University of Glasgow | School of Computing Science

IPython Observatory

Silviya Sotirova

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Masters project proposal

March 2, 2016

Contents

1 Introduction

1.1 Context and Problem Statement

Software development projects for scientific researches are difficult to organise, document and present. There are many examples of the use of software causing the introduction of subtle, but significant issues into scientific research, such as code errors and failures, lack of documentation, false results, and others.

Python is a programming language that has been developed to be an easy to learn, code and read and has becoming increasingly popular in the scientific programming community for data analysis. Running, testing, debugging and documenting Python scripts has required the usage of various software tools, such as the command prompt (or another command-line interpreter), text editors and others.

IPython has been created as a software tool, interactive shell for standard scientific Python scripts, that allows combination of styled text, code and data visualizations.[57] It is designed to mimic interactive development environments such as MatLab [15] and Maple [14] for the writing, testing and debugging of Python code and it has a productive environment for analytical computing. [57] Running Python scripts is effective for immediate response, since developers can see the results right away.

Software can be used in both the development of theoretical models and the analysis of experimental results. Scientific programming is a crucial element of research analysis, since it can give faster and more accurate results - all calculations, models and visualizations can be computed with the help of software tools. As a consequence, scientific programming can be viewed as a another section of conducting researches.[50] However, researchers are only beginning to understand how software is used in scientific research and the problems encountered when using software.

Scientific programming bring great help and support research, but they have downsides - programming scripts need to be created for a specific area of study, which needs knowledge of both software implementation and the science of the investigated field. Usually analysts are well familiar with only the science that they are doing and as a consequence, they needs to gain computer programming skills. However, programming languages are challenging to learn and a lot of time and practicing is needed for the implementation of the functionality. The analysts might not have the necessary preparation and training in order to create the computations.

Furthermore, the target readers would probably not be acquainted with the code in the research, so a proper documentation of the software scripts is required - scientists from different areas of studies would be able to understand the code in detail. Thoughts, ideas and findings also need to be recorded for analysis and investigation of the gathered information. Researchers have discover an effective approach for detailing each aspect of a project - handwritten notebooks.[49] They are commonly used for laboratory experiments.

However, researchers have encountered problems with the usage of notebooks - they might not be well organised; each scientist has different style of writing and the reader might have difficulty of reading the notebook; tracking and predicting data with the handwritten notebook is challenging and time-consuming process;

The above problems show that scientists are striving to achieve in reproducible research*" the ability of an entire experiment or study to be duplicated, either by the same researcher or by someone else working independently"*, and replication of results - *"is the repetition of an experimental condition so that the variability associated with the phenomenon can be estimated"*. [36] [35] They want to be able to produce more shareable document that would give details about the implemented code and the analysis of the findings at the same time.

IPython Notebook is an open-source projects and they have become an option for scientific researches. The easy set-up and usability is allowing scientists, from different areas of study, to operate with it. It is worth investigating IPython usage in science, since it is an electronic version of the notebooks that researchers have. Also, the main feature of IPython, combining text, code and code output in one file, is of great interest for structuring scientists' ideas and computations. In order to assess the effectiveness of IPython in researches, we have to analyze how is it applied by them and what impact will it have on the problems mentioned above, the issues encountered with the usage of scientific software.

As a consequence of that, there are specific features of software usage that the project needs to investigate in order to give effective assessment of IPython notebook - e.g. how many GitHub repositories are using IPython for scientific researches, how is it used in a specific area of study, depending on the text and code implemented, what is the proportions of amount of code and the amount of text in the IPython scripts, and others. These questions are illustrating challenges that the project has to confront in order to find the best approach of evaluation of the IPython notebook.

The *IPython Observatory* project is exploring an explicit environment of researches available for everyone - open source repositories in order to understand the structure and content of IPython notebooks. GitHub, a web-based Git repository hosting service, provides a rich source of IPython source code repositories on the Internet.[32] A huge number of researchers have started to investigate and anlayze GitHub repostories so that they can understand how users are exploiting the site for collaboration on software. [53] In order to assess the value of IPython in researches, this project will investigate, through analysis of iPython notebooks hosted on Github, how IPython is used in repositories?

1.2 Proposed Approach

The project has is analyzing in detail the IPython results from the GitHub search. GitHub is a widely used open source hosting service for projects and, recently, it has become effective place for sharing various researches and contributing to them. It has been chosen for investigation, since it allows tracking of data changes, commits, contributions and documentation.

The analysis of IPython notebooks on GitHub is broken down to several steps, which are explained in more detail in section 3. Replication and reproducibility of IPython in researches will be the main elements of investigation - they are fundamental for replicating of scientific results.

1.3 Structure of the paper

Section 1.1 of the paper is *"Introduction"*, which gives overview of project's idea, the origins and details of the problem and the paper's structure. In the *"Introduction"* the readers should be able to understand why the usage of IPython notebook is worth analysing.

Section 2 is presenting critical analysis over existing researches - since IPython is a software tool, the reader needs to understand the value of software in scientific researches 2.1, the usage of Python applications 2.2, laboratory notebooks 2.3, example researches of IPython in scientific programming 2.4 and the data support from open source repositories, such as GitHub 2.5.

Section 3 is going through the steps of the project - downloading information from GitHub repositories using IPython, analysing GitHub repositories by the aspect from figure 2, analysing reproducibility and replication of IPython scripts in the downloaded information and their nature of usage - amount of code and text inside them.

2 Background Survey

2.1 Software in Science

To clearly understand the problem that this project is trying to solve, we have to go into detail about the concept of using computational applications. The contribution of the research is to understand the usage of software in science. However, encompasses a wide range of use cases with software used for a variety of purposes. This literature review begins be reviewing existing examples of the use of software in scientific research.

2.1.1 Contributions and Approaches

A great deal of researches has been conducted for answering the above questions, which later led to to the idea of observing these investigations in order to assess broadly how software development is practiced in research.

