index, which represents stock markets. Series does not have a clear pattern and it seems to follow random walk. ARIMA models were applied in addition to simple methods. Support vector machines, neural networks and GARCH models were excluded, because they are part of master's course in advanced business analytics, not undergraduate education in econometrics.

In general, ARIMA models performed well and could accurately capture seasonality in demand data. ARIMA(0,1,0) model was able to capture linear trend in S&P 500 but was not able to predict sudden drop in the index. Key findings are in line with previous research and suggest that a) Performance of a forecasting method depends on measure of error used. b) Undergraduate education provides knowledge required to forecast seasonal demand. c) More complex methods are required to forecast stock markets. d) Simple methods perform surprisingly well. e) Model identifying and selection of learning set might influence forecasting accuracy. f) Combining forecasts might improve forecasting accuracy.

Table of contents

List of symbols and abbreviations

ARMA	Autoregressive moving average
GQ	Goldfeld-Quandt
SSE	Sum of squared errors
SES	Simple exponential smoothing
OLS	Ordinary least squares
u	residual
LM	Lagrange Multiplier
R^2	R squared
ε	white noise error term
h	forecasting horizon
α	Alpha (coefficient)
β	Beta (coefficient)
γ	Gamma (coefficient)
θ	Theta (coefficient)
ρ	Rho (coefficient)
τ	Tau (distribution)
X'	transpose of a matrix X
X^{-1}	inverse of a matrix X
χ^2	Chi-squared
$\bar{y}$	mean value of *y*
$\hat{y}$	estimated value of y

1 Introduction

Univariate time series are important measures of development in both micro- and macroeconomics. Stock prices, returns, interest rates and most macroeconomic indicators are all examples of univariate time series representing development of some measurable phenomena. Reliably forecasting econometric time series and taking use of the large amounts of data gathered by companies and international organizations might provide competitive advantage in decision making. Univariate models can be used for example in forecasting demand, stock returns and inflation rates. Aboagye-Sarfo et al. (2015) used univariate ARMA models to successfully forecast emergency department demand in Western Australia. Previous research in forecasting univariate time series has been done e.g. by (Lütkepohl 2015), who used AR-models to predict monthly development of stock indices quite successfully over a period of 6 months and noted that using log returns improves forecasting accuracy. Thomopoulos (2015) suggested MA-models for demand forecasting and Dunis, Laws and Karathanassopoulos (2012) used an applied ARMA model as an input in neural network application to forecast Greek stock markets.

Research by Makridakis & Hibon (2000) suggests that no forecasting method is superior to others in all possible time series and that the superiority depends on the measures of error used. The success of experts of ForecastPro in M3 competition provides evidence that the knowledge of statistics and a selective approach to model fitting and forecasting yields better forecasting results than blindly adopting the same method for all series.

1.1 Research problems, objectives and limitations

Besides M-competitions in 1982, 1993 and 2000, rather little research has been done in comparing the accuracy of different forecasting methods and different statistical software. The research goal in this thesis is to find out whether more advanced forecasting methods perform better than simpler methods in different time series. This leads to the main research question:

Do more advanced (ARIMA) forecasting methods improve the accuracy of forecasts over simple methods in univariate homoscedastic time series?

Forecasting econometric time series has become easier due to the development of personal computers. There are comprehensive guides with cross references to different statistical software, such as (Muenchen 2011), but very little research has been done comparing the

software in terms of usability and performance. Therefore, a secondary research question is presented as:

What kind of software resources are available to undergraduate students for econometrics?

Trying to answer these two questions leads to sub questions:

Do these econometric software packages differ in usability and performance?

Are there univariate homoscedastic econometric time series in which ARIMA models forecast better than in others?

Is it possible to improve forecasting accuracy by combining different forecasts?

Assumptions in this thesis are that more advanced forecasting methods are expected to perform better than naive forecasting methods. SAS enterprise guide is expected to be the easiest to learn, however Python and MATLAB will likely be quicker in calculations and allow more customization. R has gained high popularity in recent years among the analysts in Wall Street as well as in Google and several research universities (Vance 2009). Therefore, R is expected to be a suitable free substitute for proprietary SAS enterprise guide package.

Used forecasting methods were limited to ones generally available to undergraduate business students. Therefore, support vector machines, neural networks, fuzzy logic and GARCH models were excluded, because they are part of master's education in business analytics or machine learning and applied mathematics at LUT. Of the available methods, ARIMA models are the most complex. Software selection was limited to those available to LUT students free of charge.

1.2 Structure of the study

This thesis begins with a short description of used software, after which given software are compared in terms of usability and performance. Then key concepts and measures of error are described, followed by a description of simple methods used. After describing the simple methods, theory behind ARIMA models is described. This is followed by the empirical part of this thesis, where described methods are applied on two separate time series. Before conclusion, ensembles are considered as a method of improving forecasting accuracy. In the final chapter, conclusions of the study are presented.

1.3 Setting up the environment

1.3.1 Hardware

All estimates and forecasts were run on a single computer with an Intel i7-4790 CPU clocked at 3.6 GHz. Operating system was a 64-bit Windows 10 Pro installed on a SSD. All tested software were installed on the same internal SSD disk. Size of the random-access memory (RAM) was 16GB, type DDR3 clocked at 1600MHz and GPU used was Asus Strix-GeForce 970 with an internal memory of 4GB. Running the software was smooth and no tested software crashed under this system so that it was not able to recover. Only time SAS crashed and recovered was when trying to print out inverse of a 1500 by 1500 matrix. R console crashed and recovered when inversing 5000 by 5000 matrix.

1.3.2 Software

A list of commonly used econometric software packages was first found on Wikipedia (2017). By going through the list several commonly used packages, such as SPlus and SPSS were rejected due to lack of unit root tests. Although showing some potential, Eviews and Stata were rejected at a later phase for not being available for undergraduate students at LUT for free of charge. By looking at the functions and availability, a selection of four econometric packages emerged. These packages include two proprietary packages, SAS and MATLAB, which are available free of charge for students use through university license and two open source packages, Python and R, which are free for all. According to TIOBE index, selected four packages are also the most popular (TIOBE 2017).

SAS

SAS, which once stood for “Statistical Analysis System”, is perhaps the great grandfather of all statistical software packages. Started as a research project at North Carolina State University in 1966, SAS incorporated in 1976 and has been trying to help companies make better decisions through data analysis ever since. (SAS Institute 2017)

SAS 9.4 was selected as a benchmark, because it is used to introduce undergraduate business students to statistical computing and econometrics at LUT. Undergraduate business students can install SAS package on their computers for free with a license provided by the university. Unlike other compared software packages, SAS installation provided by the university is only available to Windows. Each year SAS license granted by the university is valid until the end of May.

SAS is mostly written in C and can be used by writing scripts in SAS programming language or by navigating the menus in graphical user interface called SAS Enterprise Guide. However, SAS does not allow much customization and creating user-built modules requires knowledge of C and a developer's kit for SAS. Therefore, one is practically limited to the prebuilt functionalities of the software. Although SAS has a comprehensive user guide available online, it has been criticized for showing the most powerful thing a given procedure can do rather than showing how to do something simple which 80% of its end-users will be doing with the procedure (Acock 2005).

Importing data to SAS is possible from a lot of filetypes, such as .xlxs, .txt, .csv and many others. However, SAS is very strict about data requirements and unlike MATLAB, does not allow previewing the data before importing. When importing files to SAS the first row of the file must contain variable names and the file may not contain anything else than these names and the actual data. SAS package includes a wide range of modules to choose from, each one usually priced separately. Getting SAS without the installation package provided by LUT is rather difficult process, as the company customizes and prices the installation separately for each customer. Prices may vary a lot but estimates found in internet for a single user license of just the core functionalities start form $5000 plus additional $2000 each year.

MATLAB

MATLAB, an acronym for matrix laboratory is built for matrix algebra and calculus. Practically all data imported to MATLAB is presented in matrix form. MATLAB has a useful tool for importing data through graphical UI. It lets user to select appropriate rows and columns from a larger datafile, preview the selection and choose the format (datetime, text or number) for each imported variable.

MATLAB was selected, because it is used in the Master's programme of Finance and Business Analytics to introduce students to advanced econometrics and applied data analysis. MATLAB is also used in engineering sciences at LUT and all students can download it for free with a license provided by the university. License grants free tech support for students and is valid for a year, starting from the day of the installation.

It should be noted that the basic MATLAB installation does not include functions for estimating ARIMA models. ARIMA models are part of *Econometrics* package, which is sold separately. Luckily this package is included in the license provided by LUT. For commercial users, standard single user MATLAB license costs 2000 € and Econometrics Toolbox additional 1150

€, but it requires Optimization Toolbox and Statistics and Machine Learning Toolbox to run. These additional packages cost 1150 € and 1000 € respectively totaling to 5300 € for functionalities required to estimate ARIMA models. For non-LUT students who study at a degree granting research university and are thus eligible for student license, this package would total to 95 €. (MathWorks 2017)

R

R was selected, because during past few years it has gained attention in business journals and surpassed SAS, MATLAB, Stata, SPSS and other proprietary packages in popularity. By popularity higher ranking in TIOBE Programming Community Index is meant. TIOBE index measures proportion of articles about specific programming language in 25 different search engines, such as Google, Yahoo, Wikipedia, YouTube, Amazon and Baidu. Basically, the index measures the activity of developer communities and students learning the language. (TIOBE 2017).

R is a programming language and an open source integrated development environment for statistical computing available online for free for both commercial and non-commercial use. Several proprietary add-ons exist, mainly to help inexperienced corporate users by offering expanded graphical user interface, tech support and cloud services.

R differs quite a bit from SAS and MATLAB. R is modular and it treats user created content, such as formulas and scripts, equal to prebuilt modules. It is possible to modify the source code of R and compile own version of it, but this requires knowledge way beyond the scope of a regular user. Being open source, R allows total customization of the program for serious computer wizards. Novice users should note that R has several graphical user interfaces available, and it is possible and relatively easy to download all of them and choose which one to launch at the startup. (Muenchen 2011) Personally, I found RStudio quite useful but felt more comfortable using R by scripting than going through menus in different GUIs.

Some older textbooks, such as Tsay (2005) may contain examples written in S or its commercial application S-Plus. This can be very beneficial for R users, because R is an implementation of S. Besides some minor differences, such as replacing underscore assignment operator "_" with an arrow "<- ", most of the code written in S runs unaltered in R. (R foundation 2017) R has a uniform namespace where each function must have different name. Functions are located inside *libraries*, which can be installed by typing install.packages("<name of the package>"). Before using a function library containing the function must be imported by typing library(<name of

the library>). By default, packages are installed for currently active user and installation does not require administrator privileges.

Python

Originally developed as a real programming language rather than a simplified language to use a statistical analysis software, Python is a bit of a wild card here. Created by a former Google employee Guido van Rossum, Python is said to be very user friendly and is one of the most versatile languages with applications ranging from scripting macros to creating cloud-based file hosting services, such as Dropbox (High Scalability 2011).

Even though Python is often perceived as a programming language, according to Nelli (2015), *"Python, with all its packages can be considered the best choice for the foreseeable future for those, who want to perform data analysis."* The ability to interface with other languages, such as C and Fortran as well as develop data analysis projects integrated to Web Servers and internet through support libraries makes Python unique among similar languages, such as R and MATLAB.

Based on a suggestion in a great introductory guide by Sheppard (2017), Anaconda package of Python was used. Besides the core version of Python 3.6, Anaconda installation includes additional packages for mathematical computing, statistical analysis and creating graphs as well as an integrated development environment called *Spyder*. Spyder is an IDE specialized for use in scientific applications of Python and it looks quite like RStudio having the variable explorer, help and the console on the right side and the editor on the left side of the screen.

Python is completely free and modular in structure. Modules are practically Python files ending with a file extension .py. These modules include source code of different functions, are usually written in Python and can be viewed in any text editor. This adds transparency and makes it easier to copy and alter prebuilt functions. Different modules are often grouped into folders containing several .py files. Before using the functions in these modules, user must import the module to active workspace. Each module has its own namespace which means it is possible to have different functions in different modules with exactly same name. To access these functions, user must include the path to the module containing the actual function. Path is separated using commas, for example SARIMAX() function for advanced ARIMA models is accessed by typing statsmodels.tsa.statespace.sarimax.SARIMAX() or alternatively: from statsmodels.tsa.statespace.sarimax import SARIMAX().

1.4 Evaluating the software

Nielsen (1993) recognizes several key elements which determine popularity of a given system. System acceptability can be divided into social acceptability and practical acceptability. Social acceptability is based on social norms and it defines what the 'general public' considers socially acceptable. Practical acceptability, on the other hand, is defined by the usefulness, compatibility, reliability and cost of the system. Apple's Keynote, for example, is perfectly viable and in some ways perhaps better presentation software than PowerPoint, but due to lack of compatibility with mainstream Windows-systems, using Keynote is less acceptable than using PowerPoint. Usability, which mainly defines usefulness of a system, has been defined by Nielsen (1993) through five key attributes. These attributes, listed below, are mostly subjective and focus on the ease of use.

Learnability – the user can swiftly start getting some work done
Efficiency – the system should allow high level of productivity
Memorability – a casual user should be able to remember how to use the system after returning to use the system from a break
Errors – a low error rate is preferred and users should be able to revert any accidental changes
Satisfaction – The system should be pleasant to use

Usability testing is usually done by having several test users to perform specific tasks on the system and then asking their opinion. This is not the case in this thesis, therefore, more objective measures are required. Learnability, as well as visual appeal of the user interface are reported as writer's opinions. Data structures, performance and prebuilt modules are more objective criteria. Performance of the software are measured by timing the processes with built-in functions of these software. Therefore, performance measures are subject to the reliability of time measures used by the software. Functions used for timing are presented in appendix one.

