Department of Information Systems
University of Hamburg
Germany

Comparing R and Python languages on the example of an econometric-statistic task

A thesis submitted for the degree of
Bachelor of Science in Information Systems

by

Alisa Dammer
Matriculation number: 6325479

Primary Supervisor: Prof. Dr. Stefan Voss
Secondary Supervisor: Dr. Stefan Lessmann

Abstract

Along with a very rapid digitalization of the world and according software and hardware development, it became possible to accumulate immense amounts of data for further analysis. However, classic statistical methods and tools appeared to be inefficient for Big Data. This is why highly skilled data scientists, data analysts and data engineers are in demand now.
There are several highly demanded skills required from a data analyst. A deep understanding of high performance programming languages is a requirement for data analysts. The most highly demanded programming languages are Python and R. The languages have different concepts and are designed for different primary purposes.
This thesis aims to answer the question which language is more suitable for a beginner data analyst. In order to answer this question two things were done: Firstly, an evaluation criterion table was suggested. Secondly, the author has implemented a medium-complexity program in both Python and R to evaluate the results accordingly to the estimation table.

Contents

1 Motivation

1.1 Introduction: The Rise Of The Data Analysis

An analysis of any available data has always played an essential role for business success. According to Wikipedia: "Since the mid-1990s, the Internet has had a revolutionary impact on culture and commerce, including the rise of near-instant communication by electronic mail, instant messaging, voice over Internet Protocol (VoIP) telephone calls, two-way interactive video calls, and the World Wide Web with its discussion forums, blogs, social networking, and on-line shopping sites" [1]. The mass digitalization of the world has allowed for easier and faster data collection. With the advancement of technology the amount of data stored jumped from a 3.75MB disk presented by IBM in 1956 [2] up to a 16TB solid-state drive produced by SAMSUNG [3]. Many different classical econometric and statistic tools became unusable for the amount of the data collected. This happened because the methods were not optimized for such a big amount of information. Most of the algorithms were sequential as opposed to parallel and thus their performance suffered.
With the rise of opportunities to collect and store the data, a new sphere of analysis appeared to be important for business: Big Data was known long before it became popular. In 1944 the first estimate of data growth was given by Fremont Rider [4]. In his work, Rider pointed out the problem of information growth: "[...] of all the problems which have, of recent years, engaged the attention of educators and librarians, none have been more puzzling than those posed by the astonishing growth of our great research libraries... it seems, as stated, to be a mathematical fact that, ever since college and university libraries started in this country, they have, on the average, doubled in size every sixteen years". Although the main concern of the paper was directed to storing and maintaining science papers in the libraries as well as the respective growth in administration costs, these were already the questions from the Big Data domain. Another prominent work on Big Data is a paper called "Little Science, Big Science... and beyond" by Derek Price written in 1963 [5]. In his work, D. Price discussed the evolution of science institutions from their state to multi-purpose clusters of different libraries along with the data growth. This work shows the development of the infrastructure along with data growth and data transformation.

The new level of the accessible data required both new hardware and software technologies as well as new mathematical approaches. Rapid storage systems development started in the 1960s. New software was developed to take advantage of the new hardware as it became available. One of the first papers using the term 'Big Data' was written in 1999 by Steve Bryson, David Kenwright, Michael Cox, David Ellsworth, and Robert Haimes. The paper was called "Visually exploring gigabyte data sets in real time" [6]. It is believed that the term itself at the current understanding was presented by Roger Magoulas [7].
Starting from the mid 2000s the need for experienced data analysts, data scientists and data engineers grew steadily. However, the candidates should have a certain set of skills for working with huge data sets on a distributed file system like Google distributed file system (GDFS), Hadoop distributed file system (HDFS) or other similar products. The "2015 Data Science Salary Survey" from O'Reilly sums up the current trends in the required skill set for data processing positions [8]. The key points made in the survey show that SQL, Excel, R, and Python are highly demanded skills. The same results appeared by browsing through the three big job search portals for data engineer, scientist and analyst positions. Gigajob, Monster and LinkedIn (not exactly job searching platform, but it has this functionality and results are representative) show similar results to the O'Reilly survey. The most interesting trends are:

1. The rise of interest towards Python, R and Scala (Spark).
2. Java, C++ and Haskell are demanded by specific industries (financial sectors, heavy industries).
3. Hadoop with Hive or Pig is giving up it's position to the Spark language.
4. SQL knowledge is substantial.
5. Excel knowledge is substantial.

We consider the rise of Python and R popularity a very interesting trend since both languages were designed for the data processing but weren't popular among the industry and business in the early 1990s. These two languages can be seen as substitutes or supplements depending on the task context. In this work we will try to answer the question that is: "What language would be better to start with for a beginner data analyst?". We have taken data analyst position since this position perform a variety of tasks related to collecting, organizing, and interpreting statistical information but is not responsible for building new data processing tools or creating new approaches to detect patterns in the amount of data. The tasks for a beginner analyst are considered to be data collecting and storing, data cleaning and processing, hypothesis testing and visualising the results for further interpretation and use. These tasks require basic programming skills to acquire the data from the web or database. Also, programming skills must be sufficient to visualize the data and port the graphs to the web. Another required skill is the understanding how DFS works and ability to use it. Additionally the candidate should have sufficient knowledge in mathematics and statistics.

1.2 Python language overview

Python is a relatively young language. The first public release of the language was announced 24 years ago. According to the official web page in Wikipedia: "Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages like C++ or Java" [9].
Currently, Python is widely used in both business and science. There are different application domains for the language like software development in games, management systems and financial sector [10]. An interesting example of Python use is ForecastWatch [11]. This projects helps evaluating the accuracy of weather forecasts. The project is implemented solely in Python. This language was chosen because of existing parsing and statistical libraries as well as back end and front end technologies. The ability of Python to easily work with C++ and C made this language the choice for Industrial Light & Magic (ILM). The company uses special software to create visual special effects for movies. Their effects were used in such films like Stark Trek, Avengers, Transformers, Indiana Jones series and many others [12].
The scientific community also uses Python for different purposes: from simulation software to direct programming. The example of such language use the Molecular Modelling Toolkit [13]. The scientific publications connected with this open source library can be divided into three categories:

1. Articles about the methods extending MMTK.
2. Articles about the software built using MMTK.
3. Articles about the research using MMTK.

A good example of an article from the first group is "Harmonicity in slow protein dynamics" by K Hinsen, A J Petrescu, S Dellerue, M C Bellissent-Funel, and G R Kneller [14]. The example article from the second group is the "Analysis of domain motions in large proteins" by K Hinsen, A Thomas, and M J Field published in 1999 [15]. An example of

articles with results gotten with the help of MMTK is "A Molecular Dynamics Investigation of Vinculin Activation" by Javad Golji and Mohammad R Mofrad published in 2010 [16]. There are many other scientific spheres using Python for research. For example in neural networks and deep learning Python libraries and programs are used heavily. The "Deep Boltzmann Machines and the Centering Trick" by Gregoire Montavon and Klaus-Robert Müller in 2012 [17].
Python offers a variety of tools for both researches and business-oriented projects. The one-purpose programs and full stack web applications are built using solely Python or in a combination with other programming languages.

1.3 R Overview

Similarly to Python, R is a young language. The language was first introduced in 1993. According to the Wikipedia page: "R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis" [18]. From the description it can be seen that R was designed explicitly for scientific purposes as well as for data analysis. The community behind this language consists mostly of universities or scientific teams.
R programs allow for faster and easier prototyping in comparison to Java, C++ or C. The performance of the prototypes written in R is better than the performance of similar Python prototypes. This is why the modern trend is to use R to build an initial and simple prototype or to use R code as a part of a program solving one or more analytical tasks. For example, R scripts can be called inside of Python programs for speeding up certain computations. However this trend is new and the number of R programs used in business projects is relatively low.
As stated earlier, R is mostly used for statistical analysis by the scientific community. Similarly to the Python articles, scientific papers with R usage can be divided into two categories:

1. The articles about R extensions and packages.
2. Articles using R programs for hypothesis checking.

A good example of an article from the first category is "lavaan: An R Package for Structural Equation Modeling" by Yves Rosseel [19]. The package can be useful for social and psychological analysis. In general, every package developed by a scientific team from a university can be seen as a scientific engineering work. In this paper we

have used several such packages: Tseries [20] and Nlme packages [21]. These packages will be discussed in details in the R section. R is used not only in statistics, but also in biology. Analysis of Phylogenetics and Evolution (APE) is a package written in the R language for use in molecular evolution and phylogenetics [22]. An example of a scientific paper from the second category is the article "neuralnet: Training of Neural Networks" by Frauke Günther and Stefan Fritsch [23]. In their article the authors have used the data from the R distribution and wrote a multi-layered perceptron trained via supervised algorithm in the context of the regression analysis. Another interesting example of the R usage is the paper "Nonlinear Regression and Nonlinear Least Squares in R" by John Fox and Sanford Weisberg [24].
Although R is still mostly a scientific language, it is used as a part of the analytical systems due to its performance and simplicity.