One example for scientifically modeling natural phenomena is the formulation of a snowflake. Norman Packard created the illustration with the use of a program, which demonstrates cellular automation.[69][59] The model represents the surface as cells - they have value 0 or

1, which corresponds accordingly to water vapor(black) or to ice(colour). The construction of the model starts with a red cell in the center of the visualisation. Afterwards, it continues by developing series of steps. In order to identify the value of the next cell, the values of the six surrounding cells has to be summed - if it is an odd number, then the cell has to be ice with value 1, if not - the other way around, vapor with value 0. The snowflakes consists of red and blue colours. Its creation requires at least 10,000 calculations. The best possible approach for the accurate and fast generation of the model, was by using computer stimulation. Even if the calculations are simple, the software script helps with the correct build of the pattern.[69] In figure 1 on page 5 you can see the snowflake illustration.

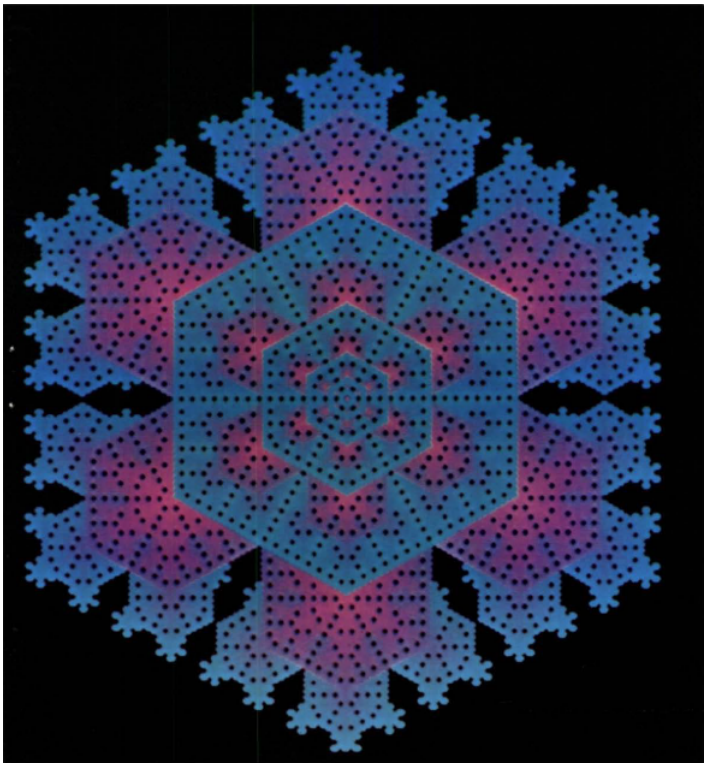

Figure 1: Computer Stimulation: Cellular Automation of a snowflake

Recording the changes of data in time is crucial for a lot of researches, such as animal tracking data - it gives us information about how individuals and populations migrate from local area, travel across oceans and continents, and develop during centuries. All of this knowledge gain from the analysis, can be a valuable tool for predicting climate changes, biodiversity loss, invasive species, diseases or even important events in the economics. [43]

Coyne and Godley's paper [43] presents details about satellite tracking and analysis tool, called STAT. Up until now, Argos systems have been used for recording animal data - *"satellite-based system which collects, processes and disseminates environmental data from fixed and mobile platforms worldwide"*.[26] Its collected data has a lot of advantages and benefits for journals, but it is not easy to analyze and manipulate by biologists - various of technical skills are required to operate with the given information. The research is demonstrating the STAT system, explaining all of its features - overview of the system, how it handles data management, data filtering and editing, integration of environmental data,

and how it influences the education and the public. It is freely available software specifically created for animal tracking biologists in order for them to easily manage, investigate and integrate data.

Finally, software has an important role in predictions, such as climate change, automation of systems and artificial intelligence. [44][68] Huge and complex simulations are computed by researchers who have little software skills and are not adapting to new technology fast. A case of such project can be considered for climate scientists - the paper of Steve Easterbrook and Timothy Johns [44], an ethnographic study of the usage and management of software tools by climate analysts. The research has been conducted at the Met Office Hadley Centre, which is one of the UK's most crucial climate change research centres and it produces essential information for climate changes and science. [9] Steve Easterbrook and Timothy Johns [44] are analysing how scientists are applying the technology in terms of software quality, correctness, managing tasks, testing and collaboration.

The approach taken in this research consisted of eight-week observational study at the Met Office Hadley Centre constructed by ethnographic techniques. In order to determine methods and performances of the climate scientists, investigators have discovered several fundamental properties of assessing the usage of software in the centre:[44]

- Correctness - how and what does it mean to the scientists?
- Reproducibility - of experiments and code.
- Shared Understanding - representation and knowledge of the large system.
- Prioritization - of requirements and tasks, which of them are achievable or worth implementing for the research.
- Debugging - catching of errors, failures and bugs.

Furthermore, the paper introduces the modeling that has been used for basic climate computational analysis - General Circulation Models (GCMs), *"which represent the atmosphere and oceans using a 3-dimensional grid and solve the equations for fluid motion to calculate energy transfer between grid points"*. [44] The paper illustrates of the complexity of generating and operating climate data and the value of using software in science. The models require great computer power so they can be analysed and visualised.

The reported findings presents greatly developed and integrated techniques of the climate researchers for maintenance and management of the system and their courses of actions to scientific analysis. The Met Office Center's team are applying software methods, such as the agile approach. Software engineering practices are proved to be of an essential support for conducting a scientific research. Moreover, the Met Office has maintained a software that is greatly accustomed to the data domain that the climate scientists have collected. Issues, such as performance and portability are tested ensuring that the software tools are correctly coordinated with the researchers' practices. The study observes that these procedures are frequently treated as controlled experiments. Also, the field of climate

science has invested in technology for efficient and accurate exploration of data, which is an indication that scientists have recognised what benefits can software tools give them. [44]

The above examples illustrate the support that computing can give to analysts when exploring a field of study. From the observed approaches we can derive common properties that each investigator is trying to cover - properties connected with the scientific method, which is *"a body of techniques for investigating phenomena, acquiring new knowledge, or correcting and integrating previous knowledge."*[38] In the Steve Easterbrook and Timothy Johns's paper [44] the approach is evaluating five main features - correctness, reproducibility, shared understanding, prioritization and debugging. Every scientists considers these properties. Nevertheless, they are not the only one that need detailed attention - the process of the scientific method includes steps that require examination, as well, so that there are no downsides of the implemented software:[38][68]

1. Formulation of a question - gathering and assessing evidence from experiments, observations and related work of other researchers.