Table 1. Comparison of selected software

	SAS	MATLAB	R	Python
Visual appeal	Looks rather outdated and confusing	Visually pleasing, the most modern look	R console looks outdated, RStudio is more modern	Customizable
Data structures	Tables with variable names	Matrices (tables) with variable names	Data frames with column	Time series, data frames,

			indexing, time series	arrays, dictionaries
Prebuilt modules	Limited and sold separately	Very versatile but sold separately	Very versatile, free	The most versatile libraries, free
Availability of help	Comprehensive	Comprehensive	Comprehensive	Comprehensive
Accessibility and Price	Hardest of these to get, individual pricing, $5K+	Easier to get than SAS, commercial license 5300 €	Free for all users	Free for all users
Memorability of syntax	Shows a list of functions when one starts to write, syntax is somewhat odd, too much abbreviations in parameter names	Rather easy to remember, functions show full names for required parameters	Very intuitive, perhaps the easiest to remember, functions show a list of acceptable parameters	Easy if one knows basics of programming, user must remember path to different functions
Unified namespace for all functions	Yes	Yes	Yes	No
User must import functions	No	No	Yes	Yes
Supports tab completion	Yes	Yes	Yes	Yes
Performance	Slowest of the tested	Very good	Good	Very good
Entry requirements	Easy to pick up with the aid of GUI, complicated menus	Requires knowledge of matrices	Might look intimidating at first, rather quick to learn	Requires basic knowledge of programming, e.g. loops
OS Support	Windows, selected Linux versions and z/OS	Windows, Unix/Linux, Mac OS	Windows, Unix/Linux, Mac OS, z/OS	Windows, Unix/Linux, Mac OS, z/OS

To evaluate performance each software were given a task to fill a 1500 by 1500 matrix with pseudorandom numbers drawn from a uniform distribution, then multiply the transpose of this matrix by the original matrix and finally return the inverse of the product matrix. Tests were

first run without printing the resulting matrix to eliminate differences in console performance. In 1500 by 1500 matrices Python and MATLAB were the fastest to complete, both a few seconds faster than R and SAS. Running a second test with print function was performed, because without it SAS does not let user view the resulting matrix. For R, MATLAB and Python printing is a bit unnecessary because they all let user view the resulting matrix even without a print statement. Significant performance differences appeared when a print statement was added to the code. In SAS computing time went up from 4,5 to 59 seconds and it took an additional two minutes and fifty seconds to add the results to the workbook after the computing had finished. Running times with print statement are not comparable between Python and other tested software, because Python only prints some of the first and last values of each row.

Table 2. Running times (seconds) on a 1500 by 1500 matrix

	SAS		MATLAB		R		Python	
	without print	with print	without print	with print	without print	with print	without print	with print
1	4.501	58.423	0.152	4.017	2.34	15.16	0.159	0.774
2	4.303	60.403	0.153	4.005	2.29	14.99	0.143	0.773
3	4.593	58.321	0.155	3.973	2.34	15.05	0.147	0.801
4	4.775	59.95	0.164	4.218	2.28	14.96	0.169	0.754
5	4.487	59.08	0.153	3.958	2.31	15.02	0.147	0.785
Avg	4.53	59.24	0.16	4.03	2.31	15.04	0.15	0.78

After completing the tests on 1500 by 1500 matrices, matrix size was increased to 5000 by 5000. Depressed by the time it took to complete in SAS this test was only run twice. Even without printing SAS took on average a bit over three minutes to complete. R started to struggle but it did complete in one minute and twenty seconds. To see if RStudio decreases performance this test was also run without it on a plain R console. Quite surprisingly completion time was approximately the same with and without RStudio, but the console crashed once without it. It seems that RStudio might improve the stability of R. As the matrix size increased, Python started to gain a small advantage over MATLAB.

Table 3. Running times (seconds) on a 5000 by 5000 matrix

	SAS	MATLAB	R	Python
1	185,86	3,84	84,69	3,04
2	191,235	4,00	83,56	2,98
Avg	188,55	3,92	84,13	3,01

It should be noted that matrices of this size are rare as they have 25 million elements. Storing and computing multiple matrices this large takes a lot of storage space, which may run out on some configurations. Together with system processes MATLAB peaked at 11 GB of total RAM usage, of which system's share was less than five. For curiosity, same R script was run on a 13 inch 2015 MacBook Air with a 1500 by 1500 matrix without print statement. Although it took on average seven seconds longer than on PC, R runs quite smoothly on Mac OS.

2. Stochastic processes and data

2.1 Methodology

For forecasting, two univariate time series were obtained. Gathered data was split into two groups, one of which used for fitting the model and the other one for validating the model. Historical datasets were used so that there was no need to wait for future values of the time series to validate the model.

Research was purely quantitative, and focusing more on the mathematical and methodological side than analyzing the results obtained by data analysis. Therefore, reliability of the data used is less important than the stochastic properties of the data itself. In the first time series, 72 observations were used for fitting the models. 11 observations were left for evaluating the accuracy of the forecasts. In the second time series, 1044 observations were used for fitting the model and 288 observations for evaluating the forecasting power. Forecasting horizon plays an important role, as smaller ARMA models tend to converge rather quickly.

2.2 Heteroscedasticity

Heteroscedasticity is an unwanted property of some datasets. It leads to less efficient estimates in OLS regression and renders ARMA models rather useless. ARMA models expect a constant variance to perform efficiently. In Watsham & Parramore (1997) heteroscedasticity is defined as inconstant variance of residuals. Heteroscedasticity can be diagnosed graphically by looking at the scatter plot with the regression line in OLS regression. In case of heteroscedasticity due to growing variance, the graph would look similar to figure 1.

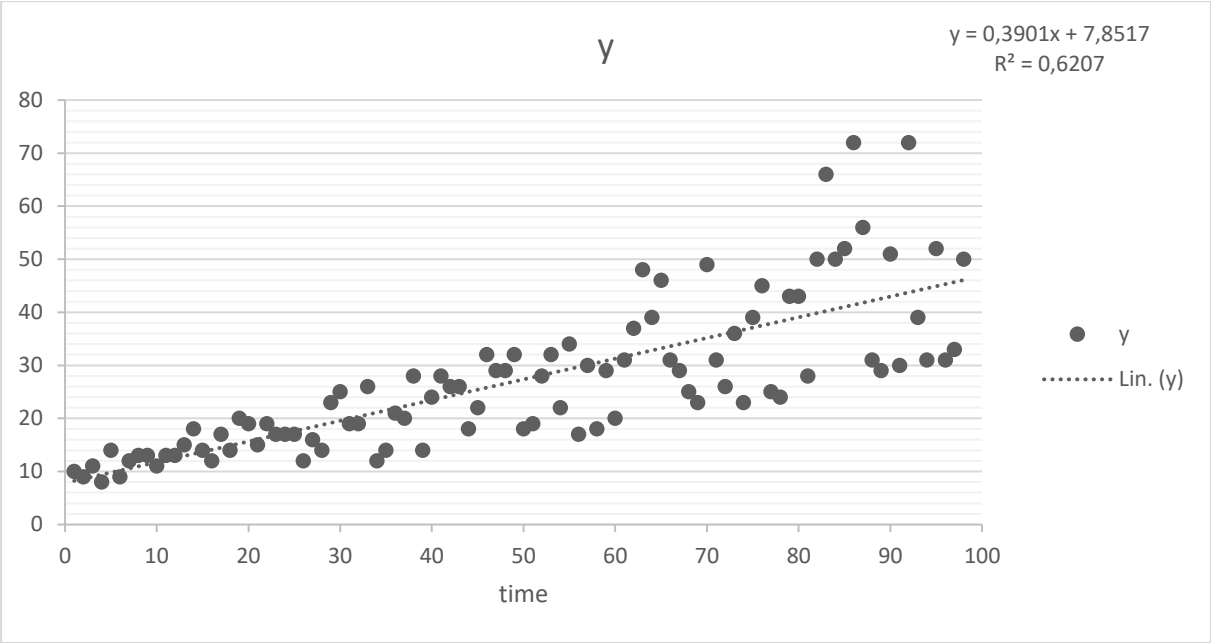

Figure 1. Simulated heteroscedastic series where variance increases in time

Heteroscedasticity tests were originally designed as diagnostics tools for OLS regression and often expect residuals not to be serially correlated. In time series data, heteroscedasticity is rarely a problem, however, it should be tested. In financial time series, heteroscedasticity can be often seen as change of volatility of the return series over time i.e. first difference of an index varies more as time progresses. In the simplest form, heteroscedasticity can be tested by splitting the sample into two subsamples and using the F-test of equality of variances. But, as noted by Box (1953), this test is very sensitive for non-normality of the original sample.

Another simple test was presented in Goldfeld & Quandt (1965). Goldfeld-Quandt test is based on dividing the residuals into two groups and calculating the ratio of error sum of squares (SSE) between these two groups. Sample residuals must be sorted into ascending order before dividing into two groups. For added robustness Wastsham & Parramore (1997) suggest one sixth of the residuals to be removed from the middle of the sample after sorting. Goldfeld-Quandt test statistic $GQ = \frac{SSE_H}{SSE_L}$ has and F-distribution with $\frac{n-c}{2-k}$ degrees of freedom, where k = number of independent regressors, n = number of observations and c = number of removed observations. Perhaps simplicity makes GQ test so popular that it is often presented as the first test of heteroscedasticity in undergraduate textbooks, such as Hill, Griffiths & Judge (2001).

By default, SAS uses White's test for heteroscedasticity. In the case of time series, Breusch-Paegan test or McLeod Li test may be preferred. The latter one tests for conditional heteroscedasticity which may be present in financial time series, such as stock prices and

returns. Conditional heteroscedasticity, i.e. volatility clustering, suggests autoregressive conditionally heteroscedastic models (ARCH-models) to be used instead of ARMA models. White's test was originally presented and proven in generalized matrix form in White (1980). A more practical example with Lagrange Multiplier is given in Brooks (2014), which is presented here slightly modified to suit univariate regression $y_t = \beta_1 + \beta_2 x_{2_t} + u_t$. White's test consists of running an auxiliary regression $\hat{u}_t^2 = \alpha_1 + \alpha_2 x_{2_t} + \alpha_3 x_{2_t}^2 + v_t$ and testing the joint null hypothesis that $\alpha_1 = \alpha_2 = \alpha_3 = 0$. Lagrange Multiplier statistic used in White's test takes R^2 from the auxiliary regression and multiplies it by the number of observations *T*. Attained LM statistic has χ^2 distribution with *k* degrees of freedom, *k* being the number of regressors in auxiliary regression excluding the constant.

2.3 Stationarity

With a practical approach from Watsham & Parramore (1997), a time series is said to be stationary if it is free of trends, shifts and periodicity. In other words, a time series fluctuates around a constant mean with a time-invariant probability distribution. A stationary time series will return to its long-term mean after a random shock, which means it is possible to reliably forecast the time series.

Using a simplified notation from Brooks (2014) and Tsay (2005), more specific definition can be drawn. A time series $\{y_t\}$ is said to be strictly stationary when the joint distribution of $(y_{t_1}, \dots, y_{t_k})$ is identical to that of $(y_{t_1+t}, \dots, y_{t_k+t})$ for all *t*, where $k \in Z$ and $(t_1,\dots,t_k)$ is a collection of *k* integers. However, strict stationarity is a very hard condition to meet in empirical data, which has led to a weaker definition of stationarity. A time series $\{y_t\}$ is said to be weakly stationary if E(y)= μ and Cov(y_t , y_{t-s}) = s, which only depends on *s*. The covariance s is called the lag-*s* autocovariance of y_t . Presented conditions for weakly stationary process state that a stationary process has a constant mean, a constant variance and a constant autocovariance structure.

Two important properties of covariance γ_s presented in Tsay (2005) state that $\gamma_0 = Var(y_t)$ and $\gamma_{-s} = \gamma_s$, which means that the autocovariance at lag 0 equals the variance of y_t and the autocovariance depends only on the difference between t_1 and t_2. Because the autocovariances depend on the units of measurement of y_t, they must be normalized by dividing by the variance to obtain the autocorrelations (Brooks 2014).

$\tau_s = \frac{\gamma_s}{\gamma_0}$, s = 0, 1, 2, …

Now, plotting the series τ_s of correlation coefficients against *s* yields to a graph known as the correlogram, a visual representation of the autocorrelation function (ACF). If the series is stationary, autocorrelation will decrease quickly when the lag increases (Kirchgssner, Wolters et al. 2013).

2.4 Random walk

Random walk is a commonly used model for financial time series. In random walk, each change is independent of all previous changes. Each change is drawn from the identical probability distribution with a constant variance and mean. With a notation from Watsham & Parramore (1997), a random walk process may be presented as $y_t = y_{t-1} + \varepsilon$, where ε exhibits zero mean and a constant variance. Because financial time series tend to increase in value over time, a drift element is often included. A random walk with drift is expressed as $y_t = y_{t-1} + \alpha + \varepsilon$, where α is the slope of the time trend.

A special case of random walk is called white noise. A white noise series has zero mean, a constant variance and zero correlation between successive observations. Therefore, white noise contains no significant information and consequently most statistical models expect error terms to be white noise. In general, random walk and white noise processes are stationary, except random walk with drift.

2.5 Measures of error

2.5.1 MSE

Mean squared error was used as one of the main measures of forecasting accuracy. As the name suggests, mean squared error is calculated by taking a simple mean of squared errors using the following formula

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

where $\hat{y}_i$ represents estimated value, y_i observed value and *n* is the number of observations. Taking a square root from mean squared error yields to a measure called Root MSE. This measure can directly be interpreted since it follows the same scale than the original values.

2.5.2 MAPE

Mean absolute percentage error will be used as a secondary measure of forecasting accuracy, because Tofallis (2015) claims that MAPE is probably the most commonly used measure of forecasting accuracy in businesses and organizations. However, as a measure of forecasting accuracy it is biased in such way that it systematically selects methods which produce too low predictions. For this reason, MAPE will not be used as a primary measure of forecasting accuracy in this thesis. MAPE is calculated using the following formula.

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right|$$

An adjusted version of MAPE, often called symmetric mean absolute percentage error or SMAPE was proposed by Armstrong (1985) an alternative to overcome the effects of biased MAPE.

$$SMAPE = \frac{100\ \%}{n}\sum_{t=1}^{n}\frac{|\hat{y}_t - y_t|}{(|y_t| + |\hat{y}_t|)/2}$$

Armstrong's original formula leads to values between 0 % and 200 %. For this reason, number two is often dropped from the denominator. Despite these modifications, Armstrong's formula was later found biased in the opposite direction, favoring over-forecasting. (Tofallis 2015)

2.6 Data

Data consists of econometric time series from ThomsonOne and Datastream by Reuters. First of the two time series is recorded monthly, second one daily. Dataset used includes demand for new cars in the United States and Standard & Poor's 500 stock index. Of these, new cars registered per month in the United States was found quite surprisingly trend stationary with a decreasing trend over time. Based on previous research, log transformation was considered relating to return series of stock indices and stock prices.

2.6.1 Demand data

First dataset used in this thesis consists of monthly demand of new cars in the United States. Original data sample was form January 1975 to November 2016 and it was split for a period of 36 years ranging from 1980 to 2015, six years from 2010 to 2015 and the validation set consisting 11 observations from 2016. By splitting the sample this way it is possible to avoid some of the influence of financial crisis in 2008 and see if it is possible to accurately forecast

near future using rather short time series. One unit in a graph indicates 1000 vehicles. Long-term sample is presented in figure 2.