1.4 Python vs R wars

As we have already mentioned above, both Python and R are widely used for scientific researches and business projects. Python is used more often for business projects since it is a multipurpose language with a low initial time investment. R requires a bit more initial time investment but yields better performance.
For a person without previous experience in programming it can be hard to decide what language to start with. The discussion of the question whether Python is more suitable for analysis than R takes place since the data analysis boom and still no objective answer was given. All discussions can be roughly divided into three categories:

1. General comparison.
2. Comparison of the languages based on a benchmarking.
3. Comparison based on the code differences for the same result.

An example of the general and mostly subjective comparison is the discussion on specialized forums. Most known sources for analysts are Biostars [25], different sub domains of stackexchange [26] and specific questions on stackoverflow [27]. The most popular opinion on the forums is to use whatever suits the task and current set of skills. The users of the biological statistics forum on average say that R is well-maintained and that is has more suitable packages, however they use Python for their analytical purposes. The reason for that conflict is the multipurpose nature of Python, which allows better data collection and direct visualization in web. All of this makes Python easier to use and eliminates the bottleneck among the parts of the project. On the other hand R, has

numerous smaller packages for analytical tasks and is faster than the average Python program. The obvious disadvantage of the general comparison approach is its biased nature. The results of such discussions are hardly usable.
Benchmarking is a good criterion to compare the performance of the languages. However, this method should be used carefully. The performance itself will only show how much faster one language can be for the specific task, the specific machine and specific implementations for languages that are compared. In order to use benchmarking all parameters should be set equally. An interesting example of a benchmarking is given in the paper "A Comparison of Programming Languages in Economics" by S. Boragan Aruoba and Jesus Fernandez-Villaverde [28]. In their paper the researchers have compared several languages implementing the stochastic neoclassical growth model. Among the languages they have chosen were R and Python (CPython and PyPy). All programs were implementing the same algorithm and run the same number of iterations. The programs were implemented on the level of an experienced programmer. The paper yields predictable and unbiased results. However the complexity of the task and according complexity of the programs written play an important role in end results. The languages have different paradigms, concepts and structure, thus there is no guaranty that the chosen code style results in optimal performance for each language.
The third group of comparison methods is comparing the complexity of the code in order to achieve same performance. An interesting example of this approach is the article written in the Dataquest blog [29]. In the article the authors have tried to reach same results using Pandas and NumPy packages for the Python program and base R package for the R program. The Dataquest team suggests R to be a more suitable language for purely statistical tasks, while Python being easier to use for complex systems. The answer the authors give can't be sconsidered to be clear and final.
The goal of this paper is to answer the question: "What language is more suitable for a beginner data analyst?" To answer that question we will need to create criteria to evaluate the languages in the most unbiased way. The criteria will depend on the tasks common for a data analyst.
The common tasks of a data analyst are data mining and data visualization. The possibility to use DFS to process the data can be seen as a big advantage and thus is considered to be important for evaluation criteria. A data analyst should have a sufficient mathematical background to understand and use different algorithms and approaches, but not necessarily be able to implement them (that is the task of a data scientist).

1.5 Thesis Description

For our work we have chosen Python and R. Both languages are widely used by data scientists, engineers and analysts across all possible domains. The popularity of this languages is rising steadily. Both Python and R offer significant amount of packages for projects of different complexity. However since these two languages are different and are substitutes at the same time, we will try to answer the question what language is more suitable for beginners.
Comparing two different programming languages is a hard task. In order to get unbiased results we will set several parameters. First, the domain of the language use. In this paper we will write a program like a beginner data analyst probably would. The main focus is a program solving a medium complexity statistical task, running within acceptable time without expensive speed up methods. In other words we will compare Python and R based on the programs written in each language to find the best regression from a given data table.
To evaluate the languages objectively we will consider the criteria listed in the table:

1. Performance.
2. Memory use.
3. Big Data tools.
4. The quality of the documentation.
5. The quality of the ecosystem of the language.

Each program will get score between 0.0 to 1.0 for each criterion.
As we have seen earlier, the comparison of the programs written on advanced level can be biased since the languages are different and the same approach can be beneficial in one case but not the other. This is why the idea of this bachelor thesis is to test which of the languages is more suitable for a specific medium complexity statistical task. Both programs will have similar structure and functions, so that they can be evaluated more properly. The program that will be evaluated consists of four parts:

1. Getting and formatting data - the program should be able to read a prepared file in csv format. Afterwards, the program should format the data into needed structure for further use.
2. Analyzing data - the program should be able to run statistical tests to figure out several data characteristics. This step is implemented in order to determine

what regression models are allowed for this data set. The following tests will be presented: stationary test, building CDF and KDE, finding moments, distribution test (goodness of fit tests).

3. Building a model - the program should be able to create a proper model using a step-forward algorithm. The preparation step is creating a correlation vector and leaving out all companies with the correlation coefficient smaller than 30%. As the first step the limits on the number of the predictors in the final regression will be set. The second step is to create a one parameter model using the correlation vector (take the company with the highest correlation coefficient). The third step is iterative model building:
   a) Create all possible double combinations consisting of fixed company name from the previous step and»> not used company name. Choose the best multiple linear regression built with GLS method among the class.
   b) Compare two models from different classes using LLR test.
   c) If the smaller model is better the search is over and the program returns the result.
   d) If the bigger model is better the iteration continues until the smaller model will be better or the limit on the parameters in the regression is reached. The program then returns the result.

   The output of the program shows the names of the companies in the final regression, main statistics, coefficients.

4. Visualization - the program should be able to present the results in the form of graphs for the needed steps.

5. The program is able to build predictions for the chosen company and evaluate them using the mean accumulated error (MAE) and or root-mean-square error (RMSE).

The task for the program is to build a best matching linear regression (if possible) for a company (always the first column in the file) using several limitations on the number of the predictors.
The program will build a simple regression (if possible). In this work we have considered the class of linear regressions to be a simple regression model. This task is considered to be a medium complexity task. The beginning programming level will allow us to avoid diving into specifics of the languages. So we will see the advantages and the disadvantages of the languages when used for a beginner task.
We will not consider data collection as a part of our main task. This is why the learning

set and validation data set will be provided upfront. The data will be chosen that way, so that results can be automatically checked or checked manually using Excel. This option will be used only if the results of the programs will look suspicious. The data will be prepared and provided by two files: training set and test set in the csv format. According to the list for objective evaluation, each point will give a score from 0 to 1 to the language. A score of 0 will be given in the negative case and a score of 1 will be given in the positive case.

1. Performance comparison will give 1 point to the fastest program and the language accordingly and 0 points to the slowest program.

2. Memory use comparison will give 1 point to the program and the language accordingly that allocates the least amount of memory and 0 points to the other program.

3. The possibility to use the language with a Big Data tool or to build a tool of that scale will bring the language with the most possibilities 1 point and 0.5 points to the second language.

4. Documentation and the ecosystem.

In order to give an accurate estimate to the documentation and the ecosystems of the chosen languages we have developed an additional criteria table. The accuracy of the estimate is needed to keep our general estimation unbiased and accurate. The quality of the documentation and the ecosystem plays an important role for the project of every complexity. Each language will get from 0 to 1 point for each criteria described in the table:

1. Documentation has an intuitive navigation.

2. A sufficient amount of examples is given within a library or a package.

3. The source code is opened.

4. The additional information is provided for complex functions and approaches.

5. Acceptable style of the documentation.

6. Community size and quality.

7. Diversity of the packages.

All these criteria cover the main issues like readability of the code and documentation, sufficient and clear information about the code and its use, possibility to get help and the reliability of the code quality. Several criteria like style, examples and navigation through the libraries are mostly important for a beginner programmer. The score system should be sufficient to objectively evaluate the documentation and the ecosystem of the languages from the beginner data analyst stand point.
The results of the programs will be also compared. If they differ, the models will be cross-compared. The program and the language accordingly will get 1 point for the accuracy relative to the accuracy of the second program. The language with the highest total score will be considered more suitable for the beginner data analyst.

