Bioinformatics software for genomic: a systematic review on GitHub

Carlos Giovanny Hidalgo Suarez Corresp., 1 , **Miguel Eduardo Guevara Burbano** 1 , **Victor Andres Bucheli Guerrero** 1 , **Pedro Antonio Moreno Tovar** Corresp. 1

1 Systems and Computer Engineering School, Universidad del Valle, CALI, Valle Del Cauca, Colombia

Corresponding Authors: Carlos Giovanny Hidalgo Suarez, Pedro Antonio Moreno Tovar
Email address: carlos.hidalgo@correounivalle.edu.co, Pedro.moreno@correounivalle.edu.co

Bioinformatics is an interdisciplinary field that develops software methods and tools to understand biological data. Actually, in branches such as genomics, there are a large number of software tools that serve to support different processes such as genetic data sequencing, genomes, biotechnological applications, among other activities, which contribute knowledge to different fields of study such as environment, medicine, energy and others. As a support to the area of genomics and specifically to the field of genomics, we describe and propose a method based on mining software repositories (MSR), which monitors, evaluates and maps the genomic software hosted in the GitHub. We use the VigHub tool to extract meta-data from projects and create visualizations on technological maps. We present a detailed systematic review of the genomic software projects on GitHub, where the highlights of the genomics area are shown. Specifically we show the technological maps of the programming languages most used in the creation of software. The Time-line, where software projects are displayed by category, relevance and programming languages as a function of time. Classification of the repositories by software categories and the most successful genomic software repositories in GitHub according to stars and score. This paper is aimed at bioinformatics researchers that require relevant information about the current state of a specific genomic technology found in GitHub. The method facilitates the identification of ideas, source code, specific data, platforms, applications, scrips and tools that support research and innovation in genomic software projects. The analysis provided in this paper allowed to identify software trends in the area, as well as new perspectives and future technologies.

Bioinformatics software for genomic: a systematic review on GitHub

Carlos G. Hidalgo[1], **Miguel E. Guevara**[1], **Víctor A. Bucheli**[1], **and Pedro A. Moreno**[1]

[1]**Systems and Computer Engineering School,Faculty of Engineering, Universidad del Valle, Cali, Colombia**

Corresponding author:
Pedro A. Moreno[1]

Email address: Pedro.moreno@correounivalle.edu.co

ABSTRACT

Bioinformatics is an interdisciplinary field that develops software methods and tools to understand biological data. Actually, in branches such as genomics, there are a large number of software tools that serve to support different processes such as genetic data sequencing, genomes, biotechnological applications, among other activities, which contribute knowledge to different fields of study such as environment, medicine, energy and others. As a support to the area of genomics and specifically to the field of genomics, we describe and propose a method based on Mining Software Repositories (MSR), which monitors, evaluates and maps the genomic software hosted in the GitHub. We use the VigHub tool to extract meta-data from projects and create visualizations on technological maps.
We present a detailed systematic review of the genomic software projects on GitHub, where the highlights of the genomics area are shown. Specifically we show the technological maps of the programming languages most used in the creation of software. The Time-line, where software projects are displayed by category, relevance and programming languages as a function of time. Classification of the repositories by software categories and the most successful genomic software repositories in GitHub according to stars and score.
This paper is aimed at bioinformatics researchers that require relevant information about the current state of a specific genomic technology found in GitHub. The method facilitates the identification of ideas, source code, specific data, platforms, applications, scripts and tools that support research and innovation in genomic software projects. The analysis provided in this paper allowed to identify software trends in the area, as well as new perspectives and future technologies.

INTRODUCTION

Bioinformatics is an interdisciplinary field that develops software methods and tools to understand biological data. This interdisciplinary field of science combines computer science, mathematics, statistics and engineering to analyze and interpret biological data. Among the many types of biological data, genomic data is one of the most widely analyzed. For that reason, the use of generic and directed software has increased in the last decade, especially with the exponential growth of data provided by next-generation DNA sequencing (NGS) Chawla S et al. (2016). This situation has created numerous challenges associated with the treatment, storage and analysis of information Hodgkinson et al. (2006). On the one hand, there is the challenge of highly specialized computing infrastructure (Hardware), mainly related to High Performance Computing (HPC). And on the other hand, the challenge is related to software development: models and algorithms, which efficiently scale on HPC infrastructures, to manage different amounts of information Stephens Z et al. (2015). Therefore, bioinformatics is a highly attractive research area, where enormous efforts are made in the production of data and software, which has generated a large community of researchers J et al. (2017), companies Bucheli and González (2007), and enthusiasts that are working actively in the development of software, state-of-the-art or trying to identify the current state of software technologies for the Bioinformatic field or branches such as genomics. Cock et al. (2013a); D. (2017); Budd A et al. (2015).

Classically, the software can be classified into the following categories: systems software, app software, web applications and software product line Chapin et al. (2001); Hargitay and Dixon (1991), However, in genomics the development of software is more complex. The use of these types of software depends on the following: the type of bioinformatics tasks that must be carried out, whether those tasks are small, medium or large scale, the amount of data to be processed, calculations and technologies to be implemented, and the broad spectrum of data generated by different levels of structural and functional organization, ranging from DNA sequences to proteins.

Currently, many genomic software projects are being developed on collaborative software platforms. One of them is GitHub [1], this platform is the most popular open source repository among software developers to host their projects, with more than 3.5 million users and more than 67 million repositories, of which more than 45.3 million are active, that is, they have had some activity in the last year Github (2017). GitHub is based on the Git software, in which the software projects are linked and you can have different versions of the project in order to be able to review the historical sources of the source code. GitHub provides a web interface that provides access control and several collaboration functions. GitHub hosts projects from different areas, among which are bioinformatics developments, but they are difficult to find and categorize due to high volume of software projects, which becomes a problem when it is necessary to analyze this information.

The genomics software changes constantly, hence to know the obtain a representation of the current state of the genomic software—state -of-the-art technique, is very important for to carry out research, development and innovation (RDI) processes Reinganum (1989). In consequence, it is important to have tools that can contribute significantly to the technological development of the field. We have developed the VigHub tool (available at http://eiscapp.univalle.edu.co/VigHubjson/) Hidalgo and Bucheli (2017), which takes into account processes based on Mining Software Repositories (MSR) Benavides Velasco C (2006), which takes into account the tasks of technological watch (TW) Benavides Velasco C (2006), this do allow to do analysis and obtain information with value-added, this can be defined as a set of coordinated search actions, treatment (filtering, classification, analysis) and distribution of the information to support the decision-making process on new development projects and technological inventions in genomics. This paper proposes a method to extract relevant information to obtain a state-of-the-art technique. The method is implemented in the VigHub tool allowing to extract the meta-data housed in the repositories of GitHub, to analyze them and obtain 3 technological maps. Map of programming languages most used in the creation of software. Time-line map, where software projects are displayed by category, relevance and programming languages as a function of time. Map of successful projects genomic software repositories in GitHub according to stars and score of each repository. These maps identify the current status and answer the question: what is the status of the software used in the genomic projects on GitHub?The method presented here is a complement to the surveys or systematic reviews based on patents, books, and papers in journals or conferences.