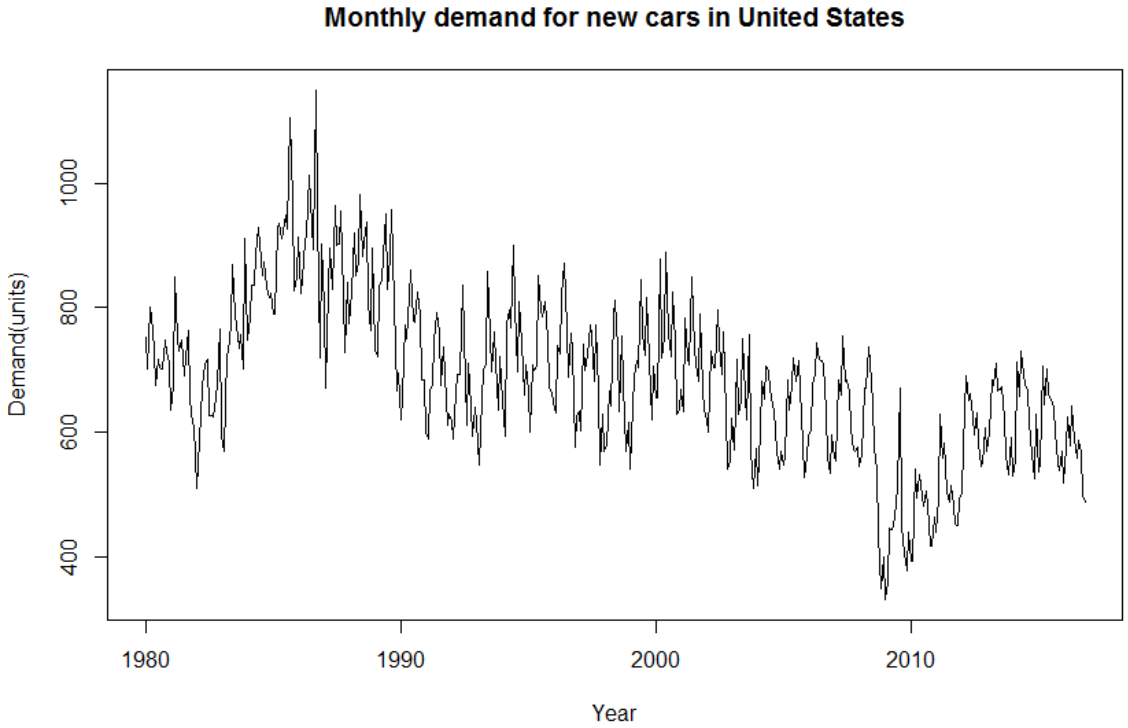

Figure 2. Monthly demand of new cars in the United States

Data was plotted in R using a built-in function for seasonality and trend decomposition. Results of the analysis are plotted in Figure 2. From the figures, it is easy to see that the demand is highly seasonal and it has a decreasing trend over time. This suggests that the time series might be trend-stationary, i.e. stationary can be introduced by removing the linear time trend.

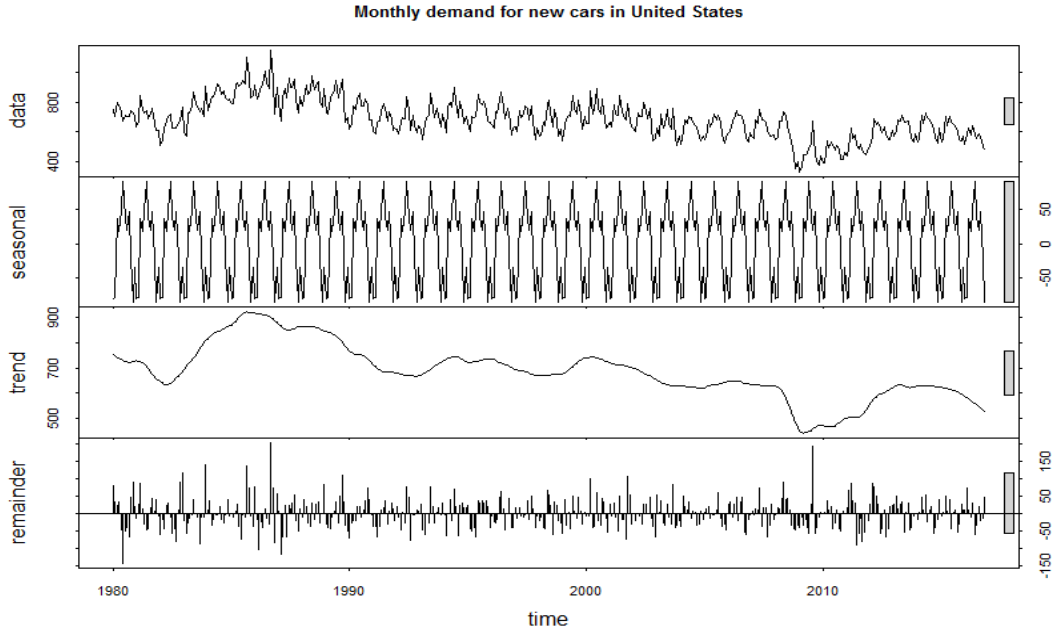

Figure 3. Decomposition of monthly demand

Diagnostics were run on the full sample from January 1980 to November 2016 and the distribution was plotted with normal curve. Using the Goldfeldt-Quandt test data was not found heteroscedastic, p-value for null hypothesis of SSE equality being 0.33. Although the sample seems to follow normal distribution as whole (see fig. 4), due to decreasing time trend subsamples from different points in time have different mean and thus do not follow the same normal distribution. Therefore, using the F-test of the equality of variance would give misleading results.

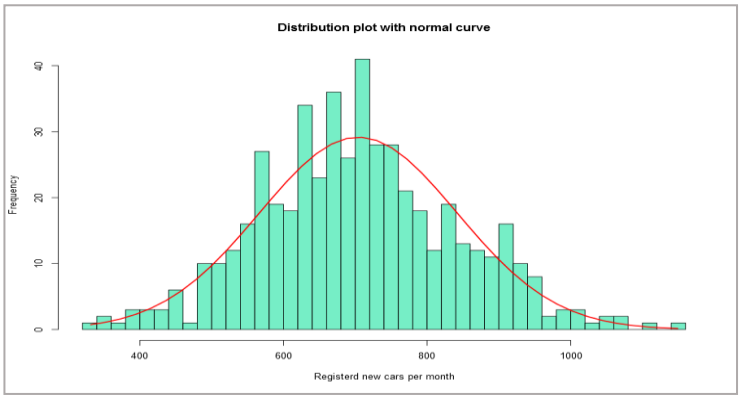

Figure 4. Distribution plot of monthly demand with normal curve

2.6.2 Stock prices

Daily values of Standard & Poor's 500 were selected as a time series representing stock data. Data used for fitting the model was gathered from January 2012 to December 2015 and is presented black in figure 5. Grey line represents data used for validating the model.

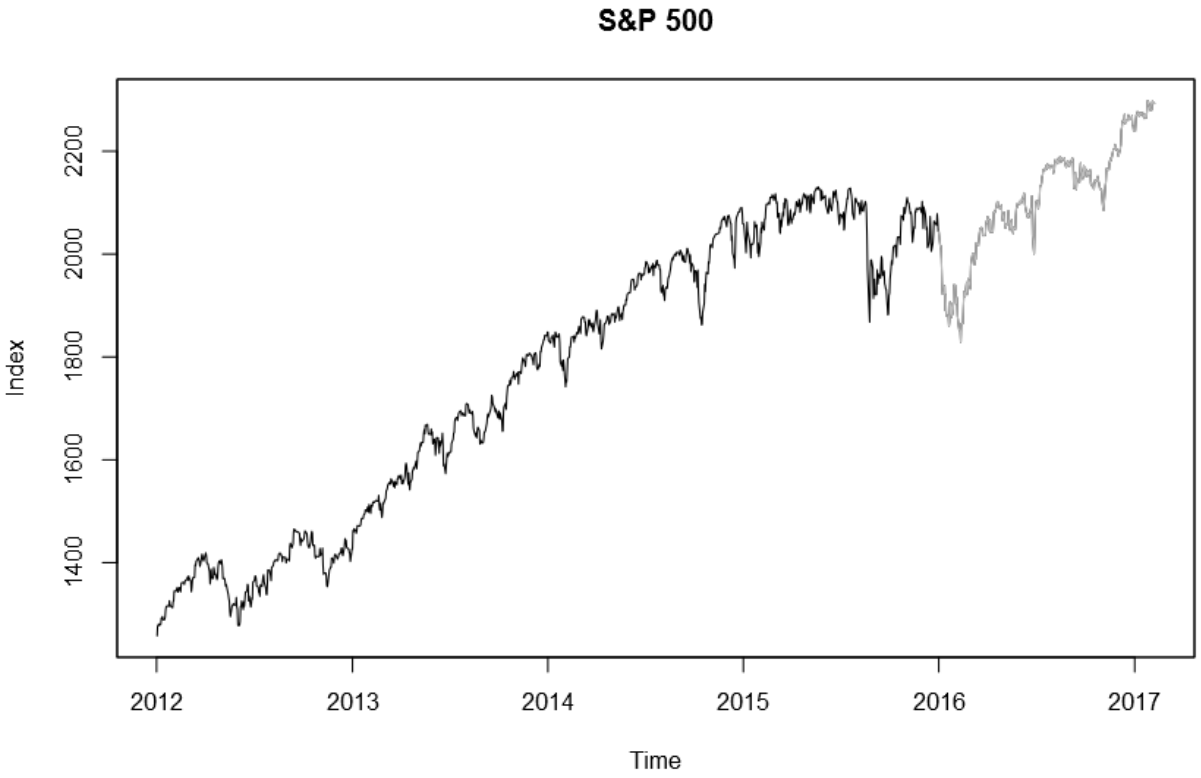

Figure 5. Standard & Poor's Index 2012-2017.

Using the White's test for heteroscedasticity, data was not found heteroscedastic. This indicates that the residuals do not correlate significantly with the past observations of the data. It can be

seen from figure four that the series has a drift element, because it tends to increase in value over time. Learning set for this series is much longer, 1044 observations to be precise. Validating set and forecasting horizon is 288 observations, from January 2016 to February 2017.

3. Naive models

3.1 Forecasting by mean

One of the simplest methods of forecasting is predicting that all future values are equal to the mean of the historical values. Here, a forecast *h* periods after time *T* for a series $\{y_t\}$ would be $\bar{y} = (y_1 + \cdots + y_T)/T$.

Using a *fpp* package in R, forecasts for horizon *h* based on a time series *y* are attained simply by typing meanf(y, h). Using meanf() instead of simply taking the mean of the series yields to automatic 80 and 95 per cent confidence limits and improved plotting, so it is the recommended way in R. In any other software, taking the mean of the series is sufficient.

3.2 Naïve method

A simple method, where all forecasts are simply set to be the value of the last observation. Here, a forecast one step ahead time *t* is defined as $y_{t+1} = y_t$. The same *fpp* package in R has a function naive(y, h) for the naïve method.

A more advanced variation of the naïve method is available for seasonal time series. Here, a forecast *h* periods after time *T* equals to the value of *y* from the corresponding previous observation. E.g. a monthly forecast for January would be the value of *y* in past January. Using a notation from Hyndman & Athanasopoulos (2013), a forecast *h* periods after time *T* would be given as y_{T+h-km}, where $m = length\ of\ the\ seasonal\ period$ and $k = \lfloor (h-1)/m \rfloor + 1$, where $\lfloor u \rfloor$ is the integer part of u. R has a function snaive(y, h) which gives the forecasts using seasonal naïve method.

3.3 Drift method

Drift method is a variation of naïve method which allows forecasts to increase or decrease over time. Drift method is suitable for time series with trend. The amount of change over time is set to be the average change in the historical values. Using a notation from Hyndman &

Athanasopoulos (2013), a forecast *h* periods after time *T* is equal to $y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1})$, which simplifies to $y_T + h(\frac{y_T - y_1}{T-1})$. R has a function rwf(y, h, drift = TRUE) for including trend in naïve method. Drift method yields to rather similar results than simple regression because they both estimate the slope of the original sample. Yet, instead of estimating intercept at time zero, the forecasted line in drift method begins from the last observed value.

3.4 Theta method

Inspired by the success of Theta method in M3 competition (Makridakis, Hibon 2000), this model was included in comparison. Originally presented as a specific decomposition technique with two autoregressive terms in Assimakopoulos & Nikolopoulos (2000), this model was later identified as simple exponential smoothing with drift by Hyndman & Billah (2003). Based on results from Makridakis & Hibon (2000), in this comparison Theta model represents the best performing exponential smoothing model. Theta method was originally presented as $y''_{new}(\theta) = \theta * y''_{data}$, where $y''_{data} = y_t - 2y_{t-1} + y_{t-2}$ at time t in Assimakopoulos & Nikolopoulos (2000), but for the ease of use formula $\hat{y}_t(h) = \tilde{y}_t(h) + \frac{1}{2}\hat{b}_0(h - 1 + \frac{1}{\alpha})$, where $\tilde{y}_n$ is SES forecast of the series $\{y_t\}$ and slope of the trend is half that of the fitted trend line through the original time series $\{y_t\}$ from Hyndman & Billah (2003) will be used. Since the Theta method has not gained wider popularity among the forecasters, it is not present as a separate function in SAS. It can be found in *forecasting* and *forecTheta* packages of R.

3.5 Simple regression (OLS)

Ordinary least squares regression, also known as the classical linear regression model fits a regression line to the data minimizing the sum of squared errors. If the assumptions of OLS regression are fulfilled, i.e. $E(u_t) = 0$, $Var(u_t) = \sigma^2 < \infty$, $Cov(u_i, u_j) = 0$, $Cov(u_t, x_t) = 0$, OLS estimates are said to be best linear unbiased estimators (BLUE). This means that estimators $\hat{\alpha}$ and $\hat{\beta}$ are true estimates of α and β and per Gauss-Markov theorem, there are no alternative linear estimators that would have smaller variance. (Brooks 2014) For diagnostics and calculation of p-values, assumption $u_t \sim N(0, \sigma^2)$ is usually also required.

Simple regression can be used in forecasting univariate time series by using time *t* or index number *i* of the observation as an explanatory variable. More importantly, autoregressive models can be seen as simple regressions with lags of y_t as independent variables. Those independent variables are functions of the original values in univariate time series. Therefore,

estimators $\hat{\beta} = \{\hat{\beta}_1, \dots, \hat{\beta}_n\}$ can be solved using the generalized form of multivariate OLS regression.