1.6 Data preparation

As we have already said, the data for the processing will be prepared in advance. To be able to subjectively estimate the results, we will have to take transparent dependent variable and transparent explanatory variables. The connections between variables should be easily explained with a plain logic. For our purpose we have decided to take Intel as a dependent variable in the regression. Predictors will be chosen among the companies from the same market of micro chips, supplier market and customers market. We have chosen Intel because it is a transparent diversified company with clear development trend this company is one of the leaders of the microchips market and all connected markets are also easy to define. For this paper we will consider only a small amount of companies influencing Intel, since we want to keep the task on a medium complexity.
Intel's competitors can be found on Wikipedia listed in several tables for years from 1998 to 2013. We have used a parser to get the following list of manufacturers for the years 2000-2013 for semiconductors market:

```
['AMD', 'Qualcomm', 'Micron Technology','Hynix',
'Infineon Technologies', 'Intel Corporation',
'STMicroelectronics', 'Texas Instruments']
```

Semiconductors are the basic component for different devices, so the potential consumer-markets may vary. In this bachelor thesis we will concentrate on microchips (CPUs) consumers. The following consumer markets are: tablets, smartphones, personal computers, automobiles, video game consoles, medical technologies, engineering technologies, aviation. Tablets and personal computers are united into one market-group.
Automobile market:

```
['Toyota', 'GM', 'Volkswagen', 'Ford', 'Nissan',
'Fiat Chrysler Automobiles', 'Honda', 'PSA', 'BMW',
'Daimler AG', 'Mitsubishi', 'Tata', 'Fuji']
```

The only significant players (manufacturers) on the gaming console market are: Microsoft, Sony and Nintendo.
The aviation market is presented by the following companies:

```
['Boeing', 'United Technologies', 'Lockheed Martin',
'Honeywell International', 'General Dynamics',
'BAE Systems', 'Northrop Grumman', 'Raytheon',
'Rolls Royce', 'Textron', 'Embraer',
'Spirit AeroSystems Holdings Inc.']
```

The next microchips consumer markets are smartphones, tablets and PCs markets. Since many companies are presented on the markets mentioned above and have further production markets, we will put them into "Diversified" category. The diversified companies that may influence Intel are:

```
['Samsung', 'Apple', 'Microsoft', 'Nokia', 'Sony', 'LG',
'Motorola', 'Lenovo', 'BlackBerry', 'Alcatel', 'Vodafone']
```

Another huge consumer market for Intel is the medical equipment market. The following companies present this sphere:

```
['Johnson & Johnson', 'General Electric Co.', 'Medtronic Inc.',
'Siemens AG', 'Baxter International Inc.',
'Fresenius Medical Care AG & Co.', 'Koninklijke Philips',
'Cardinal Health Inc.', 'Novartis AG', 'Stryker Corp.',
'Becton, Dickinson and Co.', 'Boston Scientific Corp.',
'Allergan Inc.', 'St. Jude Medical Inc.', '3M Co.',
'Abbott Laboratories', 'Zimmer Holdings Inc.',
'Smith & Nephew plc', 'Olympus Corp.', 'Bayer AG',
'CR Bard Inc.', 'Varian Medical Systems Inc.',
'DENTSPLY International Inc.', 'Hologic Inc.',
'Danaher Corp.', 'Edwards Lifesciences', 'Intuitive Surgical Inc.']
```

Additionally to the companies mentioned above, several big players from the industrial equipment market will be added. These are the following companies:

```
['ABB Robotics', 'Adept Technology', 'Bosch', 'Caterpillar',
'Denso Robotics', 'Google', 'Universal Electronics']
```

After limiting the number of the connected markets, we have limited the number the companies from the chosen markets, because of the lack of information about some companies. The reason for the lack of information is that some companies have entered the international stock exchange recently (since 2010 earliest). Some companies are still

closed to foreign investors (which is the case for giants such as Samsung, Honda and other Asian companies). The last limit on the chosen companies is the trading volume. If the trading volume for the last two years was zero, the company was considered to not have been traded on the stock exchange.
After the companies were chosen, the whole data table was split into two data sets: learning set and validation set. We have obtained daily prices from 2006 to 2015 (31.12.2014 is the last date for all indexes). The training or learning set will be approximately 70% of the whole data volume: from 2006 to 2010. The validation set is about 30%: from 2011 to 2012. For this work we use the daily frequency (only opening prices).
The LearningSet file is a csv table containing 78 companies with 1240 prices for each company. The total size of the table is approximately 500KiB. The TestingSet csv file is also a table containing 78 companies with 515 everyday prices. The TestingSet file size is approximately 200KiB.

2 Python

2.1 The structure of the program

The program consists of three functional classes, connected with each other in the main class. The first class is called DataFormating and it is responsible for getting the data out of the csv file and writing it into an instance of a proper format for the other two classes. The dependent variable will be extracted from the data set (the dependent variable is always the first company listed in the first column in the csv table).
The second class is called StatisticTests. It runs several tests to find out the main static characteristics in order to choose an appropriate model class for the data. In this work we have focused on the linear regression class.
The third class is called BuildModel. This class builds the multiple linear regression (according to the results of the StatisticTests class) using the step-forward approach:

1. The amount of companies taken into account is reduced using the correlation vector: all companies having the correlation coefficient less than 30% are omitted.
2. A limit on the maximum number of the parameters is set using the following rule: 1 company out of 10 if the number of companies exceeds 5.
3. The first step is to find the best one-parameter model using the highest correlation coefficient from the first step.
4. All possible combinations for fixed companies from the previous step and remaining companies are created. For all these combinations the linear regression using Generalized Least Squares approach [24] is built.
5. The best model among the class is chosen using the AIC criterion.
6. Best models among the classes are compared using the likelihood ratio test.
7. If the bigger model is better and the limit of the predictors in regression is not achieved, the new small model is set to be equal to the old bigger model. The whole computations are repeat from step 4. The final result is equal to the last bigger model.

8. If the smaller model is better or the maximum number of parameters in the final regression is achieved then the final result is either the small model or the last big model.

At the end the program returns the names of the companies in the final regression and the full information for the best model.

2.2 The process description

In this section we will describe the language specific difficulties we have encountered.

2.2.1 Libraries Used

In the course of writing the program several steps were implemented. These steps are: extracting and preparing the data for further use, checking statistic characteristics and building the model based on the results of the statistics. In order to build the program three main libraries were used:

1. Python Standard Library [30].
2. SciPy (unites six different libraries) [31].
3. Statmodels [32].

The functions and the data structures defined in the standard Python library were used for every simple task, except some mathematical computations, since the NumPy library offers faster implementations.
The second most used library in this work was NumPy. This library is the part of the SciPy source. NumPy offers many fast multi-dimensional computations and associated multi-dimensional structures. In our work we have used this library to compute several statistics and to build the data for displaying graphs. The functions in the standard library did not show the sufficient performance needed for our program.
The third important library used in this work is Pandas. The library is a part of the SciPy source. Pandas provides high-performance data structures and associated functions for data analysis. This library can be used for generating, accessing and formatting data. In this work we have used it to get the data from our csv file in an intuitive manner and also to avoid an unnecessary formatting step. Pandas offers the functionality to get the data directly from web sources like Yahoo Finance, Google Finance, Google Analytics and other similar sources.
The library for drawing graphs is called Matplotlib. It is also a part of the SciPy source.

Matplotlib offers different types of graphs and formatting tools. The following example represents the default graph without any additional formatting.

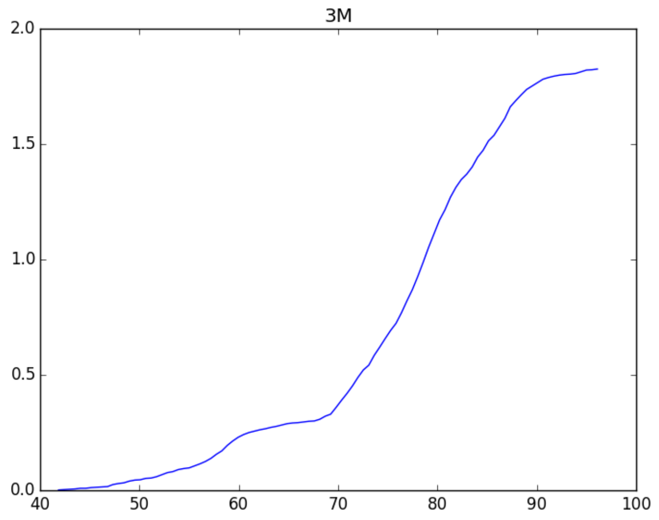

Figure 2.1: A basic plot built using Matplotlib library

The last library is called Statmodels. This library offers a variety of tools and models for statistic purposes. The main functions of the library used in our program are Augmented Dickey-Fuller test and building a multiple linear regression with generalized least squares approach.