2. Hypothesis - derived from the knowledge that the investigator has about the area of study. An important aspect of the hypothesis is that it has to be falsifiable, which means that it has to lead to predictions of the results that can be experimentally contradicted.

3. Prediction - determined from the hypothesis. It has to contain logical consequences and be eligible for testing.

4. Testing - hypothesis has to be assessed with a lot of experiments. The experiments has to determine if the made inspections of the real world support or contradict the predictions derived from a hypothesis. Also, the tests must be repeatable, which can be ac hived by well organised documentations and recordings. It is helping with the effective presentation of the hypothesis and the software usage.

5. Analysis - includes analysis over the results of the experiment and decision making for the next operation of the research. The predictions of the hypothesis are compared to those of the null hypothesis, so that it can be established which is better for describing the data. The results have to reproducible by an independent experiment and scientists need to be aware of how they are going to accomplish that. The software tools supports the recreation of the same results and the statistical analysis of the data. If the reproducibility feature is not met, there will be a lot concerns about the validity of the hypothesis. From this, we can derive one more aspect that scientists try to accomplish - clearly identifying the limitations of the results.

The use of software can be both beneficial and an obstacle to these steps. An example of a software defect in scientific analysis is given with the Sanders and Kelly's paper - Sander's [64]. It is reported that one of the subjects reported results that are completely different outcome of what the software was meant to do. They have several papers, such as Kelly's

papers [55][54], which are focusing on the usage of software and the necessity of testing it so that the results will be accurate.

Furthermore, replication and reproducibility are playing an important role in the usage of software in scientific researches and they are crucial aspects of the scientific method. The following implications apply for them [50][60][56]:

- Replication - "An author of a scientific paper that involves numerical calculations should be able to rerun the simulations and replicate the results upon request. Other scientist should also be able to perform the same calculations and obtain the same results, given the information about the methods used in a publication."[50]
- Reproducibility - "The results obtained from numerical simulations should be reproducible with an independent implementation of the method, or using a different method altogether."[50]

In order to succeed with this features, we have to consider:

- To keep track and record of the code version that is generating specific data.
- To have access to the environment and software that is used.
- To save and back up code and documentation for future work.
- To publish code online, so that it can gain interest of other scientists in the code and the presented findings. [50]

To sum up, this section is presenting three main features of software usage in science - it can be applied in algorithms and numerical analysis, tracking and recording of data quality and visualizations, and predicting future data. Each of them has presented a description of the conducted researches, approach and outcomes. Software is helpful and effective tool for reaching a successful analysis and unique findings. However, with the powerful applications, comes great responsibility. Maintaining, controlling, debugging and documenting software is a challenging, but necessary task for achieving outstanding results.

2.2 Python in Scientific Programming

Amongst high-level open source programming languages, Python is today the leading tool for general-purpose source scientific computing, finding wide adoption across research disciplines, education and industry [62].

Python is effective programming language for converting scientific code from another language [62]. It was created not only for application of simple calculations and operations. Python can become a high-level language, which can be used in software engineering or scientific environments. Additional extensions for Python are created everyday and they are available and understandable for everyone [65].

2.2.1 Python Facilities for Scientific Programming

A lot of scientific researches use Python for computing complex functions, which results in more accurate calculations and predictions than measuring them manually. Python has not been widely used in statistical measurements, but nowadays various scientists find its usage effective and powerful. With the creation of libraries, such as Matplotlib [8], NumPy [11] and SciPy [19], Python has became one of the most preferred software tools for research, testing and engineering [61]. There many more examples for the usage of Python package libraries, such as SkLearn - package containing implementation of various machine learning algorithms, and OpenCV - image processing and analysis. [22][17]

Data visualizations are great practice for conducting accurate analysis - they show relations between data items and objects. Python has made enormous contribution to their development and broad usage. For example, IPython Notebook is an interactive tool for combining and styling text and code concurrently. Also, parallel processing with processes and threads, Interprocess communication (MPI), GPU computing, have found great support from Python programming.

This section is explaining how researchers have tried to assess the usage of Python in scientific computing. There are a variety of approaches and methods for determining the easiness of features in Python - the best of them is the practical use of Python in different cases. SciPy conferences subsection is describing the experiences of analysts with Python.

2.2.1.1 Syntax and structure of Python code

In this subsection we are mainly going through one article, Oliphant's [58], from other research documentation.

High-level languages can enormously increase productivity and that is why they have been used for scientific computing for many years. Countless analysts and researchers have found these language to be of great use for them, since they are facing tasks of implementing nontrivial computational software prototypes in order to prove a concept for their area of exploration. Python is one of these languages and it allows engineers to be more careless with syntax, error handling and compilation time. However, comparing it with other languages, Python presents more effective environment for scientific applications.

Travis Oliphant's article continues by listing the basic features of Python: liberal open source license - allows selling, using, or distributing of Python-based applications, Python running on many platforms - no concerns about writing an application with limited portability, the languages clean syntax - object-oriented coding, depending on the situation, a powerful interactive interpreter - allows code development and experimentation, and others.[58]

All of the above features are useful for implementing scientific computations with Python. Travis Oliphant goes into more detail about the clean syntax, useful built-in objects, func-

tions and classes, standard library, ease of extension, and the importance of libraries, such as NumPy and SciPy. With examples, he is able to show that only with a few lines of code, a complex functionality can be implemented, and errors, failures and bugs can be caught.

The Travis Oliphant's paper[58] provides a "small taste" of Pythons usefulness and he is going through the language itself. The author is supporting the argument that Python is beneficial by starting with the syntax, which makes the code easy to understand and maintain, continuing with more examples for the built-in scalar types and the prepackaging libraries, and ending with SciPy package's calculation and computational ability. The structuring and syntax of the programming scripts is an important part since the concentration of a researcher will be mainly focused on the accuracy of the calculations and the discovery that was made, than in the validity of the code. With Python both features can be complete.