Although all the tested software packages have prebuilt modules for OLS regression, matrix form of the regression equation may be preferred in matrix focused software packages, such as MATLAB and Spyder. A regression equation $y_t = \beta_1 + \beta_2 x_{1t} + \cdots + \beta_k x_{kt} + u_t, \ t = 1,2, \dots, T$ can be expressed in matrix form as $y = X\beta + u$, where y is a column vector containing observed values of *y*, X is a matrix of dimension $T \times (k+1)$ containing a column of ones and the values of independent variables, β is a coefficient matrix of dimension $(k+1) \times 1$ and u is a column matrix of the error terms. Coefficient estimates $\hat{\beta}$ for the intercept and k independent variables are now given by the following equation. (See Brooks, 2014 pp. 168-169 for proof)

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_{k+1} \end{bmatrix} = (X'X)^{-1}X'y$$

Simple regression with index number i of observation can be calculated in MATLAB by creating a column vector of ones for calculating the intercept, creating another column vector for indices, concatenating these matrices horizontally, importing data to a column vector called y and simply typing $b = (X' * X)\text{^}(-1) * X' * y$. For more detailed results e.g. p-values of coefficients and F-value of the model one may type fitlm(), but this returns a linear model object, not an array of coefficients.

4. ARIMA models

ARMA/ARIMA models are a set of stochastic models made famous by Box and Jenkins. It has been shown by Newbold and Granger (1974) that in general, ARMA models outperform simpler methods such as exponential smoothing. According to Newbold in Makridakis and Wheelwright (1979), this hypothesis is also supported by Reid (1969). In later research, Makridakis and Hibon (2000) found out that ARMA family performs especially well in macro economical and financial time series. In general, ARMA models are effective in forecasting yearly and monthly time series.

Autoregressive integrated moving average models consist of two key components and a method to introduce stationarity. These components are called *autoregressive* and *moving average* and they are notated with a degree of *p* and *q* respectively. Notation AR(p) indicates that there are *p* autoregressive components in the stochastic equation and notation MA(q) indicates that there are *q* moving average components in the equation. (Box, Jenkins 1976)

4.1 Autoregressive (AR)

Except the fact that autocorrelation is an unwanted property of multivariate regression models and autoregressive models are built on serial correlation between the past values of y_t, these two types of models are alike. Indeed, autoregressive process can be presented as a multivariate linear model with past values of y_t as independent variables. For example, AR(3) process would be given as $y_t = \alpha + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 y_{t-3}$. (Watsham, Parramore 1997)

Autoregressive models can be identified by looking at the autocorrelation function. As shown in Box and Jenkins (1976), independent of the sign of the autocorrelation coefficient, autocorrelation function in first-order autoregressive processes decays exponentially to zero. In the presence of negative autocorrelation, autocorrelation function also oscillates in sign as it approaches to zero.

4.2 Moving average (MA)

Moving average models are also similar to simple regression, except now the current value of y_t is given as a linear function of the past values of the error term. Using a simple notation from Watsham & Parramore (1997), a moving average process MA(3) is given as $y_t = \alpha + \beta_1 u_{t-1} + \beta_2 u_{t-2} + \beta_3 u_{t-3}$. It should be noted that the coefficients β in this notation have changed in sign since Box and Jenkins (1976) to make interpreting similar to general regression equations. It also should be noted that despite some similarities, the moving average process is distinct and should not be confused with the simple smoothing method *running average*, sometimes referred to as *rolling average* or *moving average*.

Moving average models can be identified by looking at the partial autocorrelation function, PACF. In a pure moving average process, partial autocorrelation function decays geometrically to zero and the order q is the number of statistically significant autocorrelations. In a combination of autoregressive and moving average processes, both the ACF and PACF decay exponentially to zero. (Makridakis, Wheelwright 1979, Brooks 2014)

4.3 Degree of Integration (I)

Recall the earlier discussion about stationarity. To better understand the importance of stationarity, we can use an example from Brooks (2014). Let $y_t = \mu + \rho y_{t-1} + u_t$ be an AR(1) process. Now, there are three possible cases: a) $\rho < 1$, where $\rho^T \to 0$ as $T \to \infty$, b) $\rho = 1$, where $\rho^T = 1 \; \forall \; T$ and $y_t = y_0 + \sum_{t=0}^{\infty} u_t$ as $T \to \infty$ and c) $\rho > 1$ which is an explosive case as $\rho^3 > \rho^2 > \rho$. Case A is the stationary case, because as time goes on the effect of a random shock will die away and the series will return to its long-term average and therefore it is easy to predict. Case B is a unit root case, where the value of y_t is the initial value of y plus an infinite sum of past shocks. Forecasting a non-stationary time series is not easy, because one can't reliably forecast the next random shock. However, the unit root case is relatively easy to convert into stationarity. Finally, forecasting the case C is irrelevant, because empirically econometric time series do not explode to infinity.

As most economic time series exhibit trends over time and hence non-stationarity, they must be converted to stationarity. Depending on the case, this can either be done by removing the time trend or differencing, but not both. A trend-stationary process follows the equation $y_t = \alpha + \beta t + u_t$ and is called *deterministic non-stationarity.* This can be converted into stationarity simply by subtracting the OLS estimates from y_t and using the stationary series of residuals in further analysis. A more common case is the random walk with drift, known as *stochastic non-stationarity* or *the unit root case,* where $y_t = \mu + y_{t-1} + u_t$. This can be converted into stationarity by subtracting y_{t-1} from y_t to obtain $\Delta y_t = \mu + u_t$. While both cases of non-stationarity exhibit trends over time, care should be taken in choosing the right way to remove the time trend. (See Brooks, 2014, pp. 357-359 for proof.)

For testing the unit root stationarity, augmented Dickey-Fuller test is recommended. Generally, three alternative forms of the equation $\Delta y_t = \alpha_0 + \alpha_1 t + \gamma y_{t-1} + \sum_{i=1}^{m} a_i \Delta y_{t-1} + v_t$, where $\Delta y_{t-1} = y_{t-1} - y_{t-2}$, $\gamma = \rho - 1$ and m is the number of lags, are presented, depending on whether the time series has a zero mean, a constant non-zero mean or a trend. Augmented Dickey-Fuller test-statistic, which has a τ distribution, is calculated by dividing γ by the standard error of γ. If the H_0 is true, $\gamma = 0$ and therefore $\rho = 1$ and the series has a unit root i.e. is integrated of order one. H_1 states that $\rho < 1$ and the series is stationary. For choosing the number of lags used, a rule of thumb is suggested, i.e. for a quarterly data, choose four lags and for a monthly data choose 12 lags. (Brooks, 2014)

Augmented Dickey Fuller unit root test can be conducted in SAS by *proc arima* and it is a built-in function in Python Anaconda package. R has adf.test() function in *tseries* package. MATLAB has adftest() in *Econometrics Toolbox*.

4.4 The Box-Jenkins Methodology

The Box-Jenkins methodology, introduced in Box and Jenkins (1976), consists of three stages presented in figure 6. This is also the format in which SAS calculates ARMA models. The first phase is identifying the correct order of p and q, then using the identified model for estimating the autoregressive and moving average parameters. The adequacy of these parameters must be checked before proceeding to actual forecasting. If the model is not adequate, one should try different values for p and q.

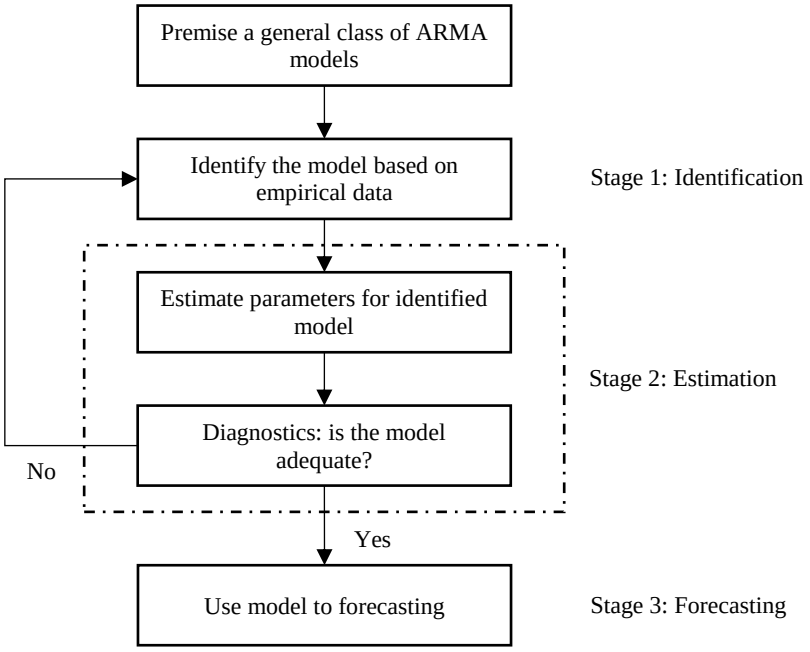

Figure 6. The Box-Jenkins Methodology

While simple exponential smoothing models (Holt-Winters) can be considered as ARIMA(0,1,1) models with a MA coefficient $(1-\alpha)$, the Box-Jenkins methodology allows generating multiple stochastic models and choosing the best of them for actual forecasting. This added versatility makes ARMA family superior to smoothing models, as they better capture the trends and seasonality commonly present in financial data. Besides being overly simplistic and inflexible, smoothing models have another major disadvantage since the forecasts generated by exponential smoothing do not converge on the long-term mean of the series as the forecasting

horizon increases. (Brooks 2014) The only drawback, presented in 1970s is that compared to exponential smoothing models, the Box-Jenkins method is far more demanding in terms of computing. (Makridakis, Wheelwright 1979) However, a lot has happened since 1970s in the semiconductor industry. What once took ages to complete is nowadays computed in several milliseconds. Therefore, the only limitations are in availability of skilled manpower.

4.5 Information criteria

Information criteria is a tool which can be used for identifying the order of autoregressive moving average models. The idea behind this concept is that each additional explanatory variable increases the goodness of fit and the accuracy of the forecasts. Simultaneously, each added variable increases uncertainty due to change in degrees of freedom. Information criteria imposes a penalty on lost degrees of freedom and thus balances the effects of adding more variables. To improve the information criterion, each added variable must improve the explanatory power of the model more than the penalty term changes the criterion. (Akaike 1974, Brooks 2014)

In Brooks (2014) Akaike's information criteria is defined as: $AIC = \ln(\hat{\sigma}^2) + \frac{2(p+q+1)}{T}$, where $\hat{\sigma}^2$ is the variance of residuals and T is the sample size. A bias correction for Akaike's criteria in small samples was proposed in Hurvich & Tsai (1989) as $AICc = n\ln(\hat{\sigma}^2) + n\frac{1+m/n}{1-(m+2)/n}$, where *m* and *n* denotes matrix dimensions. This bias corrected information criteria was later rewritten in Anderson, Burnham & White (1994) as $AICc = AIC + \frac{2(K+1)(K+2)}{n-K-2}$, where n = sample size and K is the number of regressors.

Another possibility is using Schwarz's Bayesian information criteria, which imposes a smaller penalty on added explanatory variables. Schwarz's Bayesian information criteria is defined in Brooks (2014 as $BIC = \ln(\hat{\sigma}^2) + \frac{p+q+1}{T}\ln(T)$, where $\hat{\sigma}^2$ is the variance of residuals and T is the sample size.

5. Fitting the models

This is the beginning of the empirical part of this thesis, where described methods are applied to two different time series. First of the series is demand of new passenger vehicles in the United

States. A one unit increase or decrease in the series represents change of one thousand vehicles. Second of the series is S&P 500 stock index, where one unit change represents one index point.

5.2 Demand data

In figure 7, black line indicates data sample used for fitting the models and grey line is the validation data. Simple models were fitted first and the results are presented in figure 8. Besides long-term time trend, all other models were fitted using the sample presented in figure 7. Time trends were fitted using linear OLS regression over time.

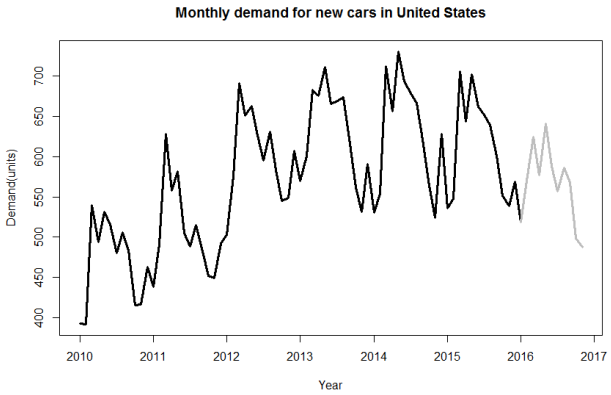

Figure 7. Monthly demand of new cars in United States

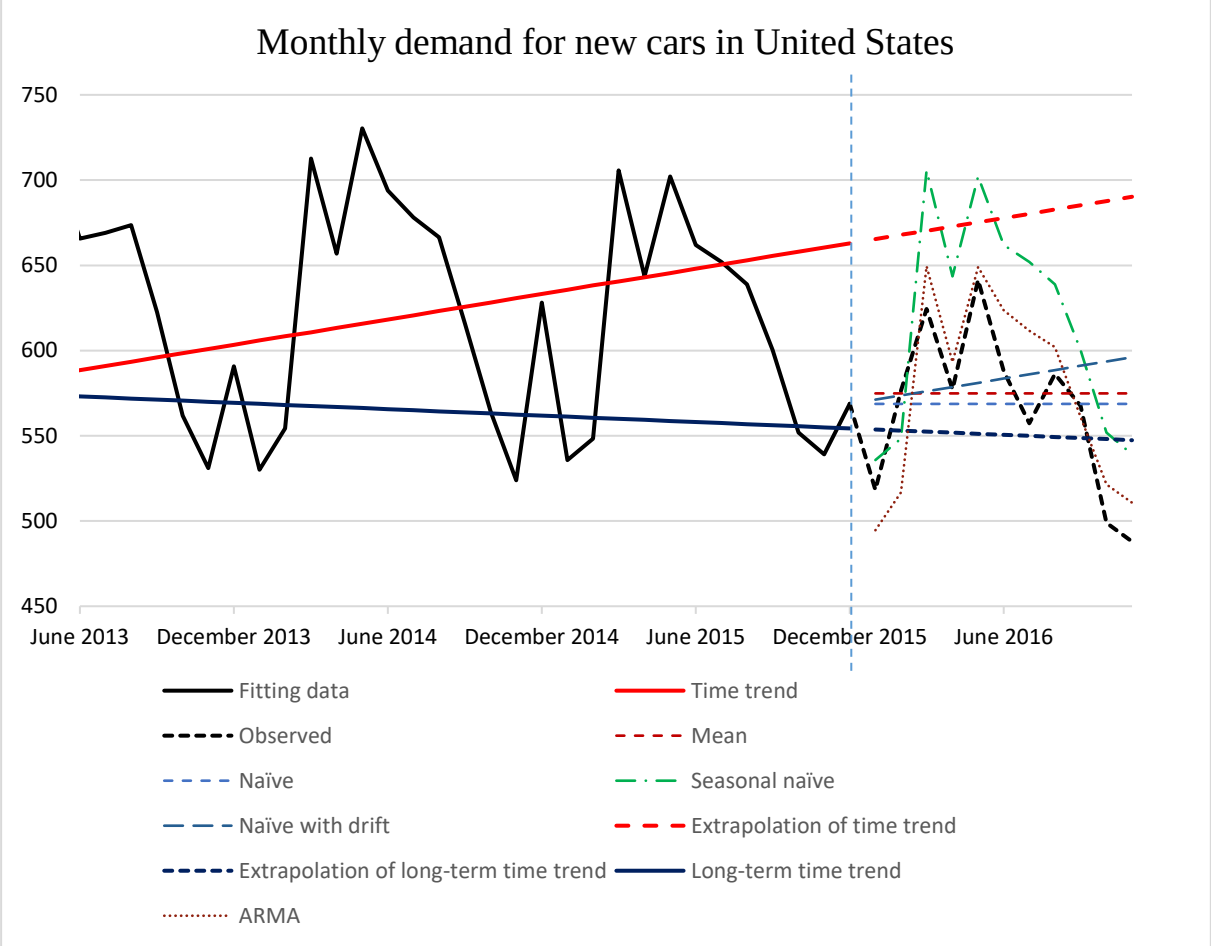

Figure 8. Forecasts, theta forecasts are presented in appendix three.