2.2.2 Documentation and The Ecosystem

Originally, Python was used a lot for scientific purposes. This is why many libraries have specific formatting and high complexity. As we have already mentioned earlier, in this work we have used the Python 3 standard library, SciPy and Statmodels.
According to the provided evaluation table all used libraries will get points and the weighted score will be added to the python total score. The weights are approximate and are based on frequency of appearance and importance of the library in the program.
The base Python library:

1. 1 Point.

2. 1 Point - each function has one simple example, some functions are explained by an "identical function" example, at the end of the class description several composite examples are given.

3. 0 Point - the code can't be accessed freely.

4. 1 Point - functions are either explained in the documentation, or provide a link to an external source.

5. 1 Point - the documentation structure is similar for every class and has an understandable visual spacing.

From the SciPy source three libraries were used explicitly. These are NumPy, Matplotlib and Pandas. The documentation of the libraries themselves contains more detailed information and some additional examples, but the structure and the formatting is the same according to the Python styling standard. This is why we will evaluate the SciPy documentation instead of evaluating each library.

1. 1 Point.

2. 1 Point.

3. 1 Point.

4. 1 Point.

5. 1 Point.

The most important library for our Python program was Statmodels since we used it to build the models.

1. 0.3 Points.

2. 0.75 Points.

3. 1 Point.

4. 1 Point.

5. 0 Point.

Python gets 1 Point for the size and the quality of the community. We give the whole point because there are many universities developing Python as well as the serious core developers team and open source contributors. One of the interesting examples of a library developed by a scientific contributor is the library to place graphs directly into web [33]. Seaborn is developed and maintained by Standford. Another example is TensorFlow by Google [34]. This tool offers Java and Python APIs to build and train the neural network.
Python also gets 1 point for the diversity of the packages and libraries. The final score for the Python documentation is:

$$\frac{(\frac{2}{5}(4) + \frac{1}{5}(5) + \frac{2}{5}(3.5))}{5} = 0.8$$

The final score for this section is 2.8.

2.3 Advantages and disadvantages of the program

During the program development phase we have encountered several language specific advantages and disadvantages.
The first advantage is the small time investment needed up front. Python is intuitive and does not require deep understanding of the programming paradigm for a simple program. We consider Python to be suitable for prototyping and checking small hypotheses.
The second advantage of the language is testing. Python has built-in doctests. Another possibility to test a program consists in unit testing. Python provides different ways to test and to debug the program in order to avoid possible computational and logic errors. We will not discuss this theme in detail since the tutorials and the documentation are simple and offer sufficient amount of examples.
There are several disadvantages of the statistic program written in Python.
The first disadvantage is the performance. This disadvantage can appear mostly for beginners by implementing middle complexity tasks. For large data sets Cython might be used instead of a simple Python.
In order to avoid data formatting difficulties we have used Pandas library. There are several limitations associated with the use of this library. For instance Pandas is not compatible with PyPy. This limitation took a cheap possibility to speed up our program.
Another disadvantage of the program written in Python is the unintuitive function usage. For example, Python does not implicitly create copies. The expression *dict(a) = dict(b)* shows two objects that are referred to the same object. During mutating one of the objects all references will change in order to keep referring to the object in its

current state. The correct way to initialize a dictionary using another dictionary is *dict(a).copy(dict(b))*.
The last disadvantage we want to mention is the unsorted nature of the dictionary. After the csv table was read and formatted into dictionary with the companies names as the keys and the associated prices lists as the values. The keys appear in dictionary in random order.

2.4 Results

The end result is printed out to a console. The final message contains the names of the companies used in the final regression, main characteristics of the regression (AIC, BIC, Loglikelihood, coefficients and errors) and the total run time of the program. The results of the program for the given data described in the data preparation section are:

```
The best model contains 6 parameters. And the model is:
['STMElectro', 'Olympus', 'St Jude', 'Lenovo', 'MicronTech', 'Google']
                          GLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.998
Model:                            GLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 8.701e+04
Date:                Do, 24 Sep 2015   Prob (F-statistic):               0.00
Time:                        20:20:45   Log-Likelihood:                -1753.3
No. Observations:                1239   AIC:                             3519.
Df Residuals:                    1233   BIC:                             3549.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
         coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
x1     0.1254      0.016      7.933      0.000         0.094     0.156
x2     0.0609      0.012      5.065      0.000         0.037     0.085
x3     0.3261      0.016     21.029      0.000         0.296     0.357
x4     0.2126      0.004     47.272      0.000         0.204     0.221
x5     0.0083      0.001     12.929      0.000         0.007     0.010
x6     0.1291      0.012     10.536      0.000         0.105     0.153
==============================================================================
[Finished in 4173.2s]
```

To sum up the objective arguments for Python program computing a linear regression for a given data and restrictions we will sum up the points from the table from the thesis description section. The points for the Python program are:

1. Performance: 0 Points.
2. Memory: 0 Points.
3. Big Data: 1 Point.
4. Documentation: 0.8 Points.
5. Ecosystem: 2 Points.

The Python program gets 3.8 points in total.
The subjective arguments for and against are:

1. The syntax is simple.
2. All standard data structures are available.
3. Slicing of the data arrays is intuitive.
4. For small data arrays the performance of the language is sufficient.
5. Debug output is easy to understand.

3 R

3.1 The structure of the program

In this work we have written an R program the same way as the Python program. The R program consists of three classes. These classes are DataFormatting, StatisticTests and BiuldModel. The functions of the classes are similar to the functions from the Python program. There are several language specific differences. The main class is implemented in the from of two separate functions. All functions and imported classes were called via console in the R run time.
R is a convenient language in terms of data formatting. The native method read.csv() returns an object called data frame. This object has different methods allowing to use the data from different spots in the program directly without extracting it and saving in a proper format. For example, the list of all companies from the table was extracted using the function **colnames(data.frame)**. As an output we have got the list object containing the elements from the first row of the table. The R program does not have inner helpers as the Python program.
The class StatisticTests runs tests to detect statistic characteristics of the data. These tests are Kolmogorov-Smirnov goodness of fit distribution test, Augmented Dickey-Fuller test. Additionally a cumulative distribution function and a kernel density function will be built and moments will be found.
The third class called BuildModel creates the multiple linear regression for the given data table. The procedure of regression building is the same step-forward approach that was implemented in the Python program:

1. Cut off all companies with the correlation coefficient smaller than 30%.
2. Set the limit for the maximum number of parameters in the final regression.
3. Choose the best one-parameter model.
4. Build all possible combinations consisting of fixed companies from the previous steps and one unused company.
5. Choose the best model among the two-parameter model class.

6. Compare the best models from two different classes using the Likelihood Ratio Test.
7. Repeat until either the smaller model will be better, or the maximum number of parameters is reached.

3.2 The process description

In this section we will describe the language specific difficulties we have encountered.

3.2.1 Used Libraries

For our R program, the following packages were used:

1. Base package [35].
2. Stats package [36].
3. Nlme package [21].
4. Tseries package [20].

The functions from the base package were used for all basic computations. This package also provides different functions for the data formatting. The following example shows the R specific object called formula:

```
formula <- as.formula(paste(dep[name], '~', as.character(names(small_model)),
collapse=''))
```

This function converts the string object from the brackets into a formula object that is further used as an input parameter for building a linear regression. The base package offers basic math operation as well.
The StatisticTests class uses both stats and Tseries packages to build the statistic tests on the data. The Tseries package offers different tests and methods widely used in the computational finances and time series analysis. In this work we have only used the Augmented Dickey-Fuller test to check the data for stationary behaviour.
The second package used in StatisticTests class offers functions to run simple statistic tests on the data and build the basic statistic characteristics and provides the possibility to build a graph. In our work we have used the stats package to build graphs for each company from the data table and test the data for its distribution. The program has yielded the same results as the Python program.

The fourth package used in this program offers a possibility to build a linear regression using given parameters. The use of the functions from the library are intuitive: The first parameter is a formula object. An example of the this particular object was already demonstrated above. The second parameter is the data source. A necessary condition for the data object is actually to contain the variables from the formula.
The following example presents a graph with a default formatting style.