The article argues that Python is clear to read, study and apply in various software applications. However, it is not going through a lot of detail for execution and compilation time [45]. Since Python is an interpreted language, it runs many times slower than compiled code. Then why we should consider using it in scientific complex functions? The next paper reviewed considers of the performance of Python.2.2.1.2

2.2.1.2 Performance of Python

Python must be compiled before it is run - it differs from C, C++ and other languages. Python programs are scripts - instead of having compilation process, Python is interpreted line-by-line. The positive side of it is the code flexibility for problem solving. However, if a complex functionality is implemented with Python rather than with some other compiled language, it will run slower. We are returning again to the question - why should we use such a time-consuming programming language in scientific computing? [66]

Hans Fangohr's paper [45] gives two answers to that criticism - implementation time versus execution time, and well-written Python code can be very fast.

The paper argues that not only the computational time has to be considered in the overall time for the whole process of scientific software prototype's implementation. Since the first scientific computing, the computer's processing has increased significantly. Also, it is of great importance how the code will be written, maintained and how number lines of code it will contain. If the code is short, there is a possibility for less errors and failure, and it gives faster approach for testing and maintenance.

The Cai's and Xing's article [41], addresses the performance of scientific applications that use the Python programming language. It introduces several algorithms for increasing the efficiency of serial Python codes, after which it goes into detail for parallelization of serial scientific applications. The result is that if the code, for array-based operations, is written efficiently and it is able to achieve satisfactory parallel performance. As mentioned in 2.2.1.1 Python is good for combination with other programming languages, such as C and

C++, which means that great serial and parallel performance can be achieved by writing small, well defined, critical parts of a code in a lower level language.

The Cai's and Xing's paper [41] compare Python with several languages, such as MatLab, Octave, Fortran, C and others, since each of these has its own advantages in term of performance and execution time. Python appear to be one of the most competitive alternatives for scientific computations. Compared with MatLab, Python also has the feature of numerical and visualization modules, which makes it so influential and dynamic. As an object-oriented language that allows handling of errors and failures, mixed-language support and cross-platform interface, it is possible to write highly readable code 2.2.1.1. Unlike MatLab, Python's creation of classes is more convenient, which is one of the reasons that researchers are using interpreted scripting language over a compiled one.

Python has been applied in various range of areas, not only in scientific researches. It can been use for web scraping, system administration, web development, distributed systems, computational steering, search engines and many others. [41] Scientific researches need a lot of background from different areas of exploration so that they can conclude the most accurate results. Powerful software tool that enables a broad range of tasks, as the ones mentioned in the previous paragraph, will be effective and efficient engine for achieving unique outcomes.

Since, the core of Python is not sufficient for scientific computations with complex and demanding computations, with the creation of NumPy library, array data structures and long nested loops are processed faster and more smooth than before. It, also, includes multi-dimensional array structures with a lot of methods for accessing or changing them. NumPy's features appear to be as a "mirror" of MatLab's ones.

The paper, also, investigates the capabilities of Python for parallel programming applications [41] With a lot of examples given, Python can be used in parallel computations, where the issues of data partitioning and structuring can be handled. It is fast enough for most computational tasks. Its readability and the re-usability of the code hide the fact that Python has reduced speed compared to other languages.

Python is a compiled language and Xing Caia's paper [41] is proving that some programming languages, such as C++ might be more flexible and elegant than Python, but *"...Python is much more convenient, and convenience seems to be a key issue when scientists choose an interpreted scripting language over a compiled language..."* [41] The research is evaluating the performance of parallel Python code in a real scientific application, which is compared to C code. The both serial and parallel performances are equal. In terms of efficiency and optimization Python is helpful for mathematical functions.

2.2.1.3 SciPy conferences

In the previous section we analyze the usage of Python in scientific computing. But how does researchers and scientists see the Python code? What kind of experience they gain

from it? This section is covering several SciPy conferences with real life examples, experiences and opinion of people that are using Python in the field of scientific programming.

For some people, it is not clear why Python has become the language of choice for so many people in scientific computing. Maybe if researchers like Travis Oliphant had decided to use some other language for scientific programming years ago, wed all be using that language now. Python wasnt intended to be a scientific programming language. And as Jake VanderPlas points out in his keynote, Python still is not a scientific programming language, but the foundation for a scientific programming stack. Maybe Pythons strength is that its not a scientific language. It has drawn more computer scientists to contribute to the core language than it would have if it had been more of a domain-specific language.

John Cook reported his experience of migrating from Ruby and other scripting languages to Python. As a mathematician, he needed to start doing computational functions. He describes that period of transferring as *"a rude awakening"*.[42] As soon as he started programming on Python, he found that the language provides *"a hoard of code"* [42], which includes a lot of libraries for various kind of data analysis. The comparison that he makes between languages, such as MatLab and R, and Python is the best description of the difference between them:

"Id rather do mathematics in a general programming language than do general programming in a mathematical language."

He is aware that the most acknowledged disadvantage of Python is its lack of speed, but for John Cook finds even harder rewriting code in other language.

The conference, by Kelsey Jordahl [51], provides a tutorial on open source tools for using geospatial data in Python. It is a great example of Python being useful and incredibly effective for analyzing data in a specific area of study. The tutorial is going through the fundamental geological libraries of Python, such as reading vector data with Fiona - minimalist python package for reading (and writing) vector data in python, and geometry with Shapely - Python library for geometric operations using the GEOS library.

Another helpful SciPy conference is by Mike McKerns [51]. It is a tutorial that provides examples and essential performance tips for writing effective parallel Python - one more prove for the usefulness of Python. The tutorial could be found on GitHub [18]. The conference is supporting the findings in the Cai's and Xing's paper [41]. SciPy conferences are not only great for teaching and practicing Python coding, but also they going through important research findings. [20]

2.2.2 Summary

This literature review is investigating the usage of Python and it is showing how easy and efficient it is for your daily computational work. Python can be of great help even in small scripts with numerous data structures, classes, nested loops, documentation and others. Python has open source libraries, such as Numpy, SciPy, IPython for interactive work

and MatPlotLib for plotting, which makes writing code quickly for machine learning and artificial intelligence. For a small amount of time, everyone can learn Python and apply powerful data structures and design patterns when needed.

2.3 Laboratory Notebooks in science

Creating and writing a research is a challenging process. It is extremely difficult for the author to reach and present unique findings or conclusions. If the analysis for the research have been completed, the scientist has to investigate appropriate arguments and evidence for supporting the ideas which he wants to prove with the research. An effective and well conducted research takes long time to be achieved. Recording all of the findings, ideas and approaches, made during the investigation of the research, is an necessary step for publishing and presenting the analysis. Nonetheless, documenting a project or scientific discovery is challenging and it has to be properly presented to the reader so that it would be understandable. Going thought this process, the analyst might find faults, errors or some unnecessary steps done during investigation, and they be the cause of invalid results. Therefore, scientists have found different forms to report project measurements and practices.[49]

In the past, the analysis in a scientific research were computed quite slowly with a few number of tests for catching bugs, errors and faulty results. The setting up has also causing issues with the output since the researchers were using various tools for achieving different computations. During the investigation of the research, scientists were cautious to write accurate reports for their ideas, findings and results. The best approach for recording the analyses at that time was handwritten lab notebooks. Researches connected with complex computations, such as finance, mathematics and informatics need a lot of support for maintaining huge amount of code and data, which are used for calculations and experiments. Documenting and recording the results will be of use for comparison of the data quality in future researches and for learning from past mistakes and failures.[48] Here are some of the challenges that scientists have to complete so the research can achieve a success:[48]

- In the scientific method of a project, testing the hypothesis is one of the most important steps. Therefore researchers have to frequently alter their work and re-execute so that the data will be up-to-date. The issue that occur in this case is forgetting which change generates a particular output.
- Accomplishing a research is a time and effort consuming process. The background research is necessary for scientists to understand the problem and to come up with variety of ideas. As in the first point, remembering all of the sources that inspire ones idea is a troublesome task.
- There is a struggle to maintain up-to-date notes. Researchers need to be able to go back into old versions of the data, so that they can compare and analyze the approaches that they use.

- The results that scientists that need to achieve have to as accurate as possible. They run various programs and software applications, that create data models, which require a lot of complex computations.

There a various ways to handle problems like the ones mentioned above. Scientists have found that using handwritten laboratory notebooks helps with tracking the history of the code alternations. Researchers are, also, taking notes by using simple text files or sticky note tools. However, scientists have to take care of the issue with organizing and connecting these electronic notes together with the handwritten report. The main drawback is that, there is no easy way of editing or executing the code. Even with the paper notebook, it is quite challenging to trail past results. Therefore, electronic notebooks were created - they cope with various issues encountered in the data collection of a research and they are fast in complex analysis.

One of the most crucial task in researches is to test how much the collected information is correct and accurate, which can be done with numerical analysis. Nowadays, computational experiments have been done in projects - e.g. software development needs computational analysis by evaluating a prototype on a specific number of users or modelling relations between data items. Theoretical work requires symbolic and numerical support as well - e.g. how much of the sources were able to prove a specific concept. Almost every scientific paper is finding an effective usage of computer science applications.

For many years researchers have tried to gather data in the fastest, most effective and most structured approach. As the boost of software tools usage in scientific researches, investigators are creating more and more numerical analysis and are able to clearly and efficiently organise their notes. Here is one suggested course of actions given by Perez, Fernando and Granger [62]. In the list below are described the phases of the process. [62].

Individual exploration
Testing ideas, questions or algorithms for a test data set.

Collaboration
If the hypothesis appears to be effective, then the investigators finds others collaborators to expand the research.

Production-scale execution
Manipulating and managing big data domains on clusters, supercomputers or cloud computing software.

Publication
Results that are presented to colleagues or researchers in the same field of study.

Education
Continuing the research as part of a teaching program or in a process of more unique results and findings.

The life-cycle of the scientific research can not be traversed with only one software tool. Investigators are using a large number of applications that are supporting each of the phases of the analysis. As a consequence, issues are rising - reduced quality, reproducibility and robustness. The scientists are unable to reuse and maintain the code - the same computational analysis might be useful in some other researches and, also, software changes rapidly and needs constant updates. Testing if the results are correct includes testing of all the pieces of code created for the project. Keeping track of the data quality and accuracy requires well structured and written documentation, which is flawless in describing the entire project's workflow.

The Frey's and Bird's paper [46] investigates the curation of research chemistry data - data management [29], organization and integration, and taking its development during the last ten years and predicting what the information might be like for the next ten years. Frey and Bird [46] are reaching the conclusion that technologies of the Semantic Web have improved, they are becoming available and various techniques for the automated computations are being designed frequently. However, there is a continuing for improving the old software tools or creating new ones.[46]

Researchers have found a software tool that helps with the analysis over data quality, share and reproducing - IPython notebook. The application is effective in teaching, collaboration with others, combining text and code and recording notes. IPython has been established by scientists as an interactive tool for journals and support for keeping track of thoughts and ideas. In the research *Interactive notebooks: Sharing the code* by *Helen Shen*, IPython has been described to have an *"easy interface"* and it is creating better communication between collaborators.[67]

Through IPython notebook, coders are able to express their ideas by combining code and text at the same time. The interactive IPython command window support users to run code, access variables and data sets, view graphs and charts.[62]

To sum up, the open source IPython project presents a solution to several issues encountered in researches and it is a single software tool enabling the spanning of the entire life-cycle of computational analysis.

2.4 IPython in Scientific Programming

One of the field that we are going to investigate with this project is the usage of one particular tool - IPython notebook. As mentioned in the Problem Statement section 1.1, IPython has become powerful coding tool for various areas in researches, which leads to the questions - is it helpful?; how many researches and explorations are using IPython, and in what way?

2.4.1 What is IPython Notebook

Created in 2011, by Fernando Prez, a data researcher at the University of California and computational physicist Brian Granger at California Polytechnic State University, IPython has become an effective tool for numerical analysis and data visualizations.[67] It is a Python environment, which is HTML-based, similar to Mathematica - *"a symbolic mathematical computation program used in many scientific, engineering, mathematical, and computing fields"*. [34] [40] Also, IPython is an interactive tool that uses cells, in which the calculations can be computed, formulated and even documented. The software application runs locally open from a web browser - the command *$ ipython notebook* is a new browser window where all noteboks can be seen and the developer can run it from any directory that s/he wants. [50]

IPython notebook has to main elements:[7]

- **Web application** - a browser-based tool for interactive management of documents which mixes text, functions, computations and visualisations, such as graphs and charts.
- **Notebook documents** -a illustration of all the information represented in the web application, including inputs and outputs of the text, functions, computations and visualisations of results.

IPython is stored as a file in JSON format - data objects are in the attribute-value pairs composition, and it is containing the Python programming language with various modules by which a developer can program in the notebook script. The unique feature of IPython is that code can be combined with the code results and text. However, it is not a stand-alone Python application and IPython notebook server installation is required.

Another feature of IPython is the capability of exporting files into different formats, such as pdf, html, LaTeX. Moreover, the IPython Notebook Viewer (nbviewer) allows a sharing options for the scripts that have been created. The function "loads the notebook document from the URL and renders it as a static web page".[6] The nbviewer feature can be presented as nbconvert in a web-based format, with which it could be created static files.

2.4.2 Why IPython Notebook was created

Data scientists are challenged with the creation of an understandable programs' descriptions, used for explaining computations in their researches. There are various reasons for this, but one of them is connected with the constant and gradual style of the analysts to explain new ideas and concepts. As a result, various versions of code that are connected between each other and shows different findings, are created. However, it is demanding for the reader to track all of the scripts which often are not well detailed. Granger has said: *"a high-level description of the algorithm that goes into the paper is light years away from the*

details that are written in the code. Without those details, there is no way that someone could reproduce it in a reasonable time scale." [67]

IPython was created to be an agile tool for scientific programming and data investigation. It supports a reproducible research, since the inputs and outputs are saved as ".ipynb" notebooks. IPython is the interactive approach of programming on Python - it is providing a web-based application of the console-based way, handling all of the scientific process: implementing functionaly, documenting and recording, running and presenting the outcomes. As a consequence of that, executable code can contains analytical text explaining mathematical functions, and results can have detailed visualisations. IPython can supply records of sessions, notebooks and findings. Since the notebooks script are documents internally JSON files saved as .ipynb files, they can be shared in version-control and with other researchers. [7]

2.4.3 Similar Software Tools

There are various applications that are used in data analysis and assessment in science researches, such as Mathematica - as mentioned in 2.4.1, and Maple - "supports mathematical structures as fundamental types, such as polynomials, complex numbers, and vectors, and it allows you to easily construct and test for more complicated structures".[34] [14] They are both analysis packages widely used in scientific programming and notebooks programs. Another software tool is MatLab - high-level language, which gives the opportunity to investigate and visualize and share ideas across fields of study including signal and image processing, managements and maintenance systems, communications, and computational finance.[15] It, also, supports notebook applications.

There are various notebooks and notebooks programs that are uploaded online as an open-source projects - e.g. knitr, which works with the R software language. R is one more language that is widely used for scientific computations and data analysis.[13] To continue with the numerous examples of statistical programming tools, Sage is also a mathematical software system, which is Python-based and it has it's own notebook, and Dexy - software system for creating documents and visualizations for results' objects.[16][12][67]

2.4.4 Contributions

Several scientific projects have exploited IPython as a platform rather than as an end-user application. Although the vast majority of IPython users do little customization beyond setting a few personal options, these projects show that there is a real use case for open, customizable interactive environments in scientific computing:[61]

- IPython is applied for astronomical image analysis from the Space Telescope Science Institute, which is a free-standing science center. Its PyRAF, which is a command language based on the Python scripting language, uses IPython and it provides a shell customized for easier interaction. [25]

- The National Radio Astronomy Observatorys Common Astronomy Software Applications, also, uses IPython. Its work includes C++ tools combined together with an IPython interface, which are applied as a data reduction tasks.[23] As mentioned in the CASA web site: *"this structure provides flexibility to process the data via task interface or as a python script. In addition to the data reduction tasks, many post-processing tools are available for even more flexibility and special purpose reduction needs."*[23]
- The Pymerase project is a tool that creates a python object model, relational database, and an object-relational model connecting the two. It uses IPython for microarray gene expression databases.[24]

More examples of projects using IPython can be found on GitHub [4], such as the Ganga system for job definition and management [2], the SimpleCV computer vision project - an open source framework for building computer vision applications [21], and the Nengo project for simulating large-scale neural systems [10].

2.5 Open Source Repositories

Open source repositories are software projects made available through the Internet.[39] GitHub is one of the most widely and largely used code host services, with more than ten million repositories.[32][52] It allows viewing the activity within projects, such as issues, commits and documentation. In recent years, the web site has present an effective way of sharing scientific researches and contributing to them. The great number of public data and the API of GitHub, has made easy for scientists to analyze project data. There are, also, different tools and data domains that support the researchers' tasks.[47]

The *IPython Observatory* project has chosen GitHub as a source of information that will be investigated for the idea of how IPython is used in science. One of the most crucial features of Git that makes it effective to science is the recorded history of changes into a repositories and that they are available to view from everyone. [63] Furthermore, GItHub presents an option for developers to show their works to colleagues, peers and potential recruiters. A lot of researchers are analysing the usage of programming tools on GitHub and what effect does the version control hosting site has on the improvement of software systems. The Kalliamvakou's paper [52] is investigating possible threats to the validity of researches involving software projects hosted on GitHub. They can be viewed as fundamental steps for analysing all of the repositories connected with IPython notebooks. Figure 2 is showing the results of the paper.

3 Proposed Approach and Work Plan

The purpose of this research is to understand the use of IPython as an instrument of scientific research. To do this, the research will develop analyses of IPython projects stored

Peril		Description
Project Related		
I	A repository is not necessarily a project	A project is typically part of a network of repositories: at least one of them will be designated as central, where code is expected to flow to and where the latest version of the code is to be found.
II	Most projects have low activity	Most projects have very few commits.
III	Most projects are inactive	Most projects do not have recent activity (only 13% of projects have been active in the last month).
IV	Many projects are not software development	A large portion of projects are not used for software development activities.
V	Most projects are personal	More than two thirds of projects (71.6% of repositories) have only have one committer: its owner.
VI	Many active projects do not use GitHub exclusively	Many active projects do not conduct all their software development activities in GitHub.
VII	Few projects use pull requests	Only a fraction of projects use pull requests. And of those that use them, their use is very skewed.
Pull Requests Related		
VIII	Merges only track successful code	If the commits in a pull request are reworked (in response to comments), GitHub records only the commits that are the result of the peer review, not the original commits.
IX	Many merged pull requests appear as non-merged	Only pull requests merged via the "Merge" button are marked as merged. But pull requests can also be merged via other methods, such as using `git` outside GitHub; in those cases, the pull-request will not appear as merged.
User Related		
X	Not all activity is due to registered users	The activity in GitHub repositories is sometimes due to non-users; in some cases, the activity of a user is not properly associated with her account.
XI	Only the user's public activity is visible	Approximately half of GitHub's registered users do not work in public repositories.
Github Related		
XII	GitHub's API does not expose all data	The GitHub API exposes either a subset of events or entities, or a subset of the information regarding the event or the entity.
XIII	Github is continuously evolving	GitHub continues to evolve and it has changed some features and provided new ones. Similarly, the projects evolve and are capable of changing their own history.

Figure 2: The Promises and Perils of Mining GitHub Results

on the GitHub open source repository. The first part of the approach is connected with high-level explanation of the project's concept - trying to identify the fundamental points of the IPython's usage. The second part is explaining the metrics done for supporting our approach.

In order to understand and investigate in detail the reproducibility, replication and the nature of IPython scripts, the research is giving high-level description of the main idea by stating the following questions, which later will be supported by various metrics, such as string manipulations and readability analysis:

- how readable is the code?
- how complete is the documentation?
- how many commits over a certain period of time have been made?
- what is the frequency of commits, pull and other requests?
- what is the log message usefulness?
- how many projects have licences - what form do they take?
- how many projects are linked to research papers?
- what are the different fields of science using iPython?
- how long is a typical script (and what variation)?
- is there a stereotypical 'template' for how a notebook script is presented?
- how long the a project is developed?
- is there any other software tools used in a specific repository?
- what type of risks did IPython scripts cost for the implementation of a project?

The findings from the analysis might help with the discovery of features for investigation. Furthermore, the steps that we need to take for the realization of the project and structuring it in a understandable course of actions, are:

1. Search GitHub results for IPython projects.
2. Download a sample of projects meeting criteria.
3. Discriminate between end-user scripts and IPython extensions.
4. Execute analytical scripts on code.
5. Along the way, implement tests for the functionality.

6. Download all of the data for the projects.

7. Test the analytical scripts by running them on the bigger data set.

8. Test if the analytical scripts will work for data from other open source repositories' information, such as BitBucket [1].

One of the first stages of the project is to download sample information from GitHub - all of the results from the GitHub search for IPython [3]. For now, only three repositories that are using IPython are extracted from the API. However, there were challenges met along the way. One of them, which has not been implemented yet, is that the search showing only one page, which includes maximum thirty results - when downloading information from the API, the algorithm has to traverse through all of the pages and download with respect to the limits given from GitHub - *"the rate limit allows you to make up to ten requests per minute"*[5]. Another challenge that we have to consider and, also, it has not been done yet, is the usage of an additional server from the University of Glasgow for storing all of the repositories' content.

Moreover, functionality for finding the overall documentation for a repository has been implemented - usually the *"README.md"* file in a repository is containing a description of the a project, but not all of them are called with the same name and are not in same folder. An automated method was designed for traversing through the structure of the repositories, but it is finding only the *"README.md"* files, which is requires improvements. Readability analysis are created for the *"README.md"* files - it is calculating number of syllables, lexicons, sentences, Flesch Reading Ease Score - measures textual difficulty, which indicates how easy a text is to read [30], Flesch Kincaid Grade level - the number of years of education generally required to understand a text [31], Gunning fog scale - measures the readability of English writing [33], Smog analysis - is a measure of readability that estimates the years of education needed to understand a piece of writing [37], and formulas gauging the understandability of a text, such as Coleman-Liau Formula [28], Automated readability index [27]. However, it is traversing only through one of the downloaded repository projects and this functionality needs rework so that it can produce results for all folders. Important aspects that we have to consider during these part of the analysis, are the perils of GitHub mentioned on figure 2. The improvement of these steps should take not more than two weeks.

The fourth stage should be the most time-consuming - around one month, since the fifth one has to be implemented in concurrence with it. The third step should be complete in three weeks, the sixth and seventh - in two weeks and the eight - again, in two weeks. The time left would be left for writing the dissertation and for creating more functionality. Overall, the predicted time for the implementation of the project is 13 weeks. The suggested approach might be extended or changed depending from the results and the challenges that are yet to be encountered.

References

[1] Bitbucket official web page. https://bitbucket.org/.

[2] Ganga official web page. http://ganga.web.cern.ch/ganga/.

[3] Github api. https://developer.github.com/v3/).

[4] Github repositories using ipython. https://github.com/ipython/ipython/wiki/Projects-using-IPython).

[5] Github traverse search results. https://developer.github.com/guides/traversing-with-pagination/).

[6] Ipython - definition. urlhttp://ipython.org/notebook.html).

[7] Ipython - features. urlhttps://ipython.org/ipython-doc/1/interactive/notebook.html).

[8] Matplotlib official web page. http://matplotlib.org/.

[9] Met office official website. urlhttp://www.metoffice.gov.uk/climate-change/resources/hadleycentre.

[10] Nengo project official web page. http://nengo.ca/docs/html/advanced/ipython_notebook.html.

[11] Numpy official web page. http://www.numpy.org/.

[12] Official web site of dexy. urlhttp://www.dexy.it/).

[13] Official web site of knitr. urlhttp://yihui.name/knitr/).

[14] Official web site of mapple. urlhttp://www.maplesoft.com/products/maple/compare/programming.a

[15] Official web site of matlab. urlhttp://uk.mathworks.com/products/matlab/?requestedDomain=www

[16] Official web site of sagemath. urlhttp://www.sagemath.org/).

[17] Opencv official web iste. http://opencv.org/).

[18] Scipy conference - python for parallel programming. https://github.com/mmckerns/tuthpc/blob/master/README.md.

[19] Scipy official web page. http://www.scipy.org/.

[20] Scipy official web site. urlhttp://www.scipy.org/.

[21] Simplecv official web page. http://www.simplecv.org/.

[22] Sklearn official web iste. http://scikit-learn.org/stable/tutorial/basic/tutorial.html).

[23] Web site of common astronomy software applications. http://casa.nrao.edu/).

[24] Web site of pymerase. http://pymerase.sourceforge.net/).

[25] Web site of pyraf. http://www.stsci.edu/institute/software_hardware/pyraf).

[26] Wikipedia - argos system. urlhttps://en.wikipedia.org/wiki/Argos_system.

[27] Wikipedia - automated readability index. https://en.wikipedia.org/wiki/Automated_readability_ind

[28] Wikipedia - colemanliau index. https://en.wikipedia.org/wiki/Coleman%E2%80%93Liau_index.

[29] Wikipedia - data curation. https://en.wikipedia.org/wiki/Data_curation).

[30] Wikipedia - flesch reading ease score. https://simple.wikipedia.org/wiki/Flesch_Reading_Ease.

[31] Wikipedia - fleschkincaid grade level. https://en.wikipedia.org/wiki/Flesch%E2%80%93Kincaid_rea

[32] Wikipedia - github. urlhttps://en.wikipedia.org/wiki/GitHub.

[33] Wikipedia - gunning fog index. https://en.wikipedia.org/wiki/Gunning_fog_index.

[34] Wikipedia - mathematica. urlhttps://en.wikipedia.org/wiki/Mathematica.

[35] Wikipedia - replication. urlhttps://en.wikipedia.org/wiki/Replication_(statistics).

[36] Wikipedia - reproducibility. urlhttps://en.wikipedia.org/wiki/Reproducibility.

[37] Wikipedia - smog. https://en.wikipedia.org/wiki/SMOG.

[38] Wikipedia - the scientific method. urlhttps://en.wikipedia.org/wiki/Scientific_method.

[39] Wikipedia-open source software(oss). https://en.wikipedia.org/wiki/Open-source_software).

[40] Wolfram - mathematica. urlhttps://www.wolfram.com/mathematica/.

[41] Xing Cai, Hans Petter Langtangen, and Halvard Moe. On the performance of the python programming language for serial and parallel scientific computations. *Scientific Programming*, 13(1):31–56, 2005.

[42] John D. Cook. Python for scientists and engineers. John D. Cook, 2015.

[43] MS Coyne and BJ Godley. Satellite tracking and analysis tool(stat): an integrated system for archiving, analyzing and mapping animal tracking data. *Marine Ecology Progress Series*, 301:1–7, 2005.

[44] Steve M Easterbrook and Timothy C Johns. Engineering the software for understanding climate change. *Computing in Science & Engineering*, 11(6):64–74, 2009.

[45] Hans Fangohr. Python for computational science and engineering. 2015.

[46] Jeremy Frey, Simon J Coles, Colin Bird, and Cerys Willoughby. Collection, curation, citation at source: Publication@ source 10 years on. *International Journal of Digital Curation*, 10(2):1–11, 2015.

[47] Christopher Gandrud. Github: A tool for social data set development and verification in the cloud. *Available at SSRN 2199367*, 2013.

[48] Philip J Guo and Margo Seltzer. Burrito: Wrapping your lab notebook in computational infrastructure. In *TaPP*, 2012.

[49] Frederic Lawrence Holmes, Jürgen Renn, and Hans-Jörg Rheinberger. *Reworking the bench: Research notebooks in the history of science*, volume 7. Springer Science & Business Media, 2003.

[50] J Johansson. Introduction to scientific computing in python. *: http://nbviewer. ipython. org/github/jrjohansson/scientific-python-lectures/blob/master/Lecture-0-Scientific-Computing-with-Python. ipynb*, 2014.

[51] Kelsey Jordahl. Efficient python for high performance parallel computing. Mike McKerns, 2015.

[52] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining github (extended version).

[53] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining github. 2007.

[54] Diane Kelly, Daniel Hook, and Rebecca Sanders. Five recommended practices for computational scientists who write software. *Computing in Science & Engineering*, 11(5):48–53, 2009.

[55] Diane Kelly and Rebecca Sanders. Assessing the quality of scientific software.

[56] Randall J LeVeque, Ian M Mitchell, and Victoria Stodden. Reproducible research for scientific computing: Tools and strategies for changing the culture.

[57] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.", 2012.

[58] Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

[59] Norman H Packard. Lattice models for solidification and aggregation. *First International Symosium for Science on Form*, 1986.

[60] Roger D Peng. Reproducible research in computational science. *Science (New York, Ny)*, 334(6060):1226, 2011.

[61] Fernando Pérez and Brian E Granger. Ipython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29, 2007.

[62] Fernando Perez, Brian E Granger, and CPSL Obispo. An open source framework for interactive, collaborative and reproducible scientific computing and education, 2013.

[63] Karthik Ram. Git can facilitate greater reproducibility and increased transparency in science. *Source code for biology and medicine*, 8(1):7, 2013.

[64] Rebecca Sanders and Diane Kelly. Dealing with risk in scientific software development. *IEEE software*, (4):21–28, 2008.

[65] Michel F Sanner et al. Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1):57–61, 1999.

[66] M.Scott Shell. An introduction to python for scientific computing. 2014.

[67] Helen Shen et al. Interactive notebooks: Sharing the code. *Nature*, 515(7525):151–152, 2014.

[68] Timothy Storer. Bridging the chasm: A survey of software engineering practice in scientifc programming. *Software in Science*, 2015.

[69] Stephen Wolfram. Computer software in science and mathematics. *Scientific American*, 251(3), 1984.