For identifying the correct ARMA model, autocorrelation function and partial autocorrelation function were plotted in R. Augmented Dickey-Fuller test was performed on the model fitting

and identification series, which was found trend stationary at p-value <0.01 for non-stationarity. Autocorrelation function shows geometrically decaying autocorrelation for the first six lags, after which the AFC increases until the twelfth lag and then decreases again. This pattern indicates seasonality in such way that the values of the current month depend on the values of the corresponding month last year. Values above or below the dotted line are statistically significant with a significance level of 0.05.

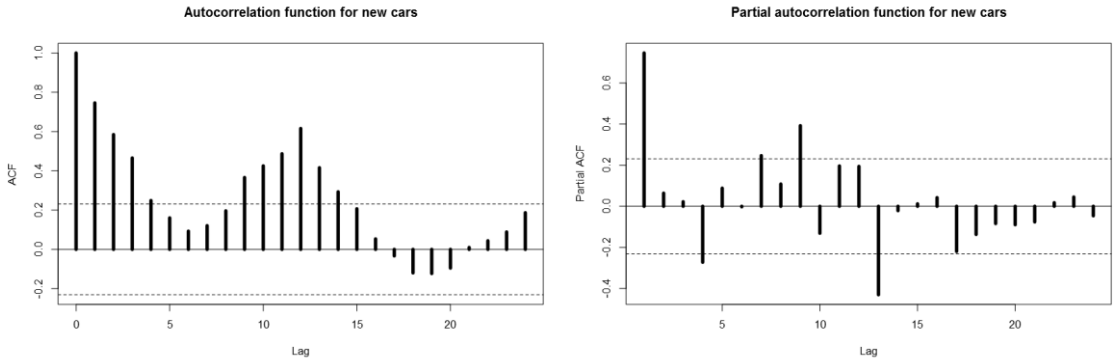

Figure 9. Autocorrelation and partial autocorrelation functions

Based on the autocorrelation and partial autocorrelation functions, ARMA(14,1) model was fitted on the unmodified time series to capture most of the variation. 13th lag in partial autocorrelation function was the last statistically significant so fourteenth autoregressive and fist moving average lags were chosen for added degree of freedom. ARMA(4,1) model was chosen as an arbitrary one and fitted on the data. R function auto.arima() from package *forecasting* was also used to autodetect another ARIMA model. Description of the method used in autodetection is presented in Hyndman & Khandakar (2008), basically it is based on minimizing corrected Akaike's information criteria. Quite surprisingly resulting model was a combination of MA model and a seasonal MA model. Finally, an intuitive ARMA(12,12) model was fitted because of the nature of seasonality in the data. After fitting these models, detrending was performed by removing the linear trend from the fitting series. Suggested by ACF and PACF for the residuals, ARMA(13,1) model was fitted. Forecasts for residuals were added to the linear extrapolation of the time trend. Root MSEs of model fitting and validation are presented in table 4 below. For comparison, Root MSE of naïve method in validation data for 11 step forecast is 45.98 and anything below that will be more informative.

Table 4. Comparison of ARMA models, root mean squared errors

	Fitting data	Validation data 3-step forecast	Validation data 11-step forecast
Auto-ARMA	22.94	40.37	43.12
ARMA(14,1)	25.10	39.80	31.04
ARMA(12,12)	22.03	45.81	41.93
ARMA(4,1)	50.68	39.48	44.65
Detrended ARMA(13,1)	24.78	39.06	76.04

As it is apparent, Root MSE in data used for fitting the model does not directly translate to Root MSE in validation data. This is especially true for detrended model, in which time trend from January 2010 to December 2015 was used to introduce horizontality. It can be seen in figure 2 that the time trend in data used for fitting the models is opposite in sign compared to the long-term time trend of the series. Although the detrended ARMA model forecasts residuals correct in sign and is best for the first three steps, in longer forecasting horizon it is incapable to overcome errors caused by change in time trend. Arbitrary ARMA(4,1) model has high amount of error in the data used for fitting the model, which indicates the model fits poorly. However, as the model converges quickly to the mean, even a poorly fitted ARMA model might yield adequate results.

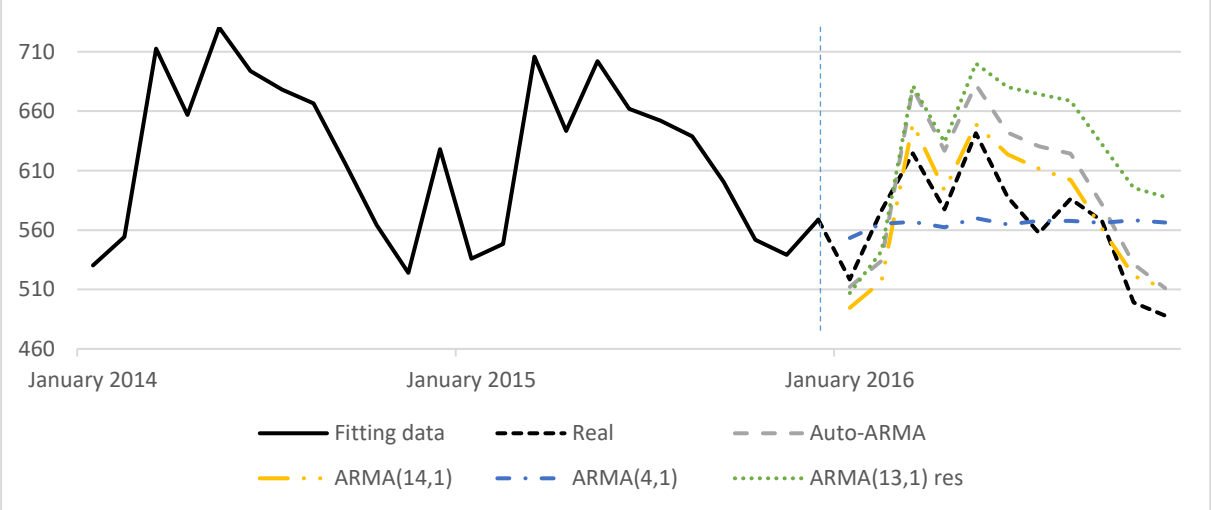

Figure 10. ARMA models

A full comparison of fitted ARMA models is presented in figure 10 above. Arbitrary ARMA(4,1) model has so short memory it does not capture any seasonality present in the data. Diagnostics, such as white noise test of residuals and overfitting could be used to assess quality of fitted model. Both of these tests suggest that ARMA(4,1) is inadequate and residuals still

contain some useful information. Largest of the fitted models, ARMA(14,1) was plotted in comparison with simpler methods in figure 8.

Table 5. Errors for a 3-step forecast

	Root MSE	Rank	MAPE	Rank	SMAPE	Rank	Overall rank
Mean	43.43	6	6.37	7	6.29	6	6.3
Naïve	43.61	7	6.66	8	6.65	8	7.7
Seasonal naïve	50.71	11	7.10	10	6.85	9	10.0
Naïve with drift	41.40	5	6.14	5	6.08	5	5.0
Time trend	103.47	12	17.20	12	15.55	12	12.0
Long-term time trend	48.23	9	7.47	11	7.66	11	10.3
ARMA(12,12)	45.81	8	6.82	9	7.16	10	9.0
Auto-ARMA	40.37	4	5.80	3	5.77	3	3.3
ARMA(14,1)	39.81	3	6.31	6	6.51	7	5.3
ARMA(4,1)	39.48	2	5.98	4	6.06	4	3.3
Detrended ARMA(13,1)	39.06	1	5.77	2	5.70	2	1.7
Theta	49.70	10	5.66	1	5.36	1	4.0

Errors for a 3-step forecast are presented in table 5 above. In short forecasting period filling the requirement of horizontality in ARMA model yields to the best results. In general, tested ARMA models perform better than naïve method. Naïve method with drift element performs surprisingly well. Errors for a longer, 11-step forecasting horizon are presented in table 6 below. Because detrended series gets different values than unaltered series, detrending was reversed by adding the extrapolation of time trend to forecasts before calculating the measures of error.

Table 6. Errors for a 11-step forecast

	Root MSE	Rank	MAPE	Rank	SMAPE	Rank	Average rank
Mean	46.78	6	6.50	4	6.30	3	4.3
Naïve	45.98	5	6.53	5	6.40	5	5.0
Seasonal naïve	59.88	10	9.78	10	9.29	10	10.0
Naïve with drift	52.93	8	7.23	7	6.89	7	7.3
Time trend	122.56	12	20.65	12	18.29	12	12.0
Long-term time trend	47.57	7	7.30	8	7.39	8	7.7
ARMA(12,12)	41.93	2	6.27	2	6.18	2	2.0
Auto-ARMA	43.12	3	6.79	6	6.58	6	5.0
ARMA(14,1)	31.04	1	4.70	1	4.67	1	1.0
ARMA(4,1)	44.65	4	6.44	3	6.35	4	3.7
Detrended ARMA(13,1)	76.04	11	12.56	11	11.71	11	11.0
Theta	54.6	9	8.53	9	8.11	9	9.0

In a longer forecasting horizon, ARMA models still perform better in general. ARMA(14,1) model, which was selected by ACF and PCF function of the unaltered data is the most accurate model by all measures. Naïve method performs better compared to its modifications and extrapolation of simple linear regression based on long term data seems to improve in accuracy as the forecasting horizon increases and the series converges to its long-term time trend. While the auto-selection function in R does not necessary give the best possible model, it does generate rather good forecasts for different forecasting periods with minimal effort. The most accurate ARMA(14,1) model is plotted in a graph 8 in comparison with simple methods. Comparison between ARMA(14,1) and Theta model is presented in appendix 3. Although ARIMA models perform better, difference between ARMA models and simple methods is not that significant besides ARMA(14,1). This indicates, that only a correctly specified ARMA model is beneficial compared to naïve method.

When the ARMA(14,1) model is inspected more thoroughly, it is noticeable that a lot of used autoregressive terms are not statistically significant. This leads to much higher Akaike's information criteria compared to the Auto-ARMA, which selects the model by minimizing corrected Akaike's information criteria. One way to improve the model would be removing the autoregressive terms which are not statistically significant. Both SAS and MATLAB excel here, because they let user to define a vector of integers representing lags that should be included in the model. In R and Python, user must define zero as a fixed value for the unwanted parameters. While the original model has AIC of 720.42 and BIC of 759.12, a model with 1st, 12th and 13th autoregressive lag has AIC of 703.83 and BIC of 715.22. These statistically significant lags contain most of the relevant information and forecasts based on them are similar to ARMA(14,1). This reduced model also passes the white noise test for residuals.

To compare possible differences between different software, AR model with 1st, 12th and 13th autoregressive lag was fitted and forecasted using the same learning data as before. Parameter estimates are presented in table 7 below. R does not print out t-values as default.

Table 7. Fitted AR models in different statistical software

	MATLAB			R		SAS			Python		
	Value	Standard Error	t-value	Value	Standard Error	Value	Standard Error	t-value	Value	Standard Error	t-value
Const	33,03	24,60	1,34	544,25	53,14	420,45	24,20	17,37	56,49	33,92	1,67
AR 1	0,83	0,09	9,66	0,86	0,06	0,83	0,07	11,70	0,76	0,11	6,66
AR 12	0,92	0,06	16,45	0,88	0,04	0,92	0,08	11,83	0,87	0,08	11,66
AR 13	-0,81	0,11	-7,15	-0,78	0,07	-0,75	0,10	-7,26	-0,72	0,15	-4,91

While autoregressive coefficients estimates are similar in all software, there is quite a difference in the constant. R and SAS estimate constant much higher than MATLAB and Python. Python seems to give more weight to more recent observations, as can be seen from a figure 11 below. Blue line represents fitted values and orange line original values. Python function ARMA() does not allow selecting which of the individual lags should be included and which excluded so a more complex SARIMAX() function was used. SARIMAX stands for *seasonal autoregressive integrated moving average with exogenous variables* and is a close match to ARIMA function in MATLAB, R and SAS.