The document is developed in four sections. In section 1, the reader is introduced to the topic and why this paper is developed. In section 2, the proposed method is described in detail. In section 3, the results of testing the method. In section 4, the discussion is presented and, finally, the conclusion.

METHODS

In this section we present the model, which is based on the technology forecasting and technological surveillance exposed in Bucheli and González (2007); León López et al. (2008); León et al. (2006). This practice allows, through systematic processes, capture and analyze information to identify opportunities and support in decision-making.

Specific queries

GitHub data is taken as input for information processing, as mentioned GitHub has a large volume of data that has information related to open source software projects K. (2013); Dabbish et al. (2012). Using the following questions, we want to discover the state of-the-art technique related to genomic software projects.

- What are the genomics projects on GitHub and classifications?

[1] https://GitHub.com/

- What is the programming language used to develop projects in genomic field?
- What is the evolution of genomics in GitHub?
- What is the most successful genomic software on GitHub?

Making of query strings

The different keywords are grouped together and the different ways of referring to genomics or its processes, The keywords we got with the help of experts in this field were: genomics, genome, sequence dna, sequence rna and sequencing.

The keywords obtained are transformed into searches that can be interpreted by the VigHub search engine, in this case chains nested by Boolean operators. Query strings are formed below.

- ”(”genomics” AND ”Software”)”
- ”(”genome” AND ”Software”)”
- ”((”genomics” OR ”genome”) AND ”Software”)”
- ”(”genomics” AND ”sequencing” AND ”Software”)”
- ”((”genomics” AND (”sequence dna” OR ”sequence rna”)) AND ”Software”)”

VigHub tool

We use the VigHub[2] tool that we have previously developed. VigHub is a prototype tool based on a MSR model Hidalgo C. (2016); Hidalgo and Bucheli (2017). It is supported by computational techniques: extract, store, process, analyze and automatic visualization data of platform GitHub. VigHub allows us to obtain the latest technological developments (repositories) specifically in any software field. We use this tool to find technological developments specifically in genomics. Figure 1 (below) shows the architecture and the five stages of the tool process.

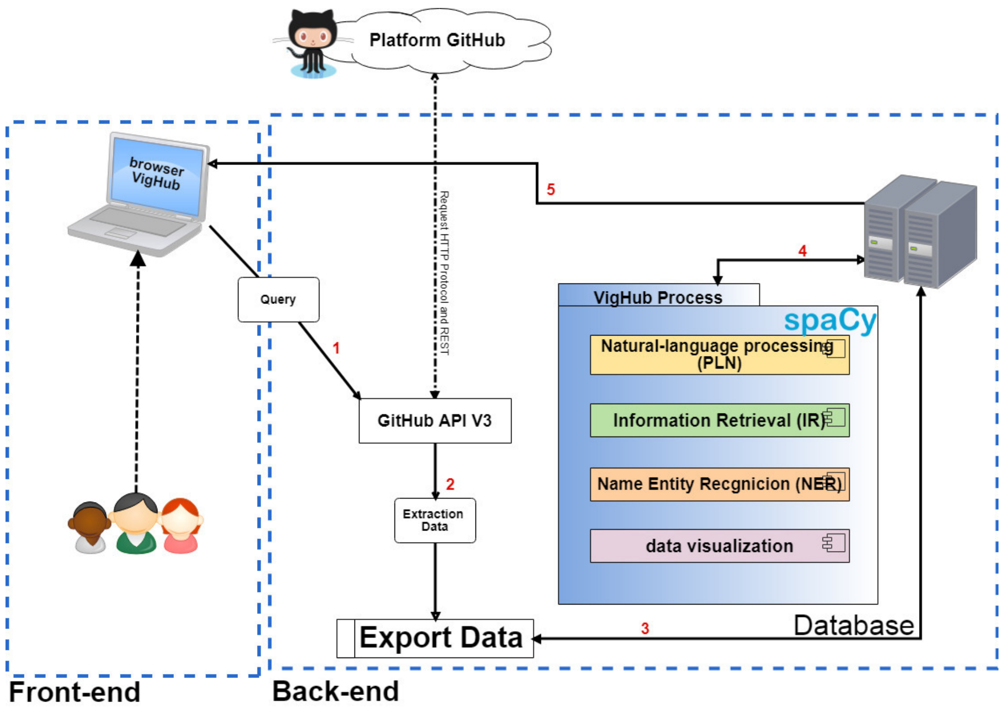

Figure 1. Diagram of VigHub tool components showing a front-end web query and a back-end with computational processes

[2]http://eiscapp.univalle.edu.co/VigHubjson/

1. The system carries out advanced search in GitHub server.
2. The system extracts the data from GitHub (GitHub API).
3. The retrieved projects are stored in a relational database.
4. The indicators and technological maps are performed by: NLP (Natural Language Processing), information retrieval metrics (IR) and DM (Data Mining).
5. Lastly, results are obtained and value-added generating visualizations.

Specialized search with VigHub

The queries in the section ”Making of query strings” were entered in the search engine on the GitHub platform, obtaining 537 results in genomics. The same queries were entered in the search engine of the VigHub tool, obtaining 457 results.

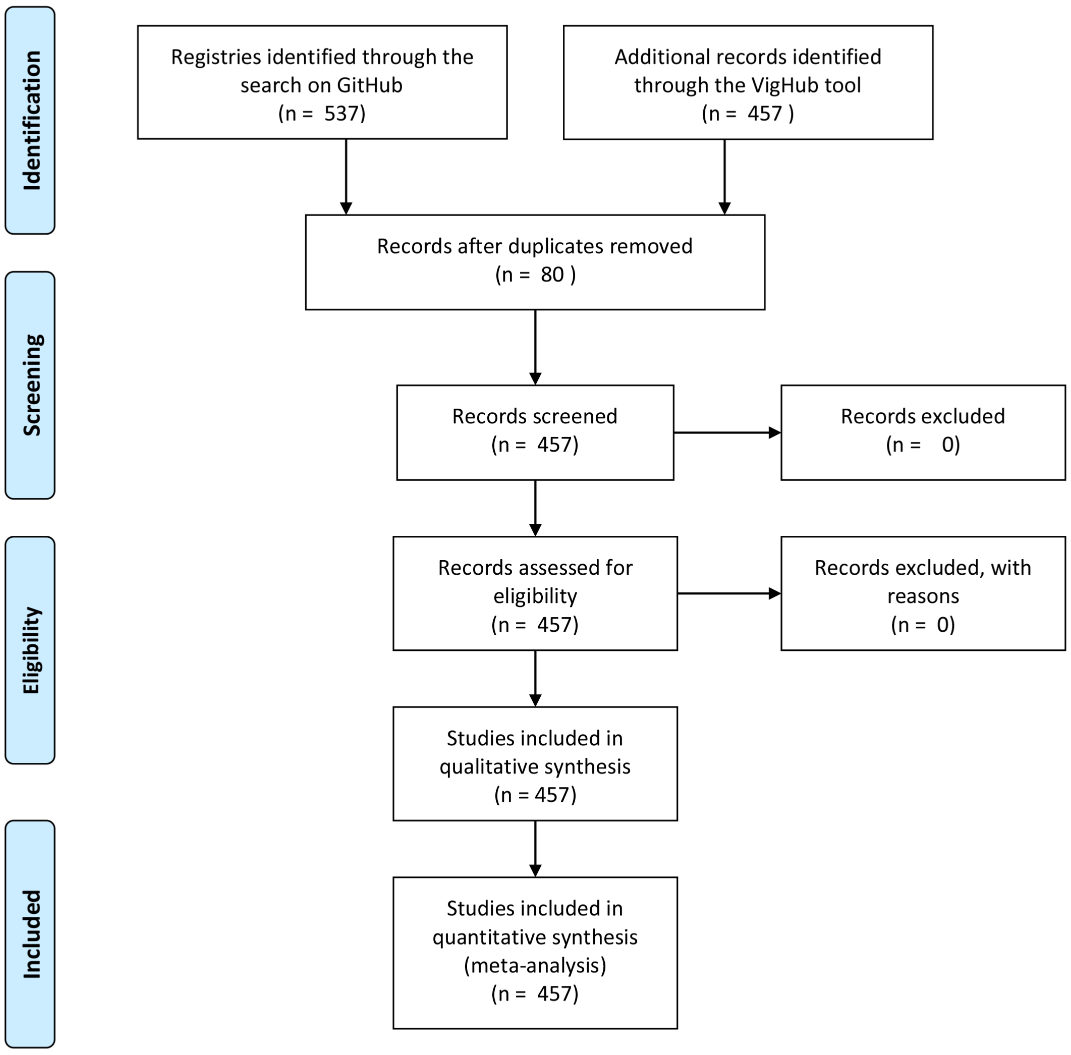
Identification
Screening
Eligibility
Included
Registries identified through the search on GitHub (n = 537)
Additional records identified through the VigHub tool (n = 457)
Records after duplicates removed (n = 80)
Records screened (n = 457)
Records excluded (n = 0)
Records assessed for eligibility (n = 457)
Records excluded, with reasons (n = 0)
Studies included in qualitative synthesis (n = 457)
Studies included in quantitative synthesis (meta-analysis) (n = 457)

Figure 2. Diagram of VigHub tool components showing a front-end web query and a back-end with computational processes

Figure 2 represents the flow chart with the search and selection processes of the included studies. The 80 ($537 - 457 = 80$) results less than GitHub, is because VigHub filters the data for several reasons: its weight in bytes is 0 (empty repositories), they are private repositories (restricted access), it already exists in the query result with the same identification name (ID) and finally the data extraction with the API can't access the URL of the repository.

Construction of maps

In order to find relevant information for data obtained, computational techniques are implemented that support the questions asked in the resolution, with the help of the SpaCy library spacy (2018), This library provides information extraction tasks which use a natural language processing technique (NLP), known as NER, to find and characterize data within repositories, such as programming languages, and to support the techniques. The library also supports information retrieval (IR), applying a technique known as indexing; which makes it possible to see relationships and groups of characteristics in order to locate the found data.

After performing the computational processes, descriptive graphs are obtained in tables, tree maps and bubbles. The Figure 3, shows the relationship of this work with the VigHub tool and reveals how each of the results seen in the paper were obtained. This is how the relevant information is obtained to answer the proposed research questions.

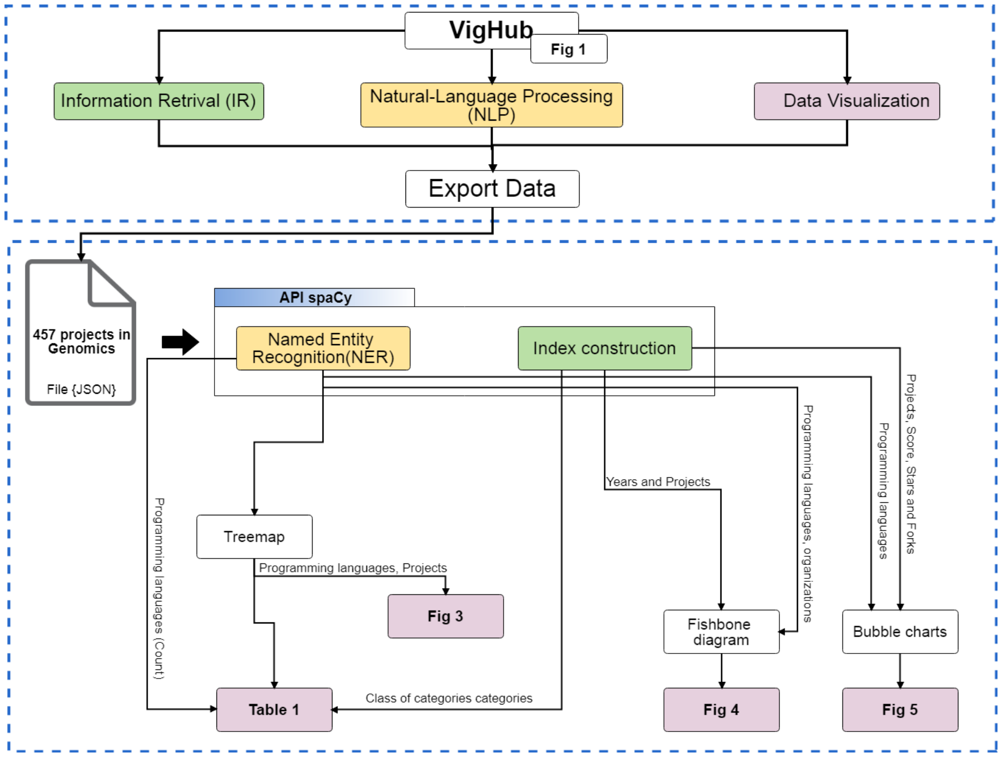
VigHub
Fig 1
Information Retrival (IR)
Natural-Language Processing (NLP)
Data Visualization
Export Data
457 projects in Genomics
File {JSON}
API spaCy
Named Entity Recognition(NER)
Index construction
Programming languages (Count)
Treemap
Programming languages, Projects
Fig 3
Table 1
Class of categories categories
Years and Projects
Programming languages, organizations
Programming languages
Projects, Score, Stars and Forks
Fishbone diagram
Bubble charts
Fig 4
Fig 5

Figure 3. Workflow: shows how the results were obtained from VigHub tool components

1. Programming languages map: with the data obtained from the GitHub API, the application of Named-entity recognition (NER) of the SpaCy library is used, where the language and the name of the repositories is taken from each repository found in the query. Based on this information, a binary vector of co-appearance of programming languages is constructed in the projects. This is in the entry for the agglomerative clustering algorithm Murtagh and Legendre (2014), which creates a hierarchical tree in which the first to appear will be the one with the most similar repositories.

2. Software time-line map: or the construction of this map, four tasks are carried out. The application NER and information retrival (IR) of Spacy is applied, as follows: 1) each repository is classified according to the year of creation and 2) the programming languages of each project are obtained for each year and they are grouped with 3 relevant colors, green, orange and red. Green indicates that it is the most used language. 3) the data of each repository is compared with a corpus that contains the types of software in a predefined way, once this is done, the category with the highest hierarchy

is obtained in each year. 4) with all the data from the repositories matching words are searched , such words are evaluated according to the number of times they appear, this way, the obtaining of the type of technology used in each year is carried out.

in order to know to which category they belong to two tasks were applied: 1) Technique used to extract information from websites, with the practical guide of ancheta (2014) for web scraping in PHP programming language. In which the information of each URL is converted into data strings. 2) A word counter was created, it receives as parameters a range of inputs (data chains) and a range of classes (proposed categories), the coincidence of each repository is identified with the categories, the highest match will be its category.

3. Successful projects map: The map is constructed through the SpaCy library, for this purpose, the indexation of the library is used as follows: feature extraction is applied to the number of stars per repository (value assigned by users to a repository), and the score of each repository (value that GitHub assigns to a repository depending on the number of visits it has). The data is taken and they are positioned from higher to lower value of stars.

   To carry out the pertinent distinction, it has been placed on each node: GitHub user/Repository name. This classification is obtained through the two measures that evaluate the ranking of the repositories: stars and score. The stars are granted by users who recommend the project in a positive way (each user can give a star to each repository). On the other hand the score is ranked according to the score compared to other developers with repositories in the same language or in the same location (by city, by country or worldwide), according to GitHub score is computed for each language using this formula: $sum(stars) + (1.0 - 1.0/count(repositories))$. Subsequently, the normalization of both values is made between 0 and 1 through the formula of min-max scaling Al Shalabi et al. (2006).

RESULTS

Programming languages map

To know the programming languages which are being used to develop genomics helps to determine trends in the market, and current and future technological needs. Taking into account the 457 projects, in Figure 4 shows the least and most relevant projects according to the number of stars, copies and visits that have been received by the platform by users. In this case, the projects that have greatest relevance are Scala and Python+CSS programming languages. Moreover, it has been found that the technological development of genomics on GitHub over the last 9 years, has been carried out using 24 programming languages. Python has been the most common, with 62 projects, followed by C++, Java, C, R and JavaScript with 53, 48, 44, and 40 repositories respectively. The other programming languages such as Ruby, HTML, Jupyter Notebook and Perl have between 30 and 19 repositories. The other programming languages (Go, Scala, Clojure, Num) have between 10 and 1 repository.

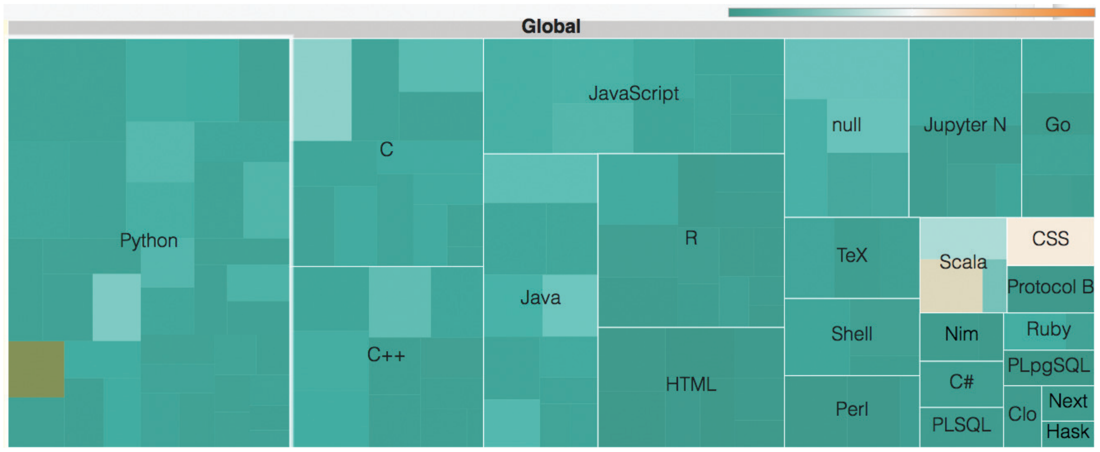

Figure 4. Language and importance of the repository in the subject; according to its size:the higher is the picture that covers a language the greater the amount of developments. According to its color: the orange color identifies the most relevant languages and developments in GitHub.

Software time-line map

Based on an exhaustive analysis of the changes in technologies and paradigms that have taken place between 2009 and 2017, it was possible to obtain a specific perspective which allowed us to identify which have been the most important projects, trend languages and types of technology that have been used for each year. (See Figure 5).

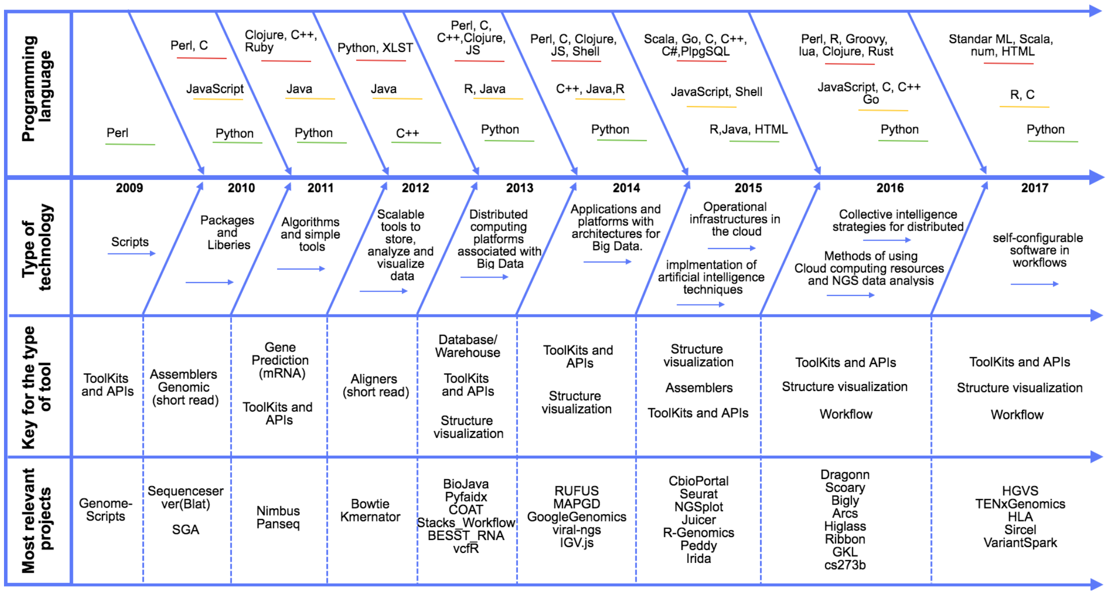

Figure 5. The graph identifies the most relevant language with green, orange and red; green indicates the most used language per year. The most important developments for this research are shown in the top of the line that corresponds to the years. Trends and classification by category for each year are shown at the bottom of each line.

A detailed description of the evolution of software in genomics over time.

- 2009: The first genomics project started, called Genome-scripts. It was developed entirely in Perl and was based on the collection of a set of scripts to format, filter, annotate and analyze fungal genomes.

- 2010: In addition to Perl, developments are added in languages, such as Python, JavaScript and C. The development models of individual scripts are extended to packages that contain libraries of bioinformatic analyzers in order to perform related calculations with assembly, annotation and comparative genomics through heuristic algorithms.

- 2011: Institutions such as the University of Manchester and renowned research centers migrate their repositories to Git. For example, the Broad Institute with GATK start to be interested. More sophisticated algorithms appear this year such as: Random Forest, tool kits for analysis of next generation sequencing data (NGS) and Comparative Genomic Hybridization (CGH) analysis. Python remains popular, but more programming languages begin to appear, for example; ruby, C ++, java, HTML and clojure. Until this year, visualizing had not been mentioned until the appearance of the genomation project. It includes convenience functions in order to visualize and summarize genomic intervals.

- 2012: Data begins to increase and the advances made up to 2012 need optimization and efficiency, which is why they begin to develop scalable tools capable of recovering, storing and analyzing data, as well as visualizing it. In 2012 all projects were able to analyze and visualize genomic data with different purposes. Up to the present, languages such as Perl and C have disappeared, and Java, C ++ and Python have gained strength by creating 5 or 6 projects per year.

- 2013: Considerable growth is seen compared with previous years. There is a large variety of languages, such as; haskell, R, the didactic support platform for programming appears for Python and Jupyter Notebook. In 2013, Perl and C resurface although Python took the lead with 11 projects. This year the trend is distributed computing platforms associated with BigData that use acyclic data flow engines and computation in memory, such as Apache Spark and Avro. There is also a trend towards the use and administration of non-SQL databases and query engines which allow translation with this type of database for the management of material data. Finally, there are libraries dedicated to the visualization of biological and genomic data.

- 2014: The amount of projects and programming languages, such as, Go, Shell, Protocol Buffer and Scala, continues to increase contributing to the field. A strong trend relating to the use of architectures for applications for big data is maintained. Each of the languages already mentioned, such as, Python, java and c ++, have 10, 5 and 6 projects respectively. These are the languages that have contributed the most so far. In 2014, there is a big expansion in some of the companies that are looking for support on their platforms. They are leaving their codes free in order to test and evaluate them. Other companies are generating APIs with the aim that people connect to them and make routines with their codes and data. On the other hand there are companies that are developing tools not only to visualize data but to work with genome variation graphs via the web.

- 2015: The previous year's languages are maintained, and new ones such as PLpgSQL and Matlab have joined. In this period artificial intelligence techniques appear, such as Deep learning, machine learning, clustering algorithms, and infrastructures started using containers or spaces in the cloud (cloud Computing), which run different operating systems based on UNIX, dedicated to analysis, testing, transmission and the visualization of genomic data. Companies are consolidated by creating platforms such as cBioPortal, R toolkit, OCOCO, GRIDSS, that group a variety of functions for the treatment of Genomics. in 2015 it is noteworthy many of the projects offered online courses for both web-based genomics learning and teaching. Another important point is that Python does not appear and those that have developed more are R, Java and HTML, with 4, 4, 5 projects respectively.

- 2016: Shows an increase in projects using new languages and frameworks such as CSS, Groovy, Makefile, Lua, Rust, OpenEdge ABL. The trend for using Python returns again in 2017 with 7 projects. Collective intelligence is used in order to create new distributed systems to work with genomic data, through methods using cloud computing resources, data analysis of NGS, systems biology and microbiome. The collectivity is also adopted by to the scientific-research field where web platforms are created to write papers in a cooperative way generating a link between multiple authors.

- 2017: A self-configurable software in workflows begins to be generated. It is capable of analyzing, formatting, validating, normalizing and assigning sequence variants. Additionally, methodologies for the use of genomic data frames similar to GRanges also appear. In almost all projects, an artificial intelligence technique for genomic variants is used, the term for the area changes to computational genomics. There are fewer languages this year, although Python and R are constant. Additionally, languages like Standard ML and Num begin to emerge. These languages, unlike many other programming languages, have a formal specification, such as operational semantics. This means that the meaning of the construction of a language is specified by the computation that into it when it is executed on a hypothetical machine. Operational semantics is more concerned with ”how” programs are executed rather than emphasizing the results.