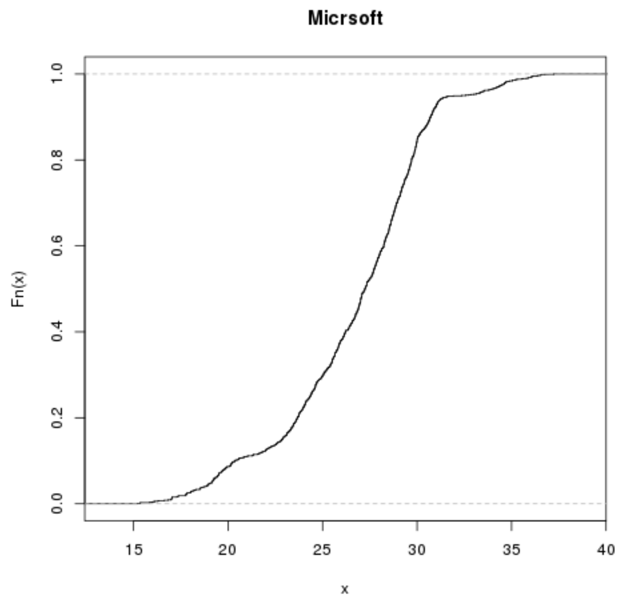

Figure 3.1: A basic plot built using The R Graphics Package library

3.2.2 Documentation and The Ecosystem

In comparison to Python, R was developed solely for analytical purposes. Since the language was built for analysis by the scientific community R's documentation is presented in a scientific manner. To evaluate the documentation we will use the points system described in the Python Documentation section.
The R Base package:

1. 1 Point.
2. 0.25 Points

3. 0 Points.
4. 0 Points.
5. 0 Points (no styling was applied).

The Stats package is organized in a similar way to the base R package and have the similar score:

1. 0.3 Points.
2. 0.5 Points.
3. 0 Points.
4. 1 Point.
5. 0 Points.

The Nlme package is hosted by the biggest R source called CRAN. This source offers a significant amount of simple and complex demos, test data sets and tutorials. CRAN is considered to be the central R source.

1. 1 Point.
2. 1 Point.
3. 0 Points.
4. 1 Point.
5. 1 Point (scientific paper formatting).

The last package used in our program is Tseries:

1. 1 Point.
2. 1 Point.
3. 0 Points.
4. 1 Point.
5. 1 Point.

The quality of the R documentation is on average on the same level for all sources. However the convenience of the documentation differs depending on the source and the development team. The total score for the R documentation is:

$$\frac{\frac{3}{10}(1.25) + \frac{1}{10}(1.8) + \frac{3}{10}(4) + \frac{3}{10}(4)}{5} = 0.591$$

The community behind R is more scientific than behind Python. Hoverer there are more scientific domain packages that the R community has developed. Additionally, to implement certain algorithm there can be more the one package available. The language gets 1 Point for the community size and quality criterion and 1 Point for the diversity of the packages. The final score for R for this section is 2.591.

3.3 Advantages And Disadvantages Of The Program

During the development phase of our second program we have encountered several difficulties associated with the language features. Since we have already written our program in Python we have decided to implement the same structure.
The first advantage of the R program is the data frame object and correlated functions. The performance of the slicing and formatting operations is good. One of the most useful functions during the work with R data objects was str(). This function shows the structure of the object:

```
> str(data)
 'data.frame':   1239 obs. of  69 variables:
  $ Intel              : num  26.6  26.6  26.4  25.8  25.9  ...
  $ AMD                : num  27.5  28.4  27.8  27.9  29.2  ...
  $ Qualcomm           : num  46.5  45.6  45.4  43.8  44.2  ...
  $ MicronTech         : num  13.5  13.6  13.6  13.6  13.9  ...
  $ Infenion           : num  9.26  9.38  9.41  9.3  9.25   ...
  $ STMElectro         : num  18.5  18.7  18.6  18.1  18.2  ...
 ...
```

There are several language specific functions to extract the sub data from the original data frame.
In the Python program where we had to store the data in a dictionary with the names as the keys and the lists of share prices as the associated values. In R the names of the companies from the data frame are coming in exact same order as they appear in the csv file.
The second advantage of the R program is connected to the the function we have used for the regression building. As we have described in the Program Structure and Used

Libraries sections for the regression we have used the GLS function from the Nlme package. In order to use this function the formula for the regression and the data containing all given names from the formula should be provided. The examples presented below is not the optimal solution for our task:

```
> model1<-gls(Intel ~ Olympus + Google + AMD + St.Jude, data)
> model1
Generalized least squares fit by REML
  Model: Intel ~ Olympus + Google + AMD + St.Jude
  Data: data
  Log-restricted-likelihood: -1959.999

Coefficients:
(Intercept)      Olympus       Google          AMD      St.Jude
-1.38174717   0.18731627   0.01450565   0.05510854   0.21830586

Degrees of freedom: 1239 total; 1234 residual
Residual standard error: 1.158712
```

Another advantage of the GLS function is its data use. In comparison to Python the data is not required to be specifically formatted. The data frame previously loaded into the environment will be used by default if no data is specified. If there are multiple data sources loaded into the environment the specification is not optional.
The third advantage of the R program is its performance. Without using the explicit multiprocessing or JIT the program has significantly outperformed the Python program. The actual results will be presented in the following section.
The first disadvantage of the R language is difficulty. The syntax is not as intuitive as the Python syntax. The data structures and their use are non trivial. In order to begin using R, a significant amount of time invested upfront is required.
The first example of R's complexity is the absence of the classes in their usual understanding. In R there are three type of classes. These are S3, S4 and Reference Class. The reference class is close to the Python class. The first two represent the attribute of an object. In R everything considered as an object. In our program we have used the reference class and didn't dive into the other two class types. For more information on OO systems in R you can visit the following source [37].
The second disadvantage of the R program is connected to Reference Class definition. In R the return statement with multiple instances is not allowed. In order to be able to return several objects it is possible to write them down into a list. The disadvantage of this workaround is the additional formatting of the resulting list.
Another disadvantage of the R program is the difficulty of the slicing. Different approaches can be used to get the same result. R provides multiple slicing possibilities

in order to simplify the formatting steps. The functions can return the instances of different classes.

3.4 Results

The main file implemented in the R program consists of two functions using implemented classes. The first function activates the libraries we have used in the program and calls the implemented classes in a specific order. The output is printed to the terminal. The extended information on the resulting regression is called separately to the result output for the GLS function:

```
> source("main.r")
> system.time(main())
Generalized least squares fit by REML
  Model: as.formula(form[[k1]])
  Data: data
       AIC       BIC    logLik
  3421.976  3462.907  −1702.988

Coefficients:
                 Value  Std.Error    t−value  p−value
(Intercept)   4.588765  0.29835356   15.38029       0
Cardinal     −0.073992  0.00540346  −13.69342       0
Olympus       0.054461  0.01080308    5.04127       0
St.Jude       0.165351  0.00801178   20.63845       0
Lenovo        0.500866  0.01499778   33.39599       0
MicronTech    0.073363  0.01235370    5.93858       0
STMElectro    0.435386  0.02655304   16.39685       0

 Correlation:
           (Intr)  Cardnl  Olymps  St.Jud  Lenovo  McrnTc
Cardinal   −0.208
Olympus    −0.492   0.210
St.Jude    −0.846  −0.113   0.212
Lenovo      0.447  −0.046  −0.834  −0.408
MicronTech −0.421   0.252   0.267   0.273  −0.180
STMElectro  0.494  −0.802  −0.590  −0.203   0.415  −0.614

Standardized residuals:
       Min          Q1         Med          Q3         Max
−2.3116653  −0.7048518  −0.0818459   0.6500465   3.5270072

Residual standard error: 0.9389829
Degrees of freedom: 1239 total; 1232 residual
```

```
     user   system  elapsed
   24.566    0.000   23.461
```

The best model returned by the R program consist of 6 companies. These companies are STMElctro, Olympus, St.Jude, Lenovo, Microntech and Cardinal. The R program has chosen Cardinal over Google on the last step of the evaluation algorithm. Since the programs are identical this deviation can be caused by the underlying implementations of the GLS function, the LLR-test, or the AIC criterion evaluation. Since we didn't find the source code for the functions we have used in this work, the cause of the difference in the results can't be explained properly. To compare the results of the programs we have built the predictions for both programs. First the accuracy of the original predictions will be compared. Afterwards the cross results accuracy will be compared. The program yielding the higher accuracy predictions will get an additional point.
To sum up the objective arguments for R we will give the points according to the evaluation table given in the description section. The score for the R program is:

1. Performance: 1 Point.
2. Memory: 1 Point.
3. Big Data tools: 1 Point (this is a relatively recent addition to the language. There are several sources and libraries offering tools to work with big data [38]).
4. The quality of the documentation: 0.591 Points.
5. The quality of the ecosystem of the language: 2 Points.

The total objective score for R is 5.591 points total.
The subjective arguments for the language are:

1. Debugging messages are hard to understand.
2. Certain amount of time is needed to understand the code since the syntax is not intuitive.
3. Layered structures were confusing.
4. No additional formatting saved time.
5. More.

4 Results

In this section we will present the final comparison of the programs written in Python and R.

4.1 Objective comparison

The first difference between the programs is the performance. In order to show measure the run time of the programs we will use the options that both language offer. These methods are *system.time()* for R and *cProfile()* for Python. Additionally to the language specific functions we will use the */bin/time* Linux utility. The following example shows us the run time for the R program.

```
> system.time(data_read())
   user  system elapsed
  0.100   0.000   0.085
> system.time(stat(build_data, companies))
   user  system elapsed
  3.627   0.000   3.465
>> system.time(model(build_data, companies, rest, dependent))
   user  system elapsed
  3.837   0.000   3.067
```

The main function consists of three parts. The first part reads the data from the csv table and formats it. The second part runs statistic tests. The third part buildings a model based on the results of the statistic tests. The overall time of the program can be measured via main and predict functions:

```
> system.time(main())
   user  system elapsed
  8.917   0.000   6.546
> system.time(build_predictions())
   user  system elapsed
  0.000   0.000   0.117
```

The run time of the main function is greater than the sum of its parts, since the main function loads needed libraries and also counts the time for the data transfer between

the classes as well as the initialization of the classes.
The overall run time of the Python program is:

```
>>> cProfile.run(formatData())
    4798725 function calls (4798229 primitive calls) in 3.626 seconds
>>> cProfile.run(getStatistics())
    32502787 function calls (31887367 primitive calls) in 2312.703 seconds
>>> cProfile.run(buildModel())
    9856742 function calls (79238036 primitive calls) in 4026.785 seconds
>>> cProfile.run('build_predictions()')
    6854581 function calls (6853565 primitive calls) in 6.057 seconds
```

The Python program performance in our implementation is 544 times slower. The R program runs for 7.41 seconds versus 4036.468 seconds of the Python program. Both programs run time may vary depending on the overall CPU load and running background processes. However these time differences are only noticeable for the Python program.
The second difference between the programs is the memory usage. There are several possibilities to measure the memory load of the program. Since every instance in R is treated as an object the size of all objects can be shown using Rprofmem() function from the utilities package. However, in order to use this function to compute the total memory use the sum of the all objects size needs to be computed. This approach is complex and does not guarantees the accurate result. Python also offers different tools to do the profiling. On of the possible functions is memory_profile [39]. In order to measure the memory use uniformly for both languages we will use the Unix time command. The printed message contains the information about the called process. Two main characteristics we are interested in are the maximum resident set size and the user time. The maximum resident set size shows the amount of the memory belonging to the process and currently presented in RAM. In another words it is the maximum memory allocated in the heap by the program.
The R memory use looks as following:

```
[20:32:19 - 15-10-25]
/home/alisa/uni-stuff/Bachelor/r % /usr/bin/time -v R  main.r
        Command being timed: "R main.r"
        User time (seconds): 7.41
        System time (seconds): 0.05
        Percent of CPU this job got: 18%
        Elapsed (wall clock) time (h:mm:ss or m:ss): 0:41.45
        Average shared text size (kbytes): 0
        Average unshared data size (kbytes): 0
        Average stack size (kbytes): 0
        Average total size (kbytes): 0
        Maximum resident set size (kbytes): 78436
        Average resident set size (kbytes): 0
```

```
          Major (requiring I/O) page faults: 0
          Minor (reclaiming a frame) page faults: 17697
          Voluntary context switches: 59
          Involuntary context switches: 2026
          Swaps: 0
          File system inputs: 0
          File system outputs: 2840
          Socket messages sent: 0
          Socket messages received: 0
          Signals delivered: 0
          Page size (bytes): 4096
          Exit status: 0
```

The maximum memory usage of the R program is 78,4 MB.
The Python memory use looks as following:

```
[19:11:27 - 15-10-25]
/home/alisa/uni-stuff/Bachelor/python % /usr/bin/time -v python main.py
          Command being timed: "python __main__.py"
          User time (seconds): 5564.19
          System time (seconds): 7.73
          Percent of CPU this job got: 100%
          Elapsed (wall clock) time (h:mm:ss or m:ss): 1:32:46
          Average shared text size (kbytes): 0
          Average unshared data size (kbytes): 0
          Average stack size (kbytes): 0
          Average total size (kbytes): 0
          Maximum resident set size (kbytes): 637036
          Average resident set size (kbytes): 0
          Major (requiring I/O) page faults: 0
          Minor (reclaiming a frame) page faults: 3633280
          Voluntary context switches: 3047
          Involuntary context switches: 22080
          Swaps: 0
          File system inputs: 0
          File system outputs: 8976
          Socket messages sent: 0
          Socket messages received: 0
          Signals delivered: 0
          Page size (bytes): 4096
          Exit status: 0
```

The maximum memory usage of the Python program is 637,0 MB.
As we have mentioned earlier, we will compare the predictions of the programs and evaluate the cross results. The predictions were built on the TestingSet file containing 515 prices for the time period from 1.01.2011 to 31.12.2012. We will show the main characteristics of the predictions row: mean and standard deviation as well as the comparison

plot.
The R program gives us following results:

```
> p<-build_predictions()
 [1] "the mean of the y is: "
 [1] 23.46462
 [1] "the std of the y is: "
 [1] 2.55532
 [1] "the mean of the predictions is: "
 [1] 20.86923
 [1] "the std of the predictions is: "
 [1] 1.544088
 [1] "the mean absolute error of the predictions is: "
 [1] 3.29319
 [1] "the root mean squared error of the predictions is: "
 [1] 4.132798
```

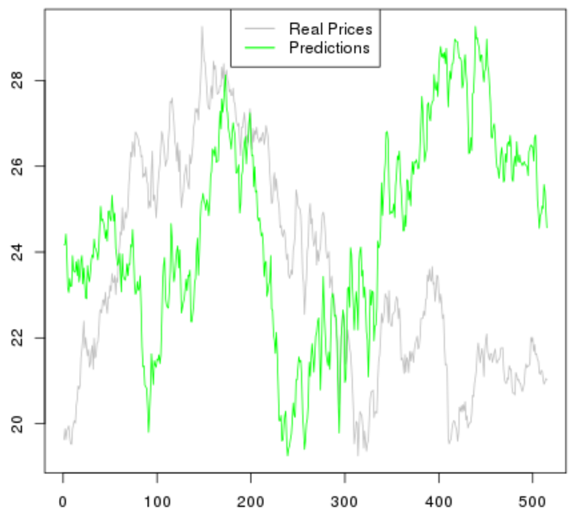

Figure 4.1: Predictions build by the R program.

The Python program results are:

```
>>> build_predictions()
the mean of the y is: 23.4646213592
the std of the y is: 2.55283813168
the mean of the prediction is: 21.8848266352
the standard deviation is: 1.35949376438
the mean absolute error of the prediction is: 2.83819120505
the root mean squared error is: 3.47169327209
```

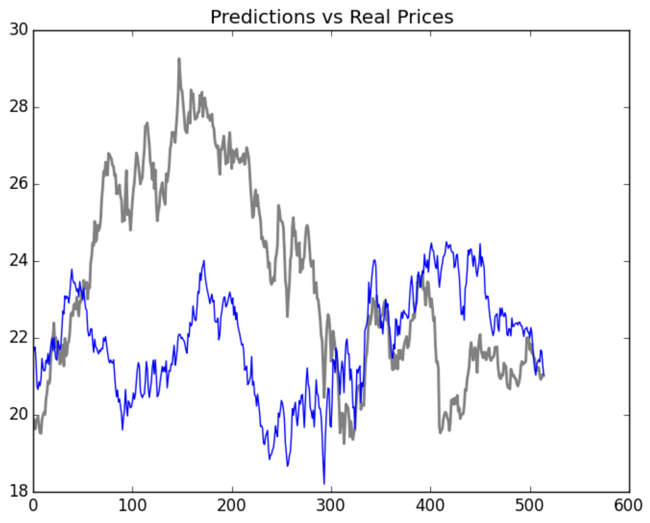

Figure 4.2: Predictions build by the Python program.

The Python predictions according to the chosen criteria are more accurate than the R predictions. Both mean absolute error (MAE) and root mean square error (RMSE) are smaller for the Python predictions. The Python program gets one additional point for the relative accuracy. On the following graph both original R and Python predictions are shown. The R forecasts have the green line and the blue line presents the Python forecasts.

As the following step we will force both Python and R programs to compute the results for the model yielded by the other program. Python will compute predictions for the final model from R: ["Cardinal", "Olympus", "St.Jude", "Lenovo", "MicronTech", "STMElectro"]. And the R program will return the forecast for the Python final model: ["Google", "Olympus", "St.Jude", "Lenovo", "MicronTech", "STMElectro"].
The R cross result is:

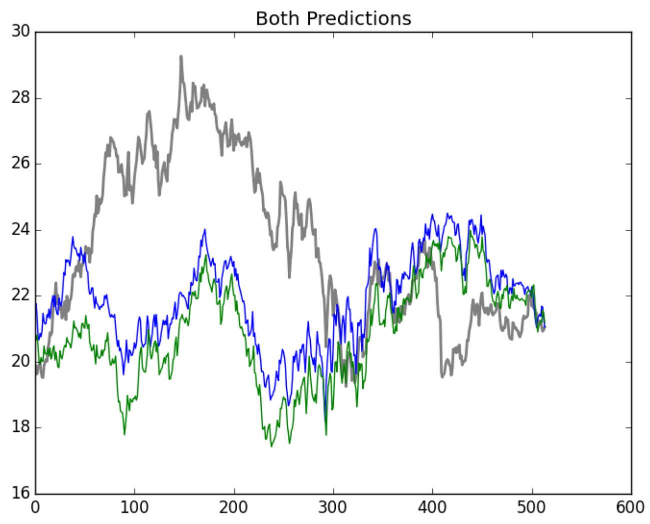

Figure 4.3: Predictions build by the both Python and R programs.

```
> p<-build_predictions()
 [1] "the mean of the y is: "
 [1] 23.46462
 [1] "the std of the y is: "
 [1] 2.55532
 [1] "the mean of the predictions is: "
 [1] 21.73608
 [1] "the std of the predictions is: "
 [1] 1.152263
 [1] "the mean absolute error of the predictions is: "
 [1] 2.631951
 [1] "the root mean squared error of the predictions is: "
 [1] 3.305496
> p$model
 Generalized least squares fit by REML
   Model: Intel ~ Olympus + Google + St.Jude + MicronTech +
   STMElectro +       Lenovo
   Data: build_data
   Log-restricted-likelihood: -1760.532

 Coefficients:
 (Intercept)      Olympus       Google       St.Jude
```

```
2.299629176     0.047291974   0.005870799 0.165562665
MicronTech        STMElectro        Lenovo
0.105875773      0.167784354   0.401918284

Degrees of freedom: 1239 total; 1232 residual
Residual standard error: 0.982245
```

On the following image the difference in two models forecasts returned by the R program is represented. The gray line shows the real Intel prices, the green line is the new model forecast and the blue line is the old model forecast.

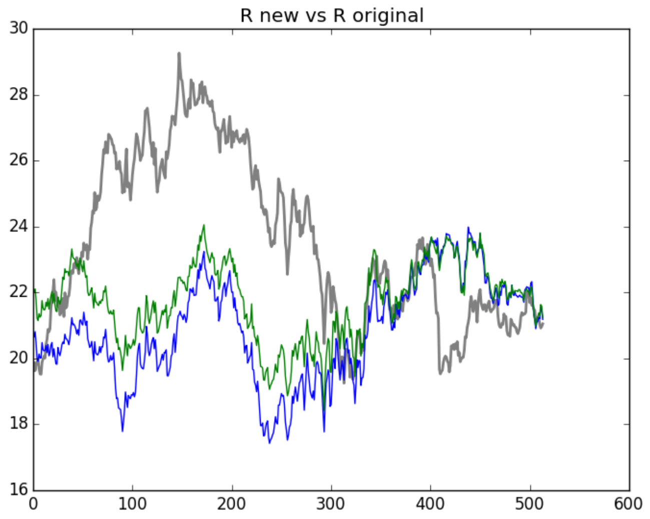

Figure 4.4: Predictions build by the R program for the original model and new model.

The R program considers the new model to be better than the original suggestion since the new MAE is 0.661 smaller and the new RMSE is 0.827 smaller. The original model was chosen based on the AIC. The forecast was built only for the final model. That means that we never considered the accuracy of the predictions as the criterion for choosing the best model in the class or among two classes.
The following graph represents the predictions for the same model but estimated by different program. The model is ["Google", "Olympus", "St.Jude", "Lenovo", "MicronTech", "STMElectro"]. As usual the gray line presents the real values, the green line is the R forecast, the blue line is the Python forecast.

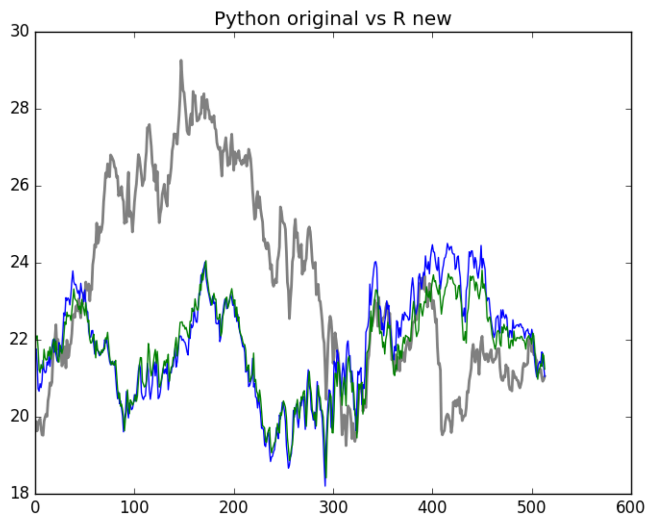

Figure 4.5: Predictions build by the Python program for the original model and predictions built by the R program for the new model.

We have noticed that the predictions built by the R program for the Python original model have higher accuracy than the Python predictions for the same model. The R MAE is 0.208 smaller and the R RMSE is 0.166 smaller than the Python results. The Python cross results are:

```
>>> build_predictions()
the mean of the y is: 23.4646213592
the std of the y is: 2.55283813168
the mean of the prediction is: 20.699905831
the standard deviation is: 2.18673197501
the mean absolute error of the prediction is: 3.9100716967
the root mean squared error is: 4.81718549588

the model is: ['STMElectro', 'Olympus', 'St Jude', 'Lenovo', 'MicronTech',
'Cardinal']
                              GLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.998
Model:                            GLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                 8.261e+04
Date:                Mon, 26 Oct 2015   Prob (F-statistic):               0.00
```

```
Time:                        14:34:25   Log-Likelihood:              -1785.4
No. Observations:                1239   AIC:                           3583.
Df Residuals:                    1233   BIC:                           3613.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
x1             0.2337      0.025      9.273      0.000         0.184     0.283
x2             0.1363      0.010     13.277      0.000         0.116     0.156
x3             0.3977      0.015     27.166      0.000         0.369     0.426
x4            -0.0567      0.006     -9.826      0.000        -0.068    -0.045
x5             0.2696      0.005     57.903      0.000         0.261     0.279
x6             0.1533      0.012     12.533      0.000         0.129     0.177
==============================================================================
```

The following graph shows the difference between two model forecasts returned by the Python program. The gray line shows the real values, the green line presents the new model forecast and the blue line is the old model forecast.

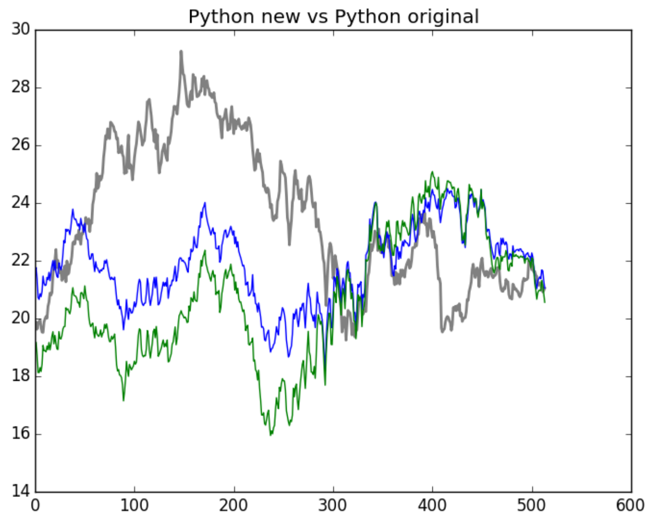

Figure 4.6: Predictions build by the Python program for the original model and new model.

The Python program considers the new model to be worse in comparison to the original suggestion since the MAE is 1.072 smaller and the RMSE is 1.4 smaller than the original results. The first Python model was chosen based on the AIC and the predictions were also not taken into account during the evaluation steps.
The following graph presents the predictions for the same model but estimated by different program. The model is ["Cardinal", "Olympus", "St.Jude", "Lenovo", "MicronTech", "STMElectro"]. Same color notation is used.

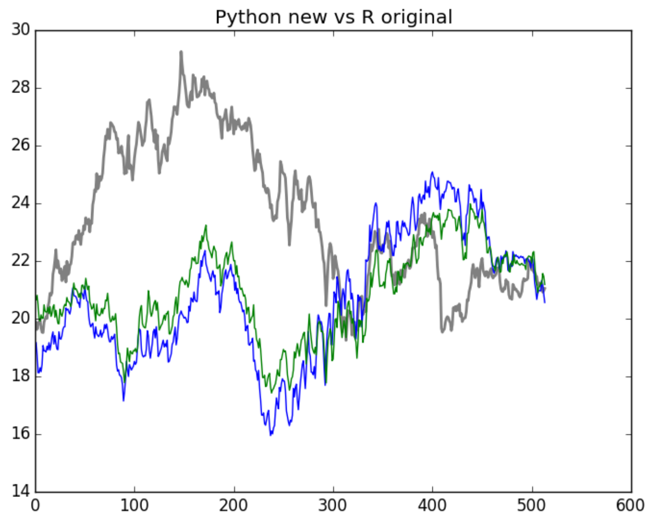

Figure 4.7: Predictions build by the R program for the original model and predictions built by the Python program for the new model.

The Python program predictions are less accurate than the R predictions for the same model. The R MAE is 0.617 smaller and the R RMSE is 0.684 smaller.
The highest accuracy forecast was built by the R program for the Python original model. This result can be explained by the fact that the R gls.predict() module uses double precision floats instead of single precision floats. At the same time the Python program has returned better regression using the AIC during the evaluation process. Python gets an additional Point for yielding higher accuracy results.
The final score for both programs is 5.591 Points for the R program versus 4.8 Points for the Python program. This mean that objectively R is more suitable for beginners to use as the programming language for a small statistic task with a small data set.

4.2 Subjective comparison

During the process of the program development both languages have shown some of their advantages and disadvantages. The subjective advantages of Python are:

1. Easy code refactoring.
2. Sufficient amount of tutorials and examples for every function used in the program.
3. Intuitive data structures.
4. Clear debugging messages.
5. Possibility to use a simple speed up in the form the of PyPy.

The subjective disadvantages of the language are:

1. Slow third-party functions for statistic tests.
2. Formatting step a lot of time to implement.
3. The Pandas library does not work with the PyPy interpreter.

The R program subjective disadvantages encountered during the program development are:

1. The run time.
2. Data structures are easy to use.
3. Low memory allocation.

The subjective disadvantages of the language are:

1. Unintuitive syntax.
2. Difficulties while calling R code outside of the run time.
3. Not enough examples for the functions usage.

Subjectively speaking R is slightly better for the data analysis programs when the program architecture was already specified. For the higher complexity programs R will be more suitable than Python.
Python proved to be more suitable for a fast prototyping using a small data set. Another advantage of the Python program is the absence of the integration necessity.

4.3 Conclusion

After point to point comparison and collecting the personal opinion, we have decided, that R programming language is bit more suitable for a purely analytical task.

1. The language is faster.
2. The language is not as memory hungry.
3. The results are adequately accurate.
4. The complex functions are easier to use.

The R language has a big high-quality community behind it. There are certain standards for the documentation and the code quality, which ensures the quality of the libraries. The negative moment can be the development of a bigger program. Prototyping in R can be tricky for medium complexity and very complex programs. You have to think in advance about class connections. Although you can also keep everything in functions form - this will allow you to prototype faster, but will cause some difficulties later on during refactoring.
Our conclusion suits the modern trend which says that R is very popular among the beginners and advanced analysts. However R programmers are slowly beginning to turn to Python due to its multipurpose nature and freedom.

Bibliography

[1] Wikipedia. History of the internet, September 2015.

[2] IBM. Ibm 350 disk storage unit, 1956.

[3] Sebastian Anthony. Samsung unveils 2.5-inch 16tb ssd: The world's largest hard drive. Aug 2015.

[4] Fremont Rider. In *The Scholar and the Future of the Research Library, a Problem and its Solution*, 1944.

[5] Derek J. de Solla Price. In *Little Science, Big Science and Beyond*, 1986.

[6] Michael Cox David Ellsworth Steve Bryson, David Kenwright and Robert Haimes. Visually exploring gigabyte data sets in real time. August 1999.

[7] LinkedIn. Personal profile.

[8] Roger Magoulas John King. 2015 data science salary survey.

[9] Wikipedia. Python (programming language).

[10] Wikipedia. Python success stories.

[11] ForecastWatch. A project built solely in python.

[12] Industrial Light & Magic. Special effects for movies.

[13] Open Source. Molecular modelling toolkit.

[14] Serge Dellerue Marie-Claire Ballisent-Funel Konrad Hinsen, Andrei-Jose Petrescu and Gerald R. Kneller. Harmonicity in slow protein dynamics. *Science Direct*, 261:25–37. [Online; accessed 10-December-2015].

[15] Aline Thomas Konrad Hinsen and Martin J. Field. Analysis of domain motions in large proteins. *PubMed*, 261:25–37. [Online; accessed 10-December-2015].

[16] Javad Golji and Mohammad R.K. Mofrad. A molecular dynamics investigation of vinculin activation. *PMC*. [Online; accessed 12-December-2015].

[17] Grégoire Montavon and Klaus-Robert Müller. Deep boltzmann machines and the centering trick. *Springer LNCS*. [Online; accessed 09-December-2015].

[18] Wikipedia. R (programming language).

[19] Yves Rosseel. lavaan: An r package for structural equation modeling. *Journal of Statistical Software*, 48. [Online; accessed 10-December-2015].

[20] Kurt Hornik Adrian Trapletti and Blake LeBaron. Time series analysis and computational finance, February 2015.

[21] Saikat DebRoy Deepayan Sarkar-EISPACK authors-R-core José Pinheiro, Douglas Bates. Linear and nonlinear mixed effects models, August 2015.

[22] Ben Bolker Julien Claude-Hoa Sien Cuong-Richard Desper Gilles Didier Benoit Durand Julien Dutheil Olivier Gascuel Emmanuel Paradis, Simon Blomberg. Analyses of phylogenetics and evolution, November 2015.

[23] Stefan Fritsch Frauke Günther. neuralnet: Training of neural networks. *The R Journal*, 2.

[24] Sanford Weisberg John Fox. neuralnet: Training of neural networks. 2.

[25] Biostars Forum. Time series analysis and computational finance, 2015.

[26] StackExchange Forum. Forum for statistical questions, 2015.

[27] Stackoverflow Forum. Forum for statistical questions in programming, 2015.

[28] Jess Fern·ndez-Villaverde S. Boragan Aruoba. A comparison of programming languages in economics. [Online; accessed 18-December-2015].

[29] DataQuest. R vs python: head to head data analysis, October 2015.

[30] Python Org. Python official documentation, 2015.

[31] SciPy.org. Python based ecosystem for statistical and analytical tasks, 2015.

[32] Statsmodels. Python module for statistical tasks, 2012.

[33] Michael Waskom. Seaborn: statistical data visualization, 2012.

[34] Google Brain Team. Open source software library for machine intelligence, 2015.

[35] R core. The r base package.

[36] R core. The r stats package.

[37] Hadley Wickham. Oo field guide.

[38] G. Ostrouchov, W.-C. Chen, D. Schmidt, and P. Patel. Programming with big data in r, 2012.

[39] Philippe Gervais Fabian Pedregosa. Memory profiler, 2015.

Eidesstattliche Erklärung

„Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Studiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.“