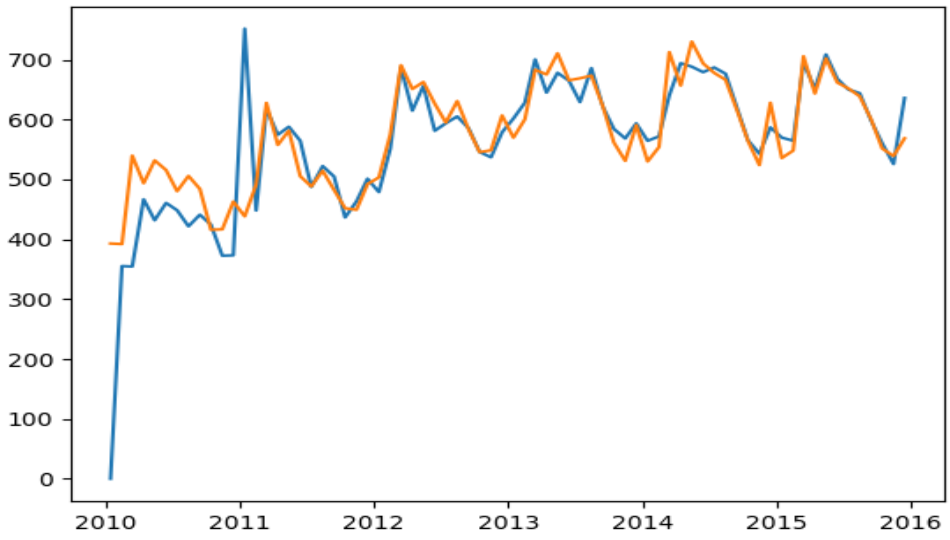

Figure 11. Fitted values in Python

Compared to SAS, Python seems to have a rather different way of fitting the model. Huge difference in fitted and observed values for 2010 to 2011 might be due to using 12^{th} and 13^{th} lag in the model. Because there are no input values for the model to use in estimating the first observed value, Python estimates it to be zero. Second estimated value gets first observed value as an input value for the first autoregressive lag so the second fitted value is 0.76 times first observed value plus a constant 56.49. Oddly Python does not include constant in the first estimated value.

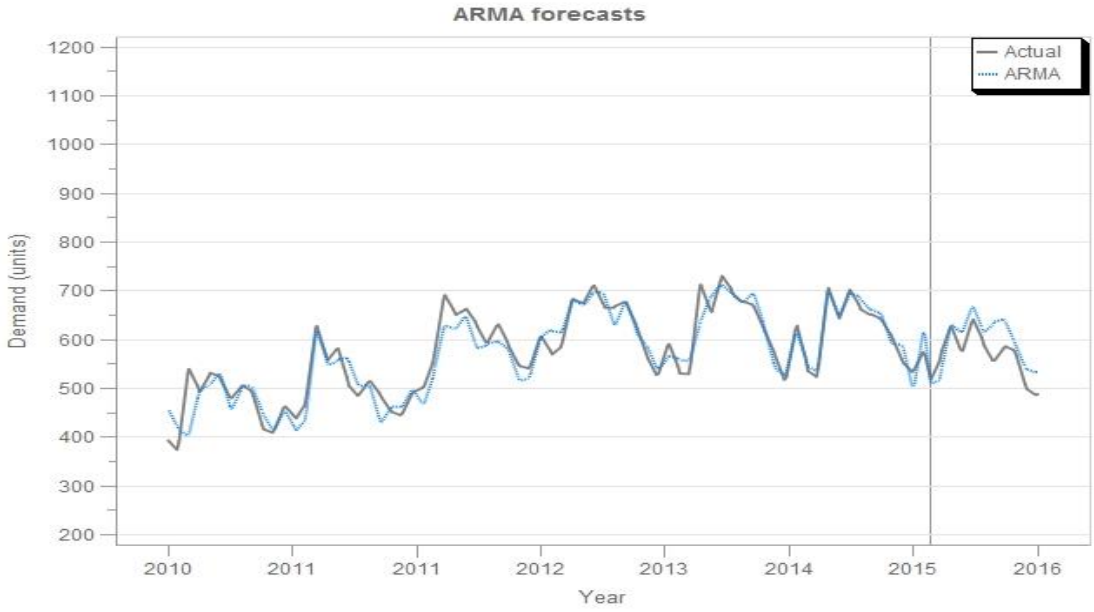

Figure 12. Fitted and forecasted values in SAS.

In terms of Root MSE, Python has the highest error (70.88), while MATLAB has the best fit (23.95). R is close (26.17) to MATLAB and SAS is the second weakest (32,12). However, root MSE in fitting data does not directly translate to forecasting accuracy. Python function takes more parameters than others so it was first estimated without trend and then with a trend component, latter one being labelled as *Python 2*. There is slight difference in forecasting accuracy between the software, as can be seen in a table 8 below.

Table 8. Forecasting accuracy of different software

	MATLAB		R		SAS		Python		Python 2	
	3-step	11-step	3-step	11-step	3-step	11-step	3-step	11-step	3-step	11-step
Root MSE	46,42	42,97	44,41	32,76	45,71	47,12	42,57	51,05	45,38	31,11
SMAPE	7,78	7,03	7,29	4,95	7,59	7,74	6,63	8,28	7,41	4,48

MATLAB and SAS provide very similar forecasts even though estimated constant differs a lot. R does a bit better job in forecasting while Python is both the worst and the best. Oddly changing a single parameter from *constant* to *constant plus trend* has a huge effect in Python, even though the estimated trend component is statistically insignificant and opposite to the long-term trend of the original data. This might be due to use of Kalman filtering, described in Makridakis & Wheelwright (1979)

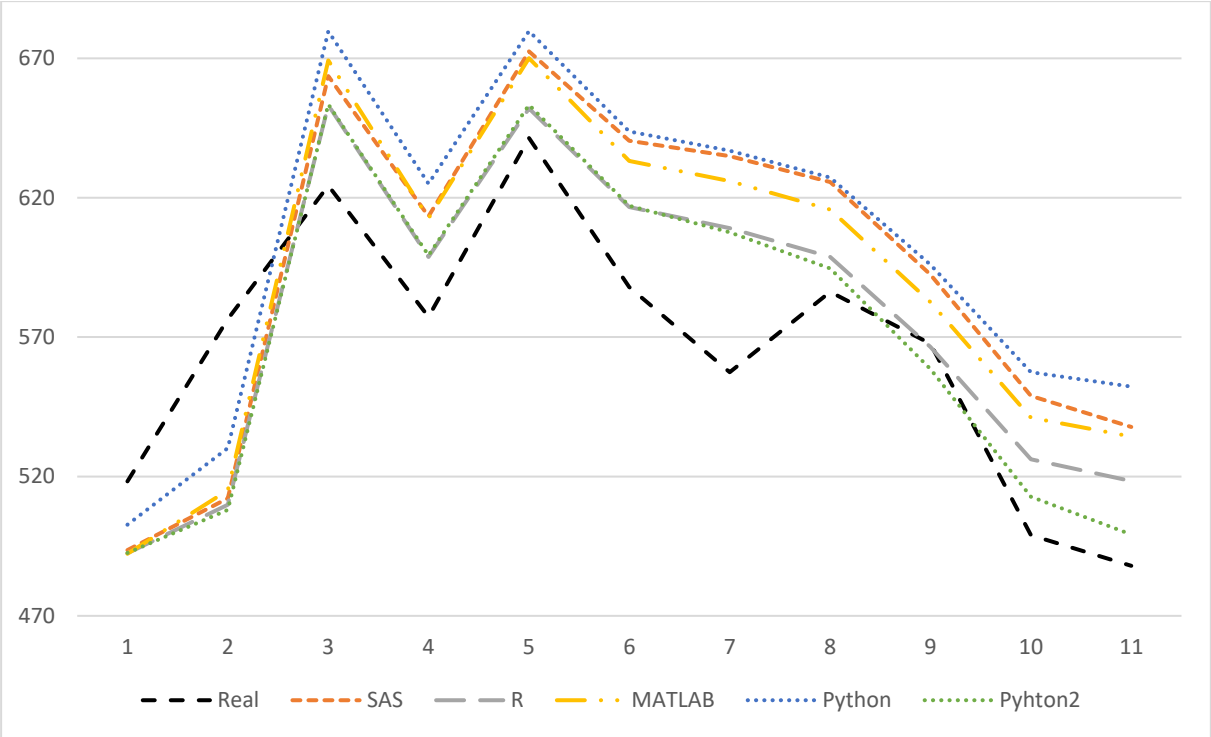

Figure 13. Comparison of forecasts from different software.

Makridakis & Hibon (2000) provided evidence that combining the results of several forecasting methods might generate more accurate forecasts. However, there is no general way to decide which forecasting methods should be included and what should be the weight vector of the system. One way of deciding the weights could be setting an equal weight for each method. Another possibility is running a multivariate regression on observed values with fitted values as independent variables.

Based on the demand data, five ensembles were created. Ensembles are forecasts generated by combining several individual forecasts from different forecasting methods. First of the ensembles combines linear time trend from 2010 to 2015 and ARIMA(13,0,0) model with 1st, 12th and 13th autoregressive lag. Forecasts from these two methods were added together and the resulting number was divided by two, i.e. combined forecast is the simple mean of these two methods. Second ensemble was created similarly to the first one, besides that the linear time trend present in the data was substituted with linear time trend in a larger window from 1980 to 2015. Third ensemble was created by running a linear regression model in MATLAB so that short-term time trend plus ARIMA(13,0,0) equals to observed values of the series. Fourth ensemble was similar to the third one, besides that the short-term time trend was replaced with the long-term time trend. Fifth ensemble was created by running a robust regression model with same parameters than the fourth one. Ensembles and their forecasting errors are presented in

table 9 below. RMSE 3 and SMAPE 3 are error measures for a 3-step forecast, RMSE and SMAPE are error measures for a 11-step forecast.

Table 9. Forecasting performance of ensembles

	Simple mean of short-term time trend and ARIMA(13,0,0)	Simple mean of long-term time trend and ARIMA(13,0,0)	OLS regression with short-term time trend and ARIMA(13,0,0)	OLS regression with long-term time trend and ARIMA(13,0,0)	Robust regression with long-term time trend and ARIMA(13,0,0)
Formula	(trend + arima) / 2	(trend + arima) / 2	13.629 + 1.0114 * arima - 0.0298 * trend	- 71.376 + 1.0114 * arima + 0.1177 * trend	- 66.612 + 1.008 * arima + 0.1113 * trend
RMSE	68.63	27.52	32.97	32.97	32.68
RMSE 3	41.75	28.91	44.99	44.99	45.10
SMAPE	10.51	4.00	4.99	4.99	4.93
SMAPE 3	6.34	4.18	7.41	7.41	7.42

When forecasting errors from ensembles are compared with forecasting errors in chapter 5.2, we can see that is possible to improve forecasting accuracy for the 11-step forecast using ensembles. First ensemble is poor, because the time trend changes as we approach validating set. Second ensemble is the most accurate for both the 3-step and 11-step forecasting horizons. This ensemble adds information not present in the learning set so it cannot be directly compared with all methods. However, it outperforms linear extrapolation of the long-term time trend which indicates that selection of the learning set influences forecasting accuracy and combining different forecasting methods might improve forecasting accuracy. Because the time trend can be seen as a scalar, it does not matter which time trend one uses in ordinary least squares regression, they both give near identical results. Robust linear regression improves forecasting accuracy a bit compared to simple regression, but it too gives too much weight to the ARIMA model. The difficulty of using ensembles lies in choosing the correct weight vectors. It is possible that 1/3 to 2/3 or some other arbitrary distribution of weight between time trend and ARIMA forecasts could further improve combined forecasting accuracy, but in general this cannot be reliably judged at the time of forecasting. Based on ensembles it seems that forecasting seasonal demand is possible using a rather short time series and that the accuracy of these forecasts could be improved by “guessing” the sign of the long-term time trend correctly. In practice this means that rather little data is required for identifying seasonality and short-term development. With an adequate guess regarding to the slope of the long-term time trend this data can be used to accurately forecast future development.

5.3 Stock prices

Stock indices are often described with random walk models. In previous research, ARIMA(0,1,0) model is generally used to describe random walk. (Pai, Lin 2005) ARIMA(0,1,0) takes the first difference of a unit root non-stationary series and estimates only

constant without autoregressive or moving average coefficients for different lags. Because the series is integrated of order one, estimated intercept represents drift element of the original series. Forecasts for differenced series may be represented as $\hat{y}_t = \mu + y_{t-1}$, where μ is the constant. This representation has an autoregressive coefficient of 1, which indicates indefinitely slow mean reversion.

At first, augmented Dickey-Fuller unit root test was conducted. Data was found non-stationary with a p-value of 0.54. To introduce stationarity, first difference was used. Differenced series is presented in figure 14. There are spikes of different height in the return series, which indicates that volatility clustering might be present. To see whether GARCH model would be applicable, one could perform heteroscedasticity test for GARCH effects, but conditionally heteroscedastic autoregressive models are part of master's education in LUT School of Business & Management so these models were not considered as something typical undergraduate students should know.

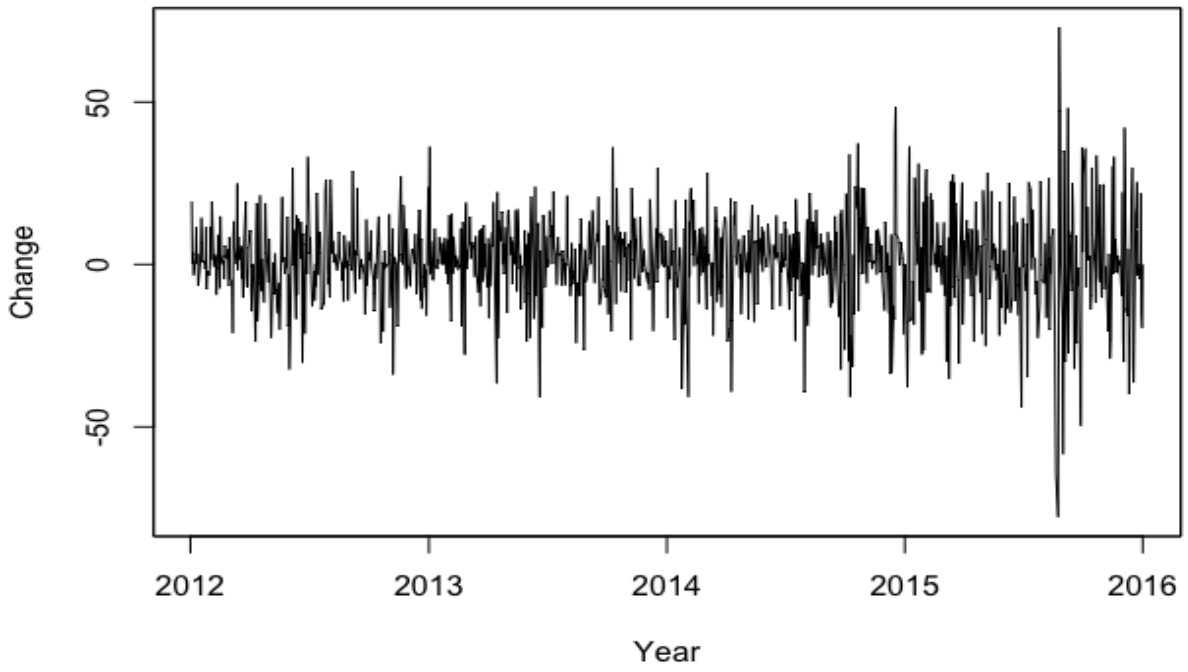

Figure 14. Return series of S&P 500 index

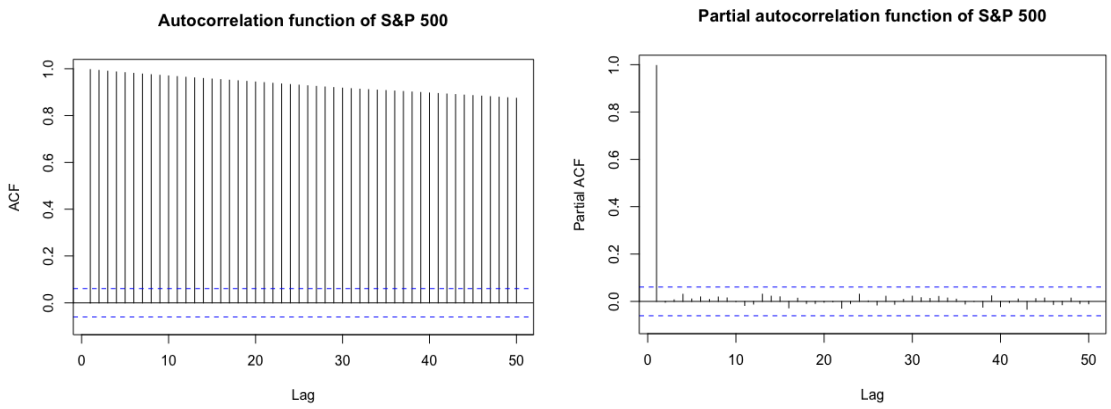

Figure 15. ACF and PACF of S&P 500

Autocorrelation function of S&P 500 index decreases very slowly, which is also an indicator of non-stationarity. After introducing stationarity by differencing, autocorrelation and partial autocorrelation funcitons were plotted again. These functions do not have a clear pattern and there are only few statistically significant lags. Neither ACF or PACF decays geometrically to zero. This supports the hyphothesis that the series follows random walk and thus ARIMA(0,1,0) is the only viable method.

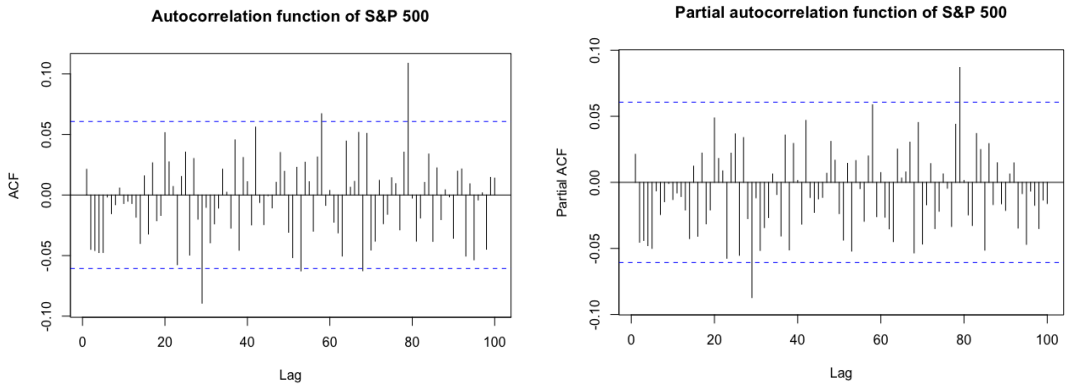

Figure 16. ACF and PACF of differenced S&P 500 index

Because naïve method and naïve method with drift are equivalents of ARIMA(0,1,0) models with and without the drift element, they are considered the best choice. Applying seasonal naïve is rather difficult, because one should choose whether the corresponding previous value is obtained by looking at the values last week, last month or last year. If the series follows random walk, there is no correlation between previous and future values and thus using a seasonal naïve method does not make much sense. For comparison, theta method and simple linear regression were included as well as Auto-ARIMA in R.

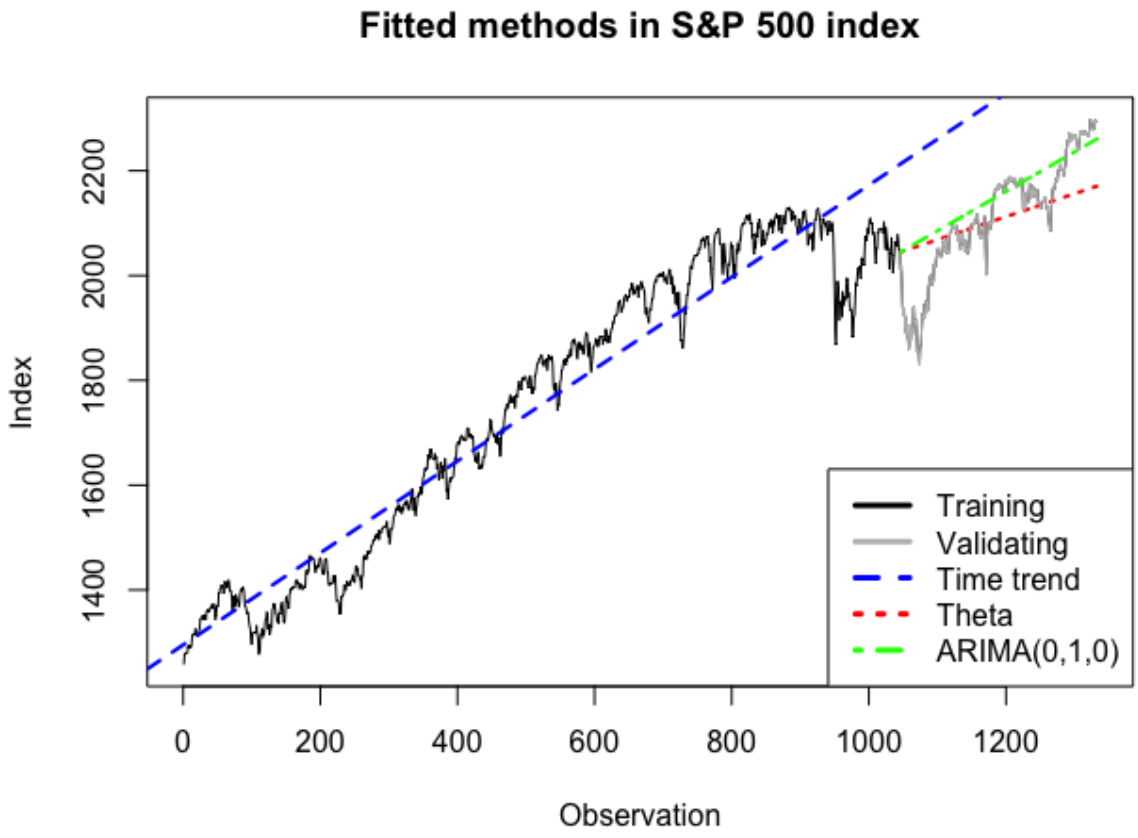

Figure 17. Comparison of fitted models

It is clear, that the time trend has changed recently. This change makes simple linear regression estimate too high values for the index. ARIMA(0,1,0), or in other words, naïve method with drift, takes linear trend present in the data and uses it to estimate future values based on the last observed value. It is not able to predict sudden drop in the series, because it is a form of univariate linear regression. The decomposition technique, or Theta method, proposed by Assimakopoulos & Nikolopoulos (2000) and identified as simple exponential smoothing with ½ drift by Hyndman & Billah (2003) takes a rather conservative approach to estimation by decreasing the slope coefficient to half of what the simple linear regression would suggest. For S&P 500 index at a given period, this method of estimation yields rather good results, as can be seen in figure 17. However, estimated smoothing parameter α for simple exponential smoothing part in theta method is 0.998989, which practically equals to one, indicating that the simple exponential smoothing forecast is identical to naïve method without drift. For the selected forecasting horizon, Theta method performs better than ARIMA(0,1,0) method only because the series suddenly drops. If the series would rise, theta method would give too low results compared to naïve method with drift.

Auto-ARIMA identifies the model as ARIMA(2,1,1), which converges rather quickly to the long-term time trend present in the series. First autoregressive lag has a coefficient of 0.9756 and second -0.0509. Moving average coefficient is -0.9652 and drift element 0.7428. With aforementioned parameters, this model behaves somewhat similar to naïve method with drift. Auto-ARIMA is presented in figure 18 below.

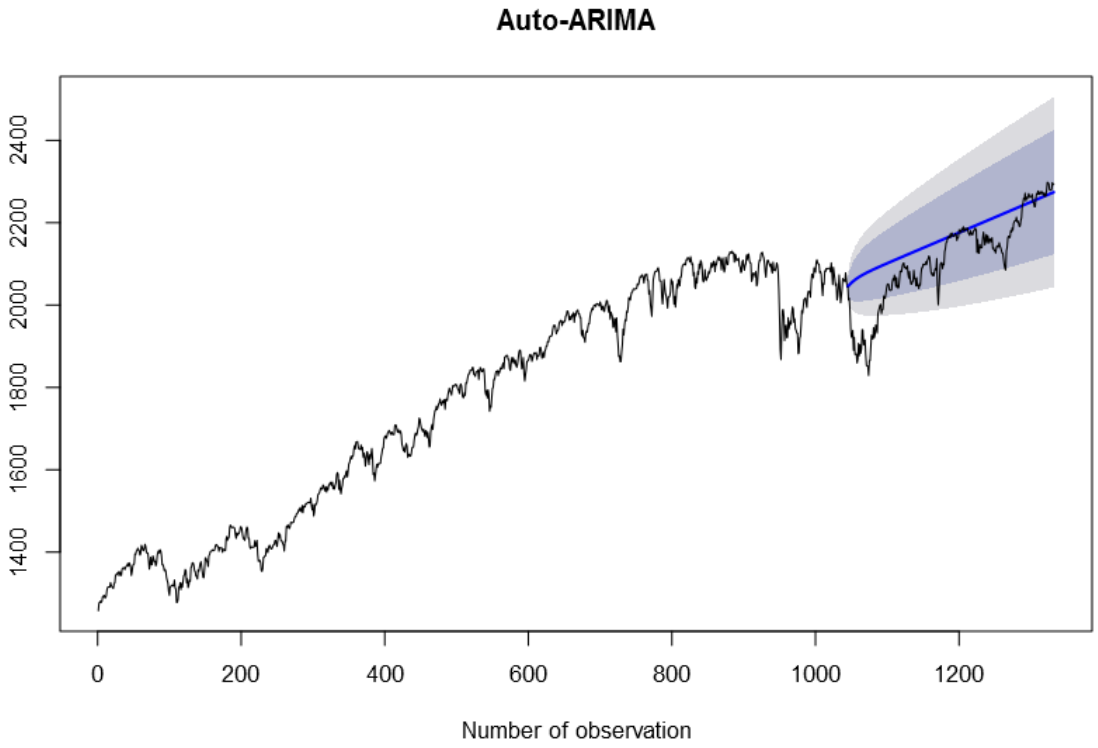

Figure 18. *Auto-ARIMA*

To see whether log transformation would improve forecasting accuracy, natural logarithm of the series was taken. The same ARIMA(0,1,0) model was then applied to transformed series. From figure 19 below, we can see logarithmic transformation with base *e* does not improve forecasting accuracy. It is not reasonable to calculate mean squared error for logarithmic and non-logarithmic series, because the measure of error depends on the scale. Logarithmic transformation has some advantages in econometrics, but the nonlinear nature of the log-transformation does decrease forecasting accuracy in this case.

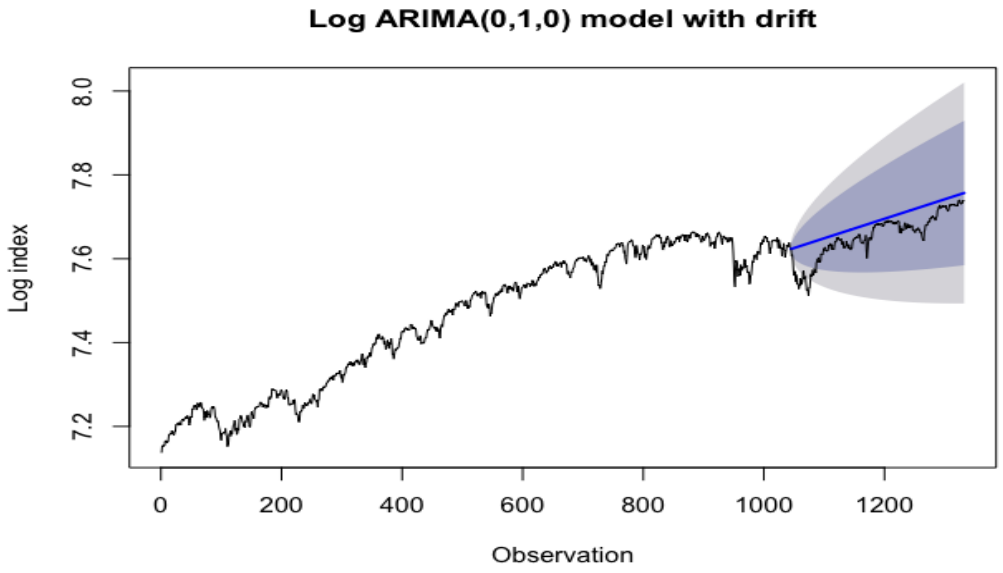

Figure 19. Log forecast for S&P 500

Finally, fitting and ARIMA(80,1,0) model using only the statistically significant 80th autoregressive lag was attempted. For this purpose, MATLAB and SAS are the best choice of software, because they allow defining the wanted autoregressive lags as integers in the input vector of autoregressive lags in ARIMA model. In Python and R one should fix the first 79 lags as zero either by looping over them or manually typing 79 zeros separated with commas or spaces before the wanted 80th AR lag. Although this ARIMA model is not a random walk model and thus not widely applicable, it could predict that the index would drop soon.

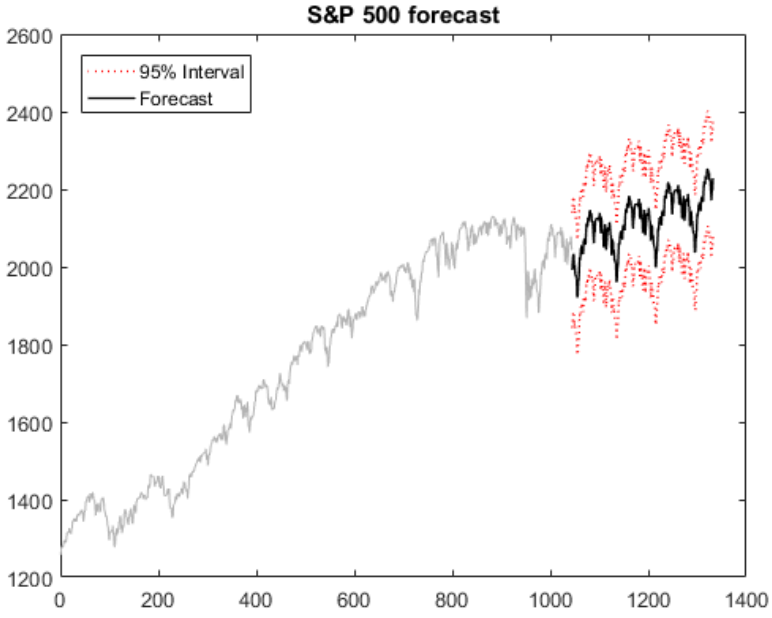

Figure 20. S&P 500 forecast using the 80th autoregressive lag

6. Conclusion

Autoregressive models are well suited for capturing seasonality in data. These models can be used for demand forecasting, at least for established products where the demand is more predictable and has clear patterns. Care should be taken in deciding the learning data, especially if the data is trending in some direction. A lot of forecasting methods will begin to fail if the trend suddenly changes. In demand forecasting, best results were obtained by combining the seasonality present in the recent observations with the long-term time trend. Running a linear regression model where autoregressive fit and a linear time trend are used as independent variables seems to eliminate the effects of the time trend. This approach gives too little value for the time trend and does not improve forecasting accuracy in tested series.

In stock markets, autoregressive models are suited for capturing linear trends. However, more complex methods are required for capturing nonlinear patterns. Ensembles, often called hybrid ARMA/ARIMA, have been built on ARIMA models to successfully predict stock market development e.g. by Pai & Lin (2005). These models often include neural network structure and support vector machines, which are beyond the scope of undergraduate education in econometrics. Perhaps, with a right choice of additional courses in business analytics or applied mathematics, these hybrid models could be used to improve forecasts in stock markets in master's thesis. Another way of trying to improve forecasting accuracy in stock markets could be using conditional heteroscedastic models, such as GARCH, which are part of master's course in advanced econometrics.

Overall, ARIMA models form a solid foundation to begin experimenting with forecasting economic time series. It cannot be stated that simple methods are better than ARIMA models, because these two groups overlap. For example, naïve method can be seen as ARIMA(0,1,0), simple exponential smoothing as ARIMA(0,1,1) and Holt's smoothing method as ARIMA (0,2,2) model (Hyndman, Koehler et al. 2008). Based on tested time series, if seasonality is present, correctly specified ARIMA models are an excellent choice for capturing seasonality in the data.

Of the available software, MATLAB does provide efficient tools for fitting an ARIMA model and forecasting future values with the estimated model. However, MATLAB lacks the tools for model identification, because it requires defining an ARIMA object of selected order before being able to fit the model. For identifying the order of the model, SAS is perhaps the most informative. Unlike other tested software, SAS automatically runs residual check for white

noise. Otherwise, SAS is the most inflexible and the most expensive. SAS might be the right choice for data analysis in large companies where the full power of SAS is utilized, but it is hard to justify the high annual cost in small to medium sized enterprises.

Since R was the only one able to estimate ARMA(14,1) model with only 72 observations, it seems that R can estimate a model with less observations than other tested software. This might indicate that it takes some shortcuts compared to MATLAB, SAS and Python. On the other hand, it is also possible that R has a better algorithm for fitting the model, because estimates from R are not that different from other tested software. R also has a function which selects the appropriate model automatically by minimizing the small sample corrected Akaike's information criteria. For forecasting univariate time series, R is the most versatile software mainly due to a great forecasting package written by an Australian professor of statistics and Editor-in-Chief of *International Journal of Forecasting* Rob Hyndman.

Syntax wise R is very intuitive and similar to Python. Both of the above software allow running one line of the code at a time with a hotkey, which is useful for several reasons. Firstly, it makes debugging a bit easier. Secondly, it makes it easier to make small changes to the code, because one does not have to run the entire program again after changing a small piece of code. Learnability of R is a bit better compared to Python, because R has only one namespace. In Python, each module has its own namespace and the user must remember the path of each function. This makes it easier for creators to name functions but decreases the memorability of the syntax for casual users.

Overall, R is the best choice for most users to begin experimenting with forecasting. It is free and easy to download. Unlike SAS, R runs on most platforms including Mac OS, Linux/Unix and Windows. The fact that SAS existed before personal computers and the currently used SAS code can still be run on 40-years-old mainframe machines is simply irrelevant for most users. It is true that MATLAB and Python have performance advantage over R, but the accessibility and low cost of R makes it a more attractive choice than MATLAB. Improved learnability of R due to simplicity and unified namespace makes it easier to pick up R compared to Python. After familiarizing oneself with the scripting in R, is not that hard to make transition from R to Python if the user outgrows R. Python is the most powerful of the tested software and hardest to outgrow. Yet it requires more knowledge of programming and has more complex data structures than the others. Basically, the creativity of the user is the only limitation to the power of Python, but the data mining capabilities of Python are simply too much for most

undergraduates. Furthermore, a combination of free and proprietary software packages may be used if institute provides license for a proprietary package. For example, Aboagye-Sarfo et al. (2015) used SAS for modelling and forecasting but opted to generate all graphs and plots using TSA (time series analysis) package in R project. For small samples, Microsoft Excel is also a great tool for generating graphs.

Bibliography

ABOAGYE-SARFO, P., MAI, Q., SANFILIPPO, F., PREEN, D., STEWART, L. and FATOVICH, D., 2015. Journal of biomedical informatics. *Journal of biomedical informatics,* **57**, pp. 62-73.

ACOCK, A., 2005. SAS, Stata, SPSS: A Comparison. *Journal of Marriage and Family,* **67**(November), pp. 1093-1101.

AKAIKE, H., 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control,* **19**(6), pp. 716-723.

ANDERSON, D., BURNHAM, K. and WHITE, G., 1994. AIC Model Selection in Overdispersed Capture-Recapture Data. *Ecology,* **75**(6), pp. 1780-1793.

ARMSTRONG, J.S., 1985. *Long-range forecasting: From crystal ball to computer.* 2. ed. edn. New York: Wiley.

ASSIMAKOPOULOS, V. and NIKOLOPOULOS, K., 2000. The theta model: a decomposition approach to forecasting. *International Journal of Forecasting,* **16**(4), pp. 521-530.

BOX, G., 1953. Non-normality and tests of variances. *Biometrika,* **40**(3/4), pp. 318-335.

BOX, G.E.P. and JENKINS, G.M., 1976. *Time series analysis - forecasting and control.* Rev. ed. edn. San Francisco [u.a.]: Holden-Day.

BROOKS, C., 2014. *Introductory econometrics for finance.* 3rd ed edn. Cambridge: Cambridge University Press.

DUNIS, C.L., LAWS, J. and KARATHANASSOPOULOS, A., 2012. Modelling and trading the Greek stock market with Hybrid ARMA-Nerural network models. *Financial decision making using computational intelligence.* New York: Springer, pp. 103-127.

GOLDFELD, S. and QUANDT, R., 1965. Some Tests for Homoscedasticity. *Journal of the American Statistical Association,* **60**(310), pp. 539-547.

HIGH SCALABILITY, 2011-last update, 6 Lessons from Dropbox - One Million Files Saved Every 15 minutes. Available: http://highscalability.com/blog/2011/3/14/6-lessons-from-dropbox-one-million-files-saved-every-15-minu.html [Mar 10, 2017].

HILL, R.C., GRIFFITHS, W.E. and JUDGE, G.G., 2001. *Undergraduate econometrics.* 2. ed. edn. New York: Wiley.

HURVICH, C. and TSAI, C., 1989. Regression and time series model selection in small samples. *Biometrika,* **76**(2), pp. 297-307.

HYNDMAN, R.J. and BILLAH, B., 2003. Unmasking the Theta method. *International Journal of Forecasting,* **19**(2), pp. 287-290.

HYNDMAN, R. and ATHANASOPOULOS, G., eds, 2013. *Forecasting: principles and practice.* Melbourne: OTexts.

HYNDMAN, R. and KHANDAKAR, Y., 2008. Automatic time series forecasting: The forecast package for R. *Journal of Statistical Software,* **27**(3), pp. 1-22.

HYNDMAN, R., KOEHLER, A., ORD, J. and SNYDER, R., 2008. *Forecasting with exponential smoothing - The state space approach.* Berlin: Springer.

KIRCHGSSNER, G., WOLTERS, J. and HASSLER, U., 2013. *Introduction to Modern Time Series Analysis.* 2nd ed. 2013. edn. Berlin, Heidelberg: Springer Berlin Heidelberg : Imprint: Springer.

LÜTKEPOHL, H., 2015. Forecasting unpredictable variables. *Empirical economic and financial research,* , pp. 287-304.

MAKRIDAKIS, S. and WHEELWRIGHT, S., eds, 1979. *Forecasting.* Amsterdam: North-Holland Publishing Company.

MAKRIDAKIS, S. and HIBON, M., 2000. The M3-Competition: results, conclusions and implications. *International Journal of Forecasting,* **16**(4), pp. 451-476.

MUENCHEN, R.A., 2011. *R for SAS and SPSS Users.* New York, NY: Springer New York.

NELLI, F., 2015. *Python Data Analytics.* Apress.

NEWBOLD, P. and GRANGER, C., 1974. Experience with Forecasting Univariate Time Series and the Combination of Forecasts. *Journal of the Royal Statistical Society,* **137**(2), pp. 131-165.

NIELSEN, J., 1993. *Usability engineering.* Boston: Acad. Press.

R FOUNDATION, 2017-last update, What is R?. Available: https://www.r-project.org/about.html [23.3., 2017].

SAS INSTITUTE, 2017-last update, About SAS. Available: https://www.sas.com/en_us/company-information.html#2010s [24.3., 2017].

SHEPPARD, K., 2017-last update, Introduction to Python for Econometrics, Statistics and Data Analysis. Available: https://www.kevinsheppard.com/Python_for_Econometrics [12.2., 2017].

THOMOPOULOS, N.T., 2015. *Demand Forecasting for Inventory Control.* 2015 edn. Cham: Springer Verlag.

TIOBE, 2017-last update, TIOBE Programming Community Index Definition. Available: http://www.tiobe.com/tiobe-index/programming-languages-definition/ [24.2., 2017].

TOFALLIS, C., 2015. A better measure of relative prediction accuracy for model selection and model estimation. *Journal of the Operational Research Society,* **66**(8), pp. 1352-1362.

TSAY, R.S., 2005. *Analysis of financial time series.* 2nd ed edn. Hoboken (NJ): Wiley.

VANCE, A., 2009-last update, Data Analysts captivated by R's power. Available: http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?pagewanted=all [Feb 12, 2017].

WATSHAM, T.J. and PARRAMORE, K., 1997. *Quantitative methods in finance.* London: Thomson Learning.

WIKIPEDIA, 2017. Comparison of statistical packages. Available: https://en.wikipedia.org/wiki/Comparison_of_statistical_packages [Feb 10, 2017]

WHITE, H., 1980. A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity. *Econometrica,* **48**(4), pp. 817-38.

Appendix

Appendix 1. Timing procedures

MATLAB

```
tic

r = randi([-1500 1500],5000, 5000);
A = r' * r;
ans = inv(A);

toc
```

SAS

```
%let start = %sysfunc(datetime());


proc iml;
seed = j(1500,1500,0);
x = uniform(seed);
A = t(x) * x;
y = inv(A);

%let end = %sysfunc(datetime());
%let runtm = %sysfunc(putn(&end-&start,12.4));
%put It took &runtm seconds to run the program;
```

Python

```
import time
import numpy
start = time.perf_counter()

r = numpy.random.randint(-1500,1500,(1500,1500))
A = numpy.matrix.transpose(r) * r
ans = numpy.matrix.getI(A)

end = time.perf_counter()
print("It took %5.4f seconds to run the program" %(end-start))
```

R

```
start <- proc.time()
r <- matrix(sample.int(1500, size = 1500*1500, replace = TRUE),1500,1500)
A <- t(r) * r
ans <- solve(A)

print(proc.time() - start)
```

Appendix 2. ARIMA procedures

MATLAB

```
Mdl = arima('ARLags',[1 2 3], 'MALags', [1], 'SARLags', [1], 'SMALags',
[1], 'Seasonality', 12);

Est_Mdl = estimate(Mdl, Cars);

[Y,YMSE] = forecast(Est_Mdl, 11, 'Y0', Cars);

lower = Y - 1.96*sqrt(YMSE);
upper = Y + 1.96*sqrt(YMSE);


figure
plot(Cars, 'Color', [.7,.7,.7]);
hold on
h1 = plot(72:82,lower, 'r:', 'LineWidth',1);
plot(72:82,upper, 'r:', 'LineWidth',1);
h2 = plot(72:82,Y, 'k', 'LineWidth',1);
legend([h1 h2],'95% Interval', 'Forecast', 'Location', 'northwest')
title('US monthly car demand forecast')
hold off
```

SAS

```
proc arima
data=sami.US_cars_2010;
identify var=Cars(1,12) nlag=15 stationarity=(ADF=(4));
run;

estimate p=(1 4 6) q=1;
run;

forecast lead=11 interval = month id = Date out = results;
run;
```

Python

```
import statsmodels.tsa.arima_model as arima
import statsmodels.tsa.stattools as stattools

model = arima.ARMA(Cars, order = (14,1))
fit = model.fit()
print(fit.summary2())
prediction = fit.predict("15/1/2016", "15/11"2016")
```

R

```
library(forecast)
```

```
#----------- Autofitting the model -------------------#

fit <- auto.arima(Cars)

fcast <- forecast(fit,11)

plot(fcast)

#---------- Manually fitting the model ---------------#
adf.test(Cars)
a <- Acf(Cars)
p <- Pacf(Cars)
plot(a)
plot(p)

final.aic <- Inf

final.order <- c(0,0,0)

for (i in 0:14) for (j in 0:12) {

  current.aic <- AIC(Arima(Cars, order = c(i,0,j)))

  if (current.aic < final.aic) {

    final.aic <- current.aic

    final.order <- c(i,0,j)

    final.arma <- Arima(Cars, order = final.order)

  }

}

fcast2 <- forecast(final.arma, 11)
plot(fcast2)
```

Appendix 3. Demand forecasts form Theta vs. ARIMA