Software classification for genomics on GitHub

The Table 1, show results were classified according to categories proposed by Holland R (2014), Out of the 457 projects it is important to highlight that:

- Most projects are focused on visualizing genomic data. These are developed in the JavaScript and Python language supported with HTML and CSS.
- The projects that are of high level processes tend to use python, but when they are of low level processes languages like C, Scala and Ruby are used.
- The databases are no longer traditional (relational model), languages such as R are used to perform semantic extensions, such as RDF

Categories	Total projects	Trend programming language
Structure visualization	74	JavaScript,R,Python,CSS,HTML
Databases	53	PLSQL,R
Sequencing tools	44	Python,R
Genomic assemblers	37	R,C++,Perl,Shell
Multiple sequence aligners	32	Python,Ruby,Scala,Shell
Aligners (short read)	28	Python,C,C++
Genome search engines	25	HTML, JavaScript, Python
Workflows	21	R, PLSQL
Aligners (pairwise)	15	Scala, C
Gene prediction (ncRNA)	14	Ruby, C
Gene prediction (mRNA)	10	C,C++,Perl
unclassified	5	HTML, CSS, Clojure

Table 1. Classification of projects by category.

Successful projects map

It is important to know which projects are the most significant on GitHub, in order to find out what trends there are in the development of the subject. The Figure 6 shows the developments that have been found. The most important aspects are as follows:

- Google/DeepVariant - Python DeepVariant (2017); Poplin et al. (2016): Developed in python, by Google uses a deep neural network to call genetic variants in DNA sequencing data NGS
- GalaxyProject/Galaxy - Python Afgan et al. (2018); Cock et al. (2013b): Developed in Python, is a platform for massive biological data analysis, currently they have dedicated their efforts to build interface applications (API), which make it easy for the user to make use of their HPC resources and self-configurable software.
- BigdataGenomics/ADAM - Scala Massie et al. (2013): Developed in Scala, ADAM is a library and command line tool that allows the use of Apache Spark to parallelize the analysis of genomic data in cluster / cloud computing environments. In a single node, ADAM offers competitive performance for optimized multi-threaded tools, while enabling scaling to clusters with over a thousand cores. ADAM APIs can be used from Scala, Java, Python, R and SQL.

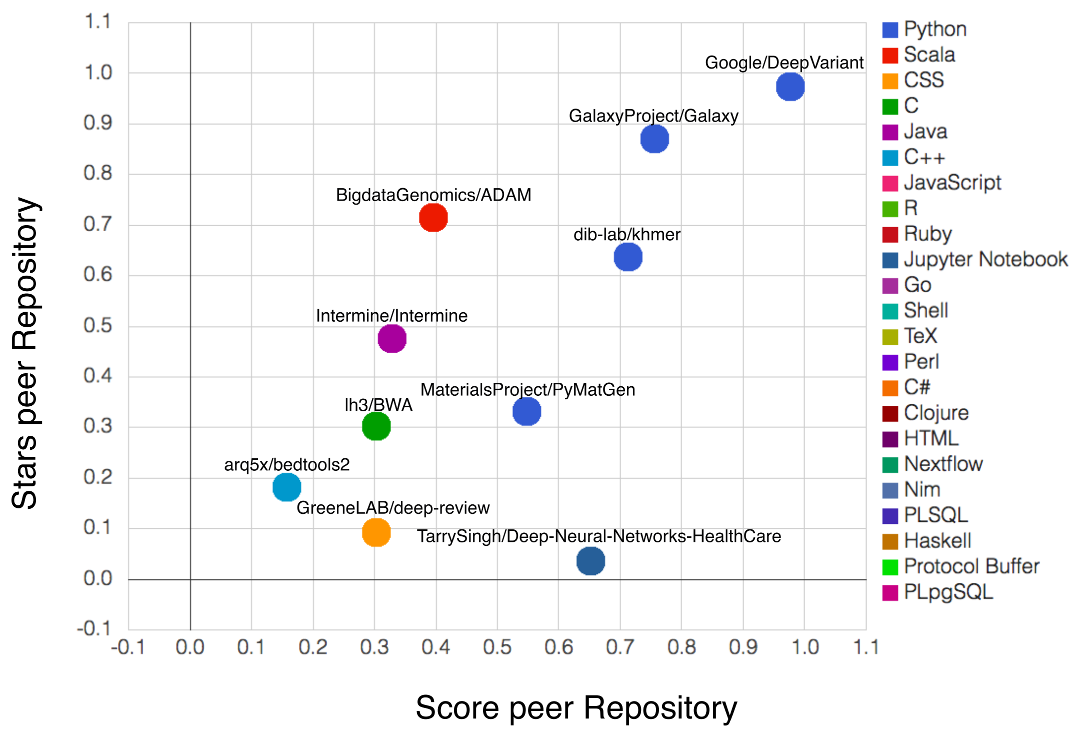

Figure 6. Most successful repositories on GitHub according to the normalized values of stars and the score of each repository.

- did-lab/khmer – Python Crusoe et al. (2015); MacManes (2018): Developed in Python, for its execution of in-memory counting, it filters K-mer structures in nucleotide sequence through networks.

- intermine/intermine - JAVA Smith et al. (2012): Developed in JAVA, it is a powerful open source data storage system. InterMine allows users to integrate multiple data sources with minimal effort, providing powerful web services and a sleek web application with minimal configuration.

- MaterialsProject/PyMatGen - Python Ong et al. (2013): A robust open source Python library for material analysis.

- lh3/BWA - C Li and Durbin (2009): BWA is a software package for mapping DNA sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. It is the first algorithm that is designed for Illumina sequence readings up to 100 bp.

- arq5x/bedtools2 – C++ Quinlan and Hall (2010): Developed in C++, is a collection of utilities like a swiss-army knife of tools for a wide-range of genomics analysis tasks. The most widely-used tools enable genome arithmetic: that is, set theory on the genome.

- GreeneLAB/deep-review – CSS Ching et al. (2018): A collaboratively written review paper on deep learning, genomics, and precision medicine.

- TarrySingh/Deep-Neural-Networks-HealthCare – Jupyter notebook Singh (2017): Tangible and Practical Deep Learning Projects Repository for Healthcare such as Cancer, Drug Discovery, Genomics and more.

DISCUSSION

In this paper, we present a systematic review of genomics software projects in GitHub. We found a strong trends and tendencies for technological maps, programming languages map, software time-line map and successful projects map.

This paper examined and analyzed relevant related information to bioinformatics development projects for use in genomics hosted on the GitHub platform, using the technology watch tool, ”VigHub”. This allowed us to capture, process and transform the information related to the genomics area available in GitHub into useful knowledge, allowing us to know which languages are the most used for the development of technologies, establish a time-line for the development of genomics software, and identify which are the most cited, positioned, copied and visited repositories, that is, which are the most successful genomics projects. These elements can help answer common concerns that may arise when initiating genomics research involving software development.

Python is the most used programming language in bioinformatics

As shown in Figure 4, the programming languages used to develop software for use in genomics were identified over the last 9 years on the GitHub platform. A similar result is also found in other types of studies (See Table 2). The difference between the two types of studies is that the former are the most widely used programming languages in genomics, while the latter are for a broader spectrum of data science.

This study	2017 top Programming languages - IEEE
Phyton	Phyton
C	C
C++	Java
JavaScript	C++
Java	C#
R	R
HTML	JavaScript
TeX	PHP
Shell	Go
Perl	Swift

Table 2. Comparison between the present study versus the 2017 top programming languages in data science.

In both cases, Python is the language that has been the most used in the contribution to the development of software, due to the enormous reception that this language has had in recent years, which highlights its ease of use and legibility of its coding. Python is a multi-paradigm and multi-platform interpreted language that relies heavily on the object-oriented paradigm, but supports imperative and functional programming, these properties make it a widely used candidate for bioinformatics scripting tasks and even its extension to work-flow modeling McKinney (2012); Oliphant (2007). In short, Python helps programmers perform coding in a few steps compared to Java or C++. In fact, a comparative experiment showed that Python requires fewer lines of code compared to other programming languages to implement three standard bioinformatics methods Fourment and Gillings (2008).

Historically, Perl is Python’s predecessor (See Figure 5). However, the choice of the most useful programming language for genomics/bioinformatics depends on what you want to do or what you know best. Both Python and Perl have strong support modules for Bioinformatics and, of course, both are powerful data processing languages Stajich et al. (2002); Tisdall (2013, 2003); Schuerer and Letondal (2002).

For beginners in bioinformatics, Python, R and bash are the most useful languages to learn at this time in bioinformatics, because script development is a routine task in bioinformatics. Deciding which one to start with depends on your goals. Recently, a new alliance has emerged between Python and R, where it is expected to have a favourable impact on the development of software for genomics software Fukushima (1980).

Other languages such as java and JavaScript are shown as trends in various API implementations of different platforms. JavaScript, being a highly useful language in web applications and being a predominantly client-side language, is a powerful alternative for the development of data visualization applications (Table 1). Featured CbioPortal projects support its performance and scalability for tools focused on genomics in these languages. On the other hand, Java is a high level language oriented to general purpose objects but with a marked use in client-server type applications. This language has the characteristic of being highly portable since it can be executed in any platform and operating system that has installed the virtual machine of java, one of the outstanding projects in this language is Intermine Smith et al. (2012).

Regarding high level compiled languages such as C and C++, it is worth noting that although they are more demanding in terms of code lines, they provide outstanding performance and better memory management, as well as high integration with system libraries such as message pass-through libraries, which makes these languages very powerful in distributed environments Fourment and Gillings (2008).

Technological trends in bioinformatics tend to increase the computational power and scale of data in genomics

On the other hand, a time-line of corresponding technological trends was detected with the growth of GitHub since its creation in 2009 (See Figure 5). The first repositories with projects associated with their use in genomics were script-based technologies, which progressed the year after the creation of new projects based on software packages. Subsequently, new technologies and programming languages were incorporated, and the need to implement projects with the inclusion of scalable and parallelizable models becomes evident, in addition to the emergence of cloud-based technologies, this is temporarily corresponding to the boom in the massification of data collection of nucleic acid sequences generated by NGS technologies (HiSeq2000 was launched in 2009 and consolidated as a leading technology between 2010 and 2013). It should be noted that the incorporation of automatic learning techniques, such as in-depth learning, is the current development trend with a high volume of downloads, these technologies based on neural networks are not recent, dating from the presentation of Fukushima's work in 1980 Fukushima (1980). With further development, it was demonstrated that these methods are highly computationally demanding and training times tend to be very long, so it is now that the emergence of new technologies such as GPU processing and the development of tensor processors has enabled the popularity of these technologies with excellent results, In other words, the existing computing power and scale of data has allowed the complex multi-layered architecture of deep neural networks to demonstrate that their ”learning curve” is significantly higher than other statistical methods that previously replaced them.

Google DeepVariant and Galaxy are the most successfull genomics projects in bioinformatics

The Figure 6 shows the 10 most successful repositories according to users and the GitHub score. It is noteworthy that 3 of these projects are linked to the deep learning trend, and that the Google DeepVariant project is the most important of them. Given these results, we believe that the trend will continue over the next few years and we even believe that we are at the beginning of an era of technological revolution associated with deep learning.

CONCLUSIONS

This method was designed by the authors Victor A Bucheli and Pedro A Moreno. The implementation of the method within the VigHub tool was carried out by Carlos G Hidalgo, and the search and application of data was carried out by Carlos Hidalgo and Miguel E Guevara. Finally, in the analysis of the data, all the authors actively participated. During the creation of the paper, there were some minor problems related to the lack of data analysis and context, but the leadership and decision making of author Pedro A Moreno were always correct.

This study allowed us to make a systematic review of 457 public genomic software development projects deposited on the GitHub platform. In each of the projects, the main software tools, technological maps (with relevant information for bioinformaticians) and genomists were identified.

The repositories provided by the GitHub platform are progressively becoming the places in which numerous organizations store and organize the results of their activities, which include; software development in all areas of computer science, biology, education, finance, administration and legislation and many more besides.

Finally, we consider these results as strategic for research, development and innovation in the scientific, academic, commercial and industrial communities, since they allow us to diagnose the status of software developments for the creation of new projects in genomics.

ACKNOWLEDGMENTS

We thank the Universidad del Valle (Faculty of Engineering, Systems and Information Engineering School) for providing us with the necessary software resources for our research.

REFERENCES

Afgan, E., Baker, D., Batut, B., Van Den Beek, M., Bouvier, D., Čech, M., Chilton, J., Clements, D., Coraor, N., Grüning, B. A., et al. (2018). The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic acids research*, 46(W1):W3–W10.

Al Shalabi, L., Shaaban, Z., and Kasasbeh, B. (2006). Data mining: A preprocessing engine. *Journal of Computer Science*, 2(9):735–739.

ancheta, W. (2014). php-webscraping.md. https://gist.github.com/anchetaWern/6150297.

Benavides Velasco C, Q. G. C. (2006). Inteligencia competitiva, prospectiva e innovación. la norma ue 166006 ex sobre el sistema de vigilancia tecnológica. *Boletín ICE Económico: Información Comercial Española*, 3(2896):47–63.

Bucheli, V. A. and González, F. A. (2007). Herramienta informática para vigilancia tecnológica-vigtech. *Revista Avances en Sistemas e Informática*, 4(1).

Budd A, Corpas M, B. M. et al. (2015). A quick guide for building a successful bioinformatics community. *PLoS computational biology*, 11(2):e1003972.

Chapin, N., Hale, J. E., Khan, K. M., Ramil, J. F., and Tan, W.-G. (2001). Types of software evolution and software maintenance. *Journal of Software: Evolution and Process*, 13(1):3–30.

Chawla S, Sangwan S, S. R. et al. (2016). Statistical Significance of Biological Data Markets. *International Journal of Engineering Science and Computing*.

Ching, T., Himmelstein, D. S., Beaulieu-Jones, B. K., Kalinin, A. A., Do, B. T., Way, G. P., Ferrero, E., Agapow, P.-M., Zietz, M., Hoffman, M. M., et al. (2018). Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387.

Cock, P. J., Grüning, B. A., Paszkiewicz, K., and Pritchard, L. (2013a). Galaxy tools and workflows for sequence analysis with applications in molecular plant pathology. *PeerJ*, 1:e167.

Cock, P. J., Grüning, B. A., Paszkiewicz, K., and Pritchard, L. (2013b). Galaxy tools and workflows for sequence analysis with applications in molecular plant pathology. *PeerJ*, 1:e167.

Crusoe, M. R., Alameldin, H. F., Awad, S., Boucher, E., Caldwell, A., Cartwright, R., Charbonneau, A., Constantinides, B., Edvenson, G., Fay, S., et al. (2015). The khmer software package: enabling efficient nucleotide sequence analysis. *F1000Research*, 4.

D., D. (2017). Bioinformatics software from india: Current status and challenges. *Journal of Proteins & Proteomics*, 8(3).

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. (2012). Social coding in GitHub. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work - CSCW '12*, page 1277.

DeepVariant (2017). Deepvariant on google cloud platform. urlhttps://github.com/google/deepvariant.

Fourment, M. and Gillings, M. R. (2008). A comparison of common programming languages used in bioinformatics. *BMC Bioinformatics*, 9.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.

Github (2017). Github octoverse 2017. urlhttps://octoverse.github.com/.

Hargitay, S. and Dixon, T. (1991). Categories of software. In *Software Selection for Surveyors*, pages 21–26. Springer.

Hidalgo, C. and Bucheli, V. (2017). Séptimo congreso internacional de computación cicom. In Maily, Q. and Lorena, M., editors, *VIGHUB: herramienta prototipo para el apoyo de la vigilancia tecnológica en el campo de desarrollo del software*, pages 231 – 245. FAbbecor.ong.

Hidalgo C., B. V. (2016). Computing colombian conference. In Society, I. C., editor, *Herramienta tecnológica de VT para GitHub*, pages 131 – 133.

Hodgkinson, J., Van Well, B., Padgett, M., and Pride, R. D. (2006). Modelling and interpretation of gas detection using remote laser pointers. In *Spectrochimica Acta - Part A: Molecular and Biomolecular Spectroscopy*, volume 63, pages 929–939.

Holland R, Karabaliev I, S. W. (2014). Eagle Genomics — The Elements of Bioinformatics. *figshare*.

J, D. L. R., C, B.-M., and Campos-Laborie, F. (2017). Bioinformatics in latin america and soibio impact, a tale of spin-off and expansion around genomes and protein structures. *Briefings in bioinformatics*.

K., F. (2013). Github has surpassed sourceforge and google code in popularity. 2011. *URL http://readwrite.com/2011/06/02/github-has-passed-sourceforge*.

León, A. M., Castellanos, O. F., and Vargas, F. A. (2006). Valoración, selección y pertinencia de herramientas de software utilizadas en vigilancia tecnológica. *Ingeniería e investigación*, 26(1).

León López, A., Castellanos Domínguez, O., and Montañez Franco, V. (2008). Tendencias actuales en el entendimiento de la vigilancia tecnológica como instrumento de inteligencia en la organización.

Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760.

MacManes, M. D. (2018). The Oyster River Protocol: a multi-assembler and kmer approach for de novo transcriptome assembly. *PeerJ*, 6:e5428.

Massie, M., Nothaft, F., Hartl, C., Kozanitis, C., Schumacher, A., Joseph, A. D., and Patterson, D. A. (2013). Adam: Genomics formats and processing patterns for cloud scale computing. *University of California, Berkeley Technical Report, No. UCB/EECS-2013*, 207:2013.

McKinney, W. (2012). Python for Data Analysis. *Climate Change 2013 - The Physical Science Basis*, 53(9):466.

Murtagh, F. and Legendre, P. (2014). Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion? *Journal of classification*, 31(3):274–295.

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science and Engineering*, 9(3):10–20.

Ong, S. P., Richards, W. D., Jain, A., Hautier, G., Kocher, M., Cholia, S., Gunter, D., Chevrier, V. L., Persson, K. A., and Ceder, G. (2013). Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319.

Poplin, R., Newburger, D., Dijamco, J., Nguyen, N., Loy, D., Gross, S. S., McLean, C. Y., and DePristo, M. A. (2016). Creating a universal SNP and small indel variant caller with deep neural networks. *bioRxiv*, page 092890.

Quinlan, A. R. and Hall, I. M. (2010). BEDTools: A flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842.

Reinganum, J. F. (1989). Chapter 14 The timing of innovation: Research, development, and diffusion.

Schuerer, K. and Letondal, C. (2002). Python course in Bioinformatics. *Pasteur Institute*, 2:182.

Singh, T. (2017). Deep-Neural-Networks-HealthCare.

Smith, R. N., Aleksic, J., Butano, D., Carr, A., Contrino, S., Hu, F., Lyne, M., Lyne, R., Kalderimis, A., Rutherford, K., Stepan, R., Sullivan, J., Wakeling, M., Watkins, X., and Micklem, G. (2012). InterMine: A flexible data warehouse system for the integration and analysis of heterogeneous biological data. *Bioinformatics*, 28(23):3163–3165.

spacy (2018). Industrial-strength natural language processing. urlhttps://spacy.io/.

Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., Fuellen, G., Gilbert, J. G., Korf, I., Lapp, H., Lehväslaiho, H., Matsalla, C., Mungall, C. J., Osborne, B. I., Pocock, M. R., Schattner, P., Senger, M., Stein, L. D., Stupka, E., Wilkinson, M. D., and Birney, E. (2002). The Bioperl toolkit: Perl modules for the life sciences. *Genome Research*, 12(10):1611–1618.

Stephens Z, Lee S, F. F. et al. (2015). Big data: Astronomical or genomical? *PLoS Biology*, 13(7):1–11.

Tisdall, J. (2013). Beginning Perl for Bioinformatics.

Tisdall, J. D. (2003). Mastering Perl for Bioinformatics. *Building*, 159:378.