Improving bioinformatics software quality through incorporation of software engineering practices

Adeeb Noor

Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

ABSTRACT

Background: Bioinformatics software is developed for collecting, analyzing, integrating, and interpreting life science datasets that are often enormous. Bioinformatics engineers often lack the software engineering skills necessary for developing robust, maintainable, reusable software. This study presents review and discussion of the findings and efforts made to improve the quality of bioinformatics software.

Methodology: A systematic review was conducted of related literature that identifies core software engineering concepts for improving bioinformatics software development: requirements gathering, documentation, testing, and integration. The findings are presented with the aim of illuminating trends within the research that could lead to viable solutions to the struggles faced by bioinformatics engineers when developing scientific software.

Results: The findings suggest that bioinformatics engineers could significantly benefit from the incorporation of software engineering principles into their development efforts. This leads to suggestion of both cultural changes within bioinformatics research communities as well as adoption of software engineering disciplines into the formal education of bioinformatics engineers. Open management of scientific bioinformatics development projects can result in improved software quality through collaboration amongst both bioinformatics engineers and software engineers.

Conclusions: While strides have been made both in identification and solution of issues of particular import to bioinformatics software development, there is still room for improvement in terms of shifts in both the formal education of bioinformatics engineers as well as the culture and approaches of managing scientific bioinformatics research and development efforts.

INTRODUCTION

Along with the many recent advancements in the life sciences, which include, but are not limited to, advancements in drug design processes and other innovations under the umbrella of biochemistry, the field of bioinformatics has arisen with scientists collecting, integrating, analyzing, and interpreting large life science datasets. The Oxford English
Dictionary defines "(molecular) bio-informatics as the conceptualization of biology in terms of molecules (in the sense of physical chemistry) and the application of "informatics techniques” (derived from disciplines such as applied mathematics, statistics, and computer science) to understand and organize the information associated with these molecules on a large scale. Bioinformatics can be seen as an management information system for molecular biology with many practical applications. According to the National Institutes of Health (NIH), bioinformatics can be described as any research, development, or application that could help scientists visualize, store, analyze, and/or organize biological data (Bayat, 2002). In the healthcare industry, application of bioinformatics techniques has seen the electronic health records annotated with disease diagnoses and medications (Hoffman, 2007) mapped to precision medicines (Tenenbaum, Shah & Altman, 2021) with worldwide development and adoption (Noor, 2019a). In a different direction, computational biology has been applied with great precision to addressing specific tasks, such as the discovery of genes (Althubaiti et al., 2019). There are other examples of the development of specialized software for solving life science problems, such as the prediction of drug–drug interactions (Noor et al., 2017), the design of novel drugs (Assiri & Noor, 2020), and the discovery of new treatment methods for diseases (Noor & Assiri, 2021). These accomplishments and many others have been made possible through advances in computation technology and the growing availability of massive life science datasets (Gupta, Gangotia & Mani, 2021).

Despite the many existing and upcoming applications of bioinformatics that demonstrate great potential for healthcare advancement, bioinformatics scientists themselves have yet, for the most part, to evolve along with the bioinformatics field in terms of acquiring refined skills for developing high-quality, maintainable software (Lawlor & Sleator, 2021). In a review of two popular bio-sequence alignment software packages for bioinformatics, FAST and BLAST (Aryal, 2019; Leimeister et al., 2014; Cashman et al., 2018) concluded that both tools would greatly benefit from improved flexibility, usability, and user-friendliness. Sanders & Kelly (2008), Oulas et al. (2019) and Lawlor & Sleator (2021) reinforced the notion that the complex bioinformatics discipline and its increasing depth and volume of applications necessitate the development of reliable and maintainable software for use and expansion by bioinformatics professionals. In a similar vein, Borgman & Bourne (2021) provided a related discussion of the issues of software data management. Clearly, unreliable and/or unmaintainable software pose a real hindrance to the success and expansion of the application of bioinformatics approaches.

In order to address the issue, it must be recognized that bioinformatics software packages are often conceptually different, ranging from throw-away scripts to one-off applications to reusable platforms, which can greatly affect the software engineering practices that should be adopted. For instance, scientific bioinformatics software has been defined as software developed in the academic setting for the solution of a domain issue (Howison et al., 2015). The development of such software faces many challenges in terms of documentation, testing, continuous integration, and containerization for portability.
One of the fundamental, exacerbating factors to the development of high-quality bioinformatics software is a lack of initiatives in formal education supporting the instillation of software engineering principles into existing and upcoming bioinformatics professionals. This is supported by studies, such as Georgeson et al. (2019), that indicate bioinformatics engineers often apply ad hoc methodologies when developing scientific bioinformatics software. The situation is compounded by the fact that traditional software engineers are usually unfamiliar with the complexities of the life sciences (Koch & Jones, 2016), resulting in bioinformatics engineers needing to develop their own software. Of the problems associated with the development of bioinformatics scientific software, gathering and defining system requirements, producing proper documentation, and eliminating code redundancy are frequent (Segal, 2007) because self-study of coding, rather than formal software development, is most common among bioinformatics engineers (Weston, 2004; Attwood et al., 2019).

Studies in bioinformatics historically began with informal workshops and training then progressed to certification and finally Ph.D. programs (Ranganathan, 2005). Unfortunately, while many of these programs have deeply focused on biological courses and shallowly focused on software development courses with many of the popular programs offering few software engineering courses specific to addressing the complexities of bioinformatics. According to the study “Software Engineering Education for Bioinformatics” (Umarji et al., 2009), only two out of 79 programs considered at popular higher-education institutions within the United States included some type of software engineering course. The study also demonstrated that 85% of bioinformatics students learned to code through self-study. Similar situations have occurred outside of the United States where significant national resources were earmarked for the development of scientific-educational programs, such as Vision 2030 in the Kingdom of Saudi Arabia, yet few opportunities exist for formal studies in bioinformatics at the tertiary level (Noor, 2019b). This phenomenon is further supported by quantities of research showing this lack of education and training among bioinformatics engineers (Baxter et al., 2006; Hannay et al., 2009; Merali, 2010; Joppa et al., 2013; Wilson, 2014; Wilson et al., 2014; Lawlor & Walsh, 2015; List, Ebert & Albrecht, 2017; Mulder et al., 2018). Additionally, a study was conducted that emphasizes the importance of collaboration/education among bioinformaticians and software engineers regarding the development of reliable software (Lawlor & Sleator, 2021). Given the body of research, the need to incorporate formal software engineering principles into the curricula of bioinformatics is evident if there is to be any tangible hope of steady, scientific, efficient progress in the future of bioinformatics.

RATIONALE
There are many established norms and patterns that are detrimental to the production of high-quality, maintainable scientific bioinformatics software, such as funding patterns that reward new development and preclude reuse, as well as the tendency of third-party code to lack sustainability and support for adoption. Issues often occur with bioinformatics software development because of the sophisticated research questions driving the
software development as well as constraints involved in the associated projects such as time and budget. Most bioinformatics engineers benefit from incorporating software engineering skills, regardless of the scale or type of software to be developed. Such skills include documentation, testing, integration, and containerization for portability. Lawlor & Sleator (2021) confirmed the generally low-quality state of bioinformatics software development and emphasized the importance of collaboration among bioinformatics and software engineers for producing reliable software. A recent study (Brandies & Hogg, 2021) emphasized ten rules for undertaking bioinformatics software development.

In light of these efforts, this study presents a review of the relevant literature and suggests that cross-education of bioinformatics engineers in software engineering appears to be the most appropriate course of action for improving scientific bioinformatics software. The findings of this work identify challenges and opportunities of bioinformatics research and development efforts for addressing life science problems that could prove beneficial to policymakers concerned with science who are looking for tools and strategies to enhance the quality of software development. The following section presents the search strategy for extracting relevant articles in the systematic review. The third section discusses the state and possibilities of key software development challenges identified by the literature review, which include requirements gathering, documentation, testing, and integration. The fourth section presents practical conclusions and discussion based on the findings of the work for the purpose of developing higher-quality bioinformatics software.

SEARCH STRATEGY

A systematic review of literature related to the aim of this work was undertaken in conjunction with a review of the challenges faced by bioinformatics engineers. The results of this effort indicate many of the presupposed solutions to the fundamental concerns. The process recommended by Briner & Denyer (2012) and some attributes of the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) statement were implemented for providing a reproducible, scientific, and transparent literature review of software engineering and bioinformatics cross-education. The entire methodological approach is encompassed in the following steps:

- Identification of need, preparation of a proposal, and development of a protocol for review
- Identification of research, selection of studies, assessment of quality, and acquisition/extraction/synthetization of data
- Reporting of the results for the review

A systematic literature search was conducted for addressing the primary research question during January of 2019 of literature ranging in publication from 1990–2019. The findings, which served as a foundation, were augmented with research of additional developments published through July of 2021. The primary scientific database used was
Scopus in which the terms “bioinformatics”, “engineering”, and “education” were searched across all available articles titles. The relevant articles selected for reference were additionally searched based on the snowball effect. Electronic searches were performed for grey literature, which includes unpublished research commissioned by private institutions, public institutions, and government entities. The first 300 article hits from Google were considered of as a sample of published grey literature after excluding unpublished reports, abstracts, review papers, and conference reports. Complete articles were retrieved and evaluated for relevance when abstracts were not available (Fig. 1). All reports and articles fulfilling the inclusion criteria were documented for bibliographical purposes with the data being categorized into emerging software challenges based upon independently-conducted content analyses.

**CHALLENGES AND THEIR OPPORTUNITIES**

From the literature search, four central areas of challenge in software development arose: requirements, documentation, testing, and integration. These challenges focus around reproducibility, which is critical for scientific bioinformatics software as well as scientific
software in general, for that matter. Significant weakness in any of the four identified areas can lead to unmaintainable and potentially unusable software. Each area has had advancements and developments that have led to actual or potential quality and longevity improvements in scientific bioinformatics software. The adoption of certain processes and systems for the development and management of scientific bioinformatics software has demonstrated benefits for multiple of the areas.

The challenge of requirements

According to Segal (2008), scientists usually lie with requirements because they are either unaware of their meaning or cannot properly define them. This is directly at odds with the practice in software engineering of gathering requirements to determine the intended purpose of the software to be developed. To be more complete, the definition process includes surveying the data, studying the current system, identifying business requirements, researching alternatives, and proposing a system (Bourgeois, 2014). Accurate definition of requirements ensures that the system to be developed will be unambiguous, testable, traceable, and measurable (Hay, 2003). Making the situation yet more challenging, in the context of bioinformatics, software requirements are naturally dynamic (Rother et al., 2012). With life science datasets continuing to yield new information and directions, the recording of novel occurrences requires a functional and flexible system (Hauth & Burger, 2008; Abdurakhmonov, 2016). Moreover, a study by Segal & Morris (2008) indicated that software requirements for scientific functions are different from those for business functions. Their study described a model of software developed by bioinformatics engineers wherein the bioinformatics engineers did not appreciate the process of gathering requirements for software engineers because they already knew the requirements of the scientific problem. The authors came to the conclusion that the bioinformatics engineers were often untruthful in discussions about their needs because they lacked awareness or clarity of definition of the requirements.

In a twist, a different study by Rother et al. (2012) showed that bioinformatics engineers were often asked only for functional requirements, preventing associated software engineers from gathering adequate information about the system itself. This seems a bit ironic given that two other studies found that bioinformatics engineers working on projects of various sizes assigned software requirements the lowest rank in terms of project importance (Hannay et al., 2009; Garousi et al., 2019). Perhaps what is missing here is establishment of effective process more than rigorous enforcement of requirements.

Due to bioinformatics being an investigative discipline, the gathering of software requirements might benefit from the Agile approach, which can be summarized as “principles that emphasize building working software people can get their hands on quickly rather than spending a lot of time writing specifications up front” (Duka, 2013). This application of Agile is supported by Kane et al. (2006) who stated that the Agile method is balanced effectively due to the iterative and exploratory nature of the scientific inquiry. Similarly, Verma et al. (2013) proposed an approach to resolve the challenge of gathering bioinformatics software requirements based on the application of four methodologies: UP, domain engineering, SSADM, and Agile. O’Connor et al. (2008)
recommended the sharing of customized analyses through the Generic Model Organism Database (GMODWeb), an integrated workflow system. A few examples of software packages that have been developed with best practices in mind include Tophat (Trapnell et al., 2012), BioPerl project (Leprevost et al., 2014), and Bowtie (Kishchuk et al., 2018). At a basic level, the usage of requirements tracking platforms (e.g., Jira: https://www.atlassian.com/software/jira) and development platforms (e.g., GitHub: https://github.com/) can help with definition of requirements, effective communications, and successful collaboration of bioinformatics engineers towards the development of high-quality software.

The challenge of documentation

Proper documentation is often essential for maintaining custom software, especially software encompassing the integral complexities of bioinformatics research. Unfortunately, bioinformatics engineers often do not focus on documentation (Karimzadeh & Hoffman, 2018). Without documentation, it can be difficult if not impossible to understand the impetus for certain software design decisions. Such documentation can often take the form of a technical manual or a user manual. A technical manual describes the underlying functionality of the software system. A user manual, as its name implies, helps end-users of a software system properly leverage the software (Chimalakonda & Venigalla, 2020; Venigalla & Chimalakonda, 2021). Additionally, a user manual can provide some indication of the purpose or input-output requirements of the software.

When it comes to bioinformatics software development, the documentation process is not only necessary for the increasing number of end-user developers (Letondal & Mackay, 2004) but also the growing number of research projects analyzing and interpreting massive life science datasets which require the development of bioinformatics software (Brandies & Hogg, 2021). When Umarji et al. (2009) questioned a group of bioinformatics engineers, 90% of them indicated that documentation was valuable. The work of Taschuk & Wilson (2017) corroborated the need for documentation. Katerbow et al. (2018) also highly recommended documentation when developing scientific software. Hoda, Salleh & Grundy (2018) described the importance of knowledge in documentation and how it aided software development when using the Agile method. This conclusion chains into the work of Kane et al. (2006) who found that the Agile method can be applied across different bioinformatics tools utilizing extensive quantities of data through the creation of data types and formats to create a functional workflow. Further, documentation helped to embed domain knowledge in source code and improve domain-specific language to promote word sense disambiguation in the software (Ecale Zhou et al., 2019). Documentation can also provide both developers and end-users with supplementary information such as benchmarking results (Ison et al., 2019). There have been other efforts made in improving documentation for software developed for scientific bioinformatics.

According to Ison et al. (2016), improved documentation can likely assist in sustaining stable sharing mechanisms for metadata, particularly for institutional collections (i.e.,
Institute of France Bioinformatics (IFB)) and specialized registries (i.e., BioContainers), as well as improve community standards through OpenAPI and many biological databases (i.e., Web APIs).

To aid in the creation and formatting of documentation to improve its effectiveness, templates can be constructed for each type of documentation (e.g., user guides and system requirements) to that will give guidance to bioinformatics engineers (Karimzadeh & Hoffman, 2018). Furthermore, the use of the Agile methodology for effective documentation and software functionality development helps meet the information needs of the user while simultaneously allowing the application of different tools, such as automated workflow composition (Harris et al., 2019). Automated workflow, which refers to process design, execution, and automation based on workflow rules, can make the curation of bioinformatics software easier by employing different curation tools to derive information from standardized environments (e.g., Galaxy) or other code repositories (e.g., GitHub) to help identify and correct inconsistent information. Proper documentation is particularly important for the bioinformatics community and the international infrastructure of life science research when it comes to explaining and evaluating discoveries and building upon the research of others (Orengo et al., 2020).

The challenge of testing

Software testing assesses the functionality of a software application to determine if predetermined requirements have been met and the resulting product performs to expectations (Patton, 2006). Each step of the software life-cycle is testable. Examples of types of software testing include validation testing, function testing, and unit testing. For scientific bioinformatics software, testing has often proven the most ambiguous and challenging task in the development process due to the software frequently being either data-driven or algorithm-driven, which can prove difficult to validate or verify. This is a type of oracle test problem where given a particular set of inputs for a system, there exists a “challenge of distinguishing the corresponding, desired, correct behavior from potentially incorrect behavior” (Barr et al., 2014).

This challenge can prove quite detrimental for bioinformatics engineers lacking software engineering skills who develop ad-hoc software for specific projects/tasks (Georgesen et al., 2019). The resulting ad-hoc used to determine software performance at an early stage, especially without proper documentation, planning, or processes, yields results that tend to be unstructured and inconsistent, frequently relying on the knowledge and memory of the expert doing the testing (Chhabra, 2012). When developing bioinformatics software, developers must have an in-depth understanding of the knowledge domain, whereas in other areas of software development (e.g., ERP systems), developers often do not need such extensive knowledge (Amershi et al., 2019).

Furthermore, the interdisciplinary nature of bioinformatics requires its professionals to possess a wide range of abilities, resulting in bioinformatics engineers being more likely to be end-users (Allen & Mehler, 2019). To address testing issues (Soergel, 2014) argued that multiple sets of eyes are required in order to validate bioinformatics software and any research results that it may deliver. Hardware issues present a somewhat unique
challenge for testing bioinformatics software. Limited hardware infrastructure can greatly restrict solution options for enormous life science datasets, affecting scalability, latency, performance of data infrastructures, and processing the data itself (Khan et al., 2014).

There have been some efforts to address testing of scientific bioinformatics software. Chen et al. (2009) proposed a metamorphic testing technique that tests properties from software outputs and end-users in parallel rather than using the traditional input-output testing approach. Kamali et al. (2015) surveyed current efforts in bioinformatics software testing and concluded that more research is needed. Lundgren & Kanewala (2016) experimented with subtle faults in testing bioinformatics software by applying pseudo-oracle and metamorphic tests on a genome alignment tool (BBMap). They found that metamorphic tests were effective in detecting subtle faults. Studies in scalability and validation have been performed and extensively reviewed with solutions such as cloud-based (Troup et al., 2016), divide-and-conquer, and multiple execution being examined for large bioinformatics datasets (Yang, Troup & Ho, 2017). Georgeson et al. (2019) created a command-line tool for developing scientific bioinformatics software that provides integrated unit and integration testing capabilities.

Version control systems can also aid in the testing of bioinformatics software through the use of certain tools, such as regression tests. Many free services exist, such as thefreecountry.com and Github.com, that provide version control solutions with tools that could prove helpful for bioinformatics software testing. Mulder et al. (2018) strongly encouraged bioinformatics engineers to utilize version control systems in order to maintain code quality and availability for the community. This also has the added benefit of exposure of the utilized software management processes to experienced software developers who can provide professional feedback and assistance. The Agile methodology can be applied when managing a project to divide the project into several different phases that involve constant collaboration with stakeholders and continuous enhancement at early stages. Two studies of software development using Agile methodology proved that scientists, in general, were well-satisfied with the resulting software testing (Umarji et al., 2009; Dingsøyr et al., 2018). A testing approach that could prove to be very effective involves scientists exposing all relevant source code during publication processes. This would not only allow for third-party validation, but also bolster testing results for reproducibility. Numerous efforts such as the Bioconductor (Gentleman et al., 2004) and OpenEBench (https://openebench.bsc.es) platforms have been undertaken for this purpose (Capella-Gutierrez et al., 2017).

The challenge of integration
Being able to replicate and reproduce scientific results has progressively grown in significance for most of academic disciplines (Stodden, Borwein & Bailey, 2013; Howison et al., 2015; Miyakawa, 2020). For bioinformatics software, integration plays an important role in reproducibility. Gulledge (2006) concluded that “integration implies that all relevant data for a particular bounded and closed set of business processes are processed in the same software application.” According to Barker & Thornton (2004), integration has been proven to be the most important challenge for bioinformatics software over the
next decade. However, the vast diversity of life science datasets, including genome sequences, literature, and software, can make integration seemingly impossible (Hannay et al., 2009; Lapatas et al., 2015). Problems can be faced when trying to handle the challenges of integrating data of differing qualities while incorporating sources of disseminated information (Miyakawa, 2020). Platform and system peculiarities can pose particular problems for integration due to interoperability concerns between them (Karasavvas, Baldock & Burger, 2004; Koru et al., 2007; Chilana, Palmer & Ko, 2009; Mangul et al., 2019). Differences between libraries from one operating system or version to another can result in varying outcomes when making system calls with software. If bioinformatics software is developed on a UNIX platform, for example, a developer will incorporate some libraries in the code that will enable it to operate on the UNIX platform (Gentleman et al., 2004). Public sharing of developed bioinformatics software, especially porting efforts, can help to address issues with system and platform variations.

(Steinberg, 2003) asked bioinformatics engineers to create an online community in which they could share not only data but also developed software and stressed the value of reusing software by encouraging bioinformatics software be made open-source. Following Steinberg’s research, great standardization efforts have been developed and deployed. Standardized API documentation using OpenAPI (Mulligan, 2008) and BrAPI (Selby et al., 2019), ontologies for annotating bio tools (e.g. EDAM (Ison et al., 2013)), workflow systems including a great number in life sciences ranging from analytic platforms like KNIME (Fillbrunn et al., 2017) to Jupyter Notebooks (Kluyver et al., 2016), command-line workflow tools like Nextflow (Di Tommaso et al., 2017), and dependency systems like SnakeMake (Köster & Rahmann, 2012) have all aided the bioinformatics community. Additionally, containers such as Bioboxes (Belmann et al., 2015) and BioContainers (da Veiga Leprevost et al., 2017), packagers such as Bioconda (Grüning et al., 2018), and distributions (https://wiki.debian.org/DebianScience/Biology) have proven very useful. The Common Workflow Language (http://www.commonwl.org) is an effort primarily driven by the life science community. All of these efforts have helped to tackle the challenge of integration for scientific bioinformatics software.

With integration and reproducibility of bioinformatic software being so closely tied when performing data-driven research, embracing open approaches to managing software development have proven highly beneficial. It enables bioinformatics engineers to build on the work of other works while also allowing others to shape their efforts and help solve their collective problems. This leads to the establishment the software with greater longevity, more effective problem solving, and longer sustainability (Ivie & Thain, 2018).

CONCLUSION

This study explores several areas of software development that pose both challenges and possibilities for the development of scientific bioinformatics software, which include requirements gathering, documentation, testing, and integration. When it comes to bioinformatics engineers mastering such skills, many cultural factors must be considered, including funding patterns that reward new development and preclude reuse as well as
lack of sustainability of third-party codes and reluctant support for their adoption. Bioinformatics engineers are often isolated or alone in research groups. They may not be afforded the time or capacity to incorporate outside software nor prepare their own software for broader usage, possibly as a result of project or PI requirements. To really understand the relationship between bioinformatics and software engineering requires understanding the drivers.

Bioinformatics is still in its infancy compared with the majority of other scientific disciplines. The combination of mathematics, statistics, and computation in a cross-disciplinary fashion in conjunction with numerous fields is a driving factor in adopting bioinformatics practices. The objective is to merge extensive knowledge of both computation and biology into one brain in order to appropriately address the complexities of life science datasets and problems with effective software solutions. This can prove costly and unrealistic in many scenarios. Software engineering practices for producing valid, maintainable, and sustainable solutions, while abstract in their nature, can prove challenging to apply directly to bioinformatics research due to factors inherent within the life sciences and bioinformatics themselves. This basically necessitates that bioinformatics engineers incorporate and adapt software engineering skills and principles into their skillsets rather than attempting to collaborate with software engineers lacking expertise in bioinformatics. However, when aspects of project management of the development of bioinformatics software are considered, it is possible for bioinformatics engineers to collaborate openly with one another and professional software engineers to facilitate and enhance their efforts. These efforts may be the key to producing higher-quality scientific bioinformatics software for the future through the structured development of more robust software and the reduced recreation of the wheel. Bioinformatics engineers can reap the benefits of applying software engineering principles in terms of the efficiency and effectiveness of their works by keeping their collective software development efforts DRY (don’t repeat yourself).

**ADDITIONAL INFORMATION AND DECLARATIONS**

**Funding**
We received support from the Deanship of Science Research (DSR) at King Abdulaziz University, Jeddah, Saudi Arabia. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

**Grant Disclosures**
The following grant information was disclosed by the authors:
King Abdulaziz University, Jeddah, Saudi Arabia.

**Competing Interests**
The author declares that he has no competing interests.
Author Contributions

- Adeeb Noor conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.

Data Availability

The following information was supplied regarding data availability: This is a literature review.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.839#supplemental-information.

REFERENCES

Abdurakhmonov IY. 2016. *Bioinformatics: basics, development, and future*. London: InTech.

Allen C, Mehler DM. 2019. Open science challenges, benefits and tips in early career and beyond. *PLoS Biology* 17(5):e3000246 DOI 10.1371/journal.pbio.3000246.

Althubaiti S, Karwath A, Dallol A, Noor A, Alkhayyat SS, Alwassia R, Mineta K, Gojobori T, Beggs AD, Schofield PN, Gkoutos GV. 2019. Ontology-based prediction of cancer driver genes. *Scientific Reports* 9(1):1–9 DOI 10.1038/s41598-019-53454-1.

Amershi S, Begel A, Bird C, DeLine R, Gall H, Kamar E, Nagappan N, Nushi B, Zimmermann T. 2019. Software engineering for machine learning: a case study. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice. Piscataway: IEEE, 291–300.

Aryal S. 2019. FASTA and BLAST. Available at https://microbenotes.com/fasta-and-blast/.

Assiri A, Noor A. 2020. A computational approach to predict multi-pathway drug–drug interactions: a case study of irinotecan, a colon cancer medication. *Saudi Pharmaceutical Journal* 28(12):1507–1513 DOI 10.1016/j.jsps.2020.09.017.

Attwood TK, Blackford S, Brazas MD, Davies A, Schneider MV. 2019. A global perspective on evolving bioinformatics and data science training needs. *Briefings in Bioinformatics* 20(2):398–404 DOI 10.1093/bib/bbx100.

Barker J, Thornton J. 2004. Software engineering challenges in bioinformatics. In: Proceedings. 26th International Conference on Software Engineering. Piscataway: IEEE, 12–15.

Barr ET, Harman M, McMinn P, Shahbaz M, Yoo S. 2014. The oracle problem in software testing: a survey. *IEEE Transactions on Software Engineering* 41(5):507–525 DOI 10.1109/TSE.2014.2372785.

Baxter SM, Day SW, Fetrow JS, Reisinger SJ. 2006. Scientific software development is not an oxymoron. *PLoS Computational Biology* 2(9):e87 DOI 10.1371/journal.pcbi.0020087.

Bayat A. 2002. Science, medicine, and the future: Bioinformatics. *BMJ: British Medical Journal* 324(7344):1018–1022 DOI 10.1136/bmj.324.7344.1018.

Belmann P, Dröge J, Bremges A, McHardy AC, Sczyrba A, Barton MD. 2015. Bioboxes: standardised containers for interchangeable bioinformatics software. *Gigascience* 4(1): s13742-s13015 DOI 10.1186/s13742-015-0087-0.

Borgman CL, Bourne PE. 2021. Why it takes a village to manage and share data. *ArXiv. Available at http://arxiv.org/abs/2109.01694.*
Bourgeois D. 2014. *Information systems for business and beyond*. Arlington: The Saylor Foundation.

Brandies PA, Hogg CJ. 2021. Ten simple rules for getting started with command-line bioinformatics. *PLoS Computational Biology*. 17(2):e1008645 DOI 10.1371/journal.pcbi.1008645.

Briner RB, Denyer D. 2012. Systematic review and evidence synthesis as a practice and scholarship tool. In: Rousseau DM, ed. *Handbook of Evidence-Based Management: Companies, Classrooms and Research*. Oxford: Oxford University Press, 112–129.

Capella-Gutierrez S, de la Iglesia D, Haas J, Lourenco A, Fernández JM, Repchevsky D, Dessimoz C, Schwede T, Notredame C, Gelpi JL, Valencia A. 2017. Lessons learned: recommendations for establishing critical periodic scientific benchmarking. *BioRxiv* 49(1):181677 DOI 10.1101/181677.

Cashman M, Cohen MB, Ranjan P, Cottingham RW. 2018. Navigating the maze: the impact of configurability in bioinformatics software. In: *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 757–767.

Chen TY, Ho JW, Liu H, Xie X. 2009. An innovative approach for testing bioinformatics programs using metamorphic testing. *BMC Bioinformatics* 10(1):1–12 DOI 10.1186/1471-2105-10-24.

Chhabra MN. 2012. Introduction to adhoc testing. *International Journal of Scientific and Technology Research* 1(7):66–67.

Chilana PK, Palmer CL, Ko AJ. 2009. Comparing bioinformatics software development by computer scientists and biologists: an exploratory study. In: 2009 *ICSE Workshop on Software Engineering for Computational Science and Engineering*. Piscataway: IEEE, 72–79.

Chimalakonda S, Venigalla ASM. 2020. Software documentation and augmented reality: love or arranged marriage? In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. New York: ACM, 1529–1532.

da Veiga Leprevost F, Grünig Börn A, Alves Aflitos S, Röst HL, Uszkoreit J, Barsnes H, Vaudel M, Moreno P, Gatto L, Weber J, Bai M, Jimenez RC, Sachsenberg T, Pfeuffer J, Vera Alvarez R, Griss J, Nesvizhskii AI, Perez-Riverol Y, Valencia A. 2017. BioContainers: an open-source and community-driven framework for software standardization. *Bioinformatics* 33(16):2580–2582 DOI 10.1093/bioinformatics/btx192.

Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. 2017. Nextflow enables reproducible computational workflows. *Nature Biotechnology* 35(4):316–319 DOI 10.1038/nbt.3820.

Dingsøyr T, Moe NB, Fægri TE, Seim EA. 2018. Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. *Empirical Software Engineering* 23(1):490–520 DOI 10.1007/s10664-017-9524-2.

Duka D. 2013. Adoption of agile methodology in software development. In: 2013 *36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Piscataway: IEEE, 426–430.

Ecalle Zhou CL, Malatti S, Kimbrel J, Philipson C, McNair K, Hamilton T, Edwards R, Souza B, Souza B. 2019. multiPhATE: bioinformatics pipeline for functional annotation of phage isolates. *Bioinformatics* 35(21):4402–4404 DOI 10.1093/bioinformatics/btz258.

Fillbrunn A, Dietz C, Pfeuffer J, Rahn R, Landrum GA, Berthold MR. 2017. KNIME for reproducible cross-domain analysis of life science data. *Journal of Biotechnology* 261(1):149–156 DOI 10.1016/j.jbiotec.2017.07.028.
Garousi V, Tarhan A, Pfahl D, Coşkunçay A, Demirörs O. 2019. Correlation of critical success factors with success of software projects: an empirical investigation. *Software Quality Journal* 27(1):429–493 DOI 10.1007/s11219-018-9419-5.

Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JYH, Zhang J. 2004. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology* 5(10):R80 DOI 10.1186/gb-2004-5-10-r80.

Georgeson P, Syme A, Sloggett C, Chung J, Dashnow H, Milton M, Lonsdale A, Powell D, Seemann T, Pope B. 2019. Bionitio: demonstrating and facilitating best practices for bioinformatics command-line software. *GigaScience* 8(9):giz109 DOI 10.1093/gigascience/giz109.

Grüning B, Dale R, Sjödin A, Chapman BA, Rowe J, Tomkins-Tinch CH, Valieris R, Köster J. 2018. Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nature Methods* 15(7):475–476 DOI 10.1038/s41592-018-0046-7.

Gulledge T. 2006. What is integration? *Industrial Management & Data Systems* 106(1):5–20 DOI 10.1108/02635570610640979.

Gupta A, Gangotia D, Mani I. 2021. Bioinformatics tools and software. In: *Advances in Bioinformatics*. Singapore: Springer, 15–35.

Hannay JE, MacLeod C, Singer J, Langtangen HP, Pfahl D, Wilson G. 2009. How do scientists develop and use scientific software? In: 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering. Piscataway: IEEE, 1–8.

Harris NL, Cock PJ, Fields CJ, Tzovaras BG, Heuer M, Hokamp K, Munoz-Torres M, Peltzer A, Rieck B, Wiencko H, Yehudi Y. 2019. Bosc 2019, the 20th annual bioinformatics open source conference. *F1000Research* 8:2132 DOI 10.12688/f1000research.21568.1.

Hauth AM, Burger G. 2008. Methodology for constructing problem definitions in bioinformatics. *Bioinformatics and Biology Insights* 2:BBL.S706–BBL.S706 DOI 10.4137/BBL.S706.

Hay DC. 2003. *Requirements analysis: from business views to architecture*. Hoboken: Prentice Hall Professional.

Hoda R, Salleh N, Grundy J. 2018. The rise and evolution of agile software development. *IEEE Software* 35(5):58–63 DOI 10.1109/MS.2018.29011318.

Hoffman MA. 2007. The genome-enabled electronic medical record. *Journal of Biomedical Informatics* 40(1):44–46 DOI 10.1016/j.jbi.2006.02.010.

Howison J, Deelman E, McLennan MJ, Ferreira da Silva R, Herbsleb JD. 2015. Understanding the scientific software ecosystem and its impact: current and future measures. *Research Evaluation* 24(4):454–470 DOI 10.1093/reseval/rvv014.

Ison J, Ienasescu H, Chmura P, Rydza E, Menager H, Kalaš M, Brunak S. 2019. The bio. tools registry of software tools and data resources for the life sciences. *Genome Biology* 20(1):1–4 DOI 10.1186/s13059-019-1772-6.

Ison J, Kalaš M, Jonassen I, Bolser D, Uludag M, McWilliam H, Rice P. 2013. EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics* 29(10):1325–1332 DOI 10.1093/bioinformatics/btt113.

Ison J, Rapacki K, Ménager H, Kalaš M, Rydza E, Chmura P, Anthon C, Beard N, Berka K, Bolser D, Booth T, Bretaudeau A, Brezovský J, Casadio R, Cesareni G, Coppens F, Cornell M, Cuccuru G, Davidsen K, Vedova GD, Dogan T, Doppelt-Azeroual O, Emery L, Gasteiger E, Gatter T, Goldberg T, Grosjean M, Grünning Bön, Helmer-Citterich M, Ienasescu H, Ioannidis V, Jespersen MC, Jimenez R, Juty N, Juvan P, Koch M, Laibe C,
Li J-W, Licata L, Mareuil F, Mičetić I, Friborg RM, Moretti S, Morris C, Möller S, Nenadic A, Peterson H, Profeti G, Rice P, Romano P, Roncaglia P, Saidi R, Schafferhans A, Schwämmle V, Smith C, Sperotto MM, Stockinger H, Vařková RS, Tosatto SCE, de la Torre V, Uva P, Via A, Yachdav G, Zambelli F, Vriend G, Rost B, Parkinson H, Longreen P, Brunak S. 2016. Tools and data services registry: a community effort to document bioinformatics resources. *Nucleic Acids Research* 44(D1):D38–D47 DOI 10.1093/nar/gkv1116.

Ivie P, Thain D. 2018. Reproducibility in scientific computing. *ACM Computing Surveys* 51(3):1–36 DOI 10.1145/3186266.

Joppa LN, McInerny G, Harper R, Salido L, Takeda K, O’Hara K, Gavaghan D, Emmott S. 2013. Troubling trends in scientific software use. *Science* 340(6134):814–815 DOI 10.1126/science.1231535.

Kamali AH, Giannoulatou E, Chen TY, Charleston MA, McEwan AL, Ho JW. 2015. How to test bioinformatics software? *Biophysical Reviews* 7(3):343–352 DOI 10.1007/s12551-015-0177-3.

Kane DW, Hohman MM, Cerami EG, McCormick MW, Kuhlmann KF, Byrd JA. 2006. Agile methods in biomedical software development: a multi-site experience report. *BMC Bioinformatics* 7(1):273 DOI 10.1186/1471-2105-7-273.

Karasavvas KA, Baldock R, Burger A. 2004. Bioinformatics integration and agent technology. *Journal of Biomedical Informatics* 37(3):205–219 DOI 10.1016/j.jbi.2004.04.003.

Karimzadeh M, Hoffman MM. 2018. Top considerations for creating bioinformatics software documentation. *Briefings in Bioinformatics* 19(4):693–699 DOI 10.1093/bib/bbw134.

Katerbow M, Feulner G, Bornschein M, Brembs B, Erben-Russ M, Förstner K, Zielke D. 2018. Recommendations on the development, use and provision of research software. *Zenodo* DOI 10.5281/zenodo.1172988.

Khan N, Yaqoob I, Hashem IAT, Inayat Z, Mahmoud Ali WK, Alam M, Gani A. 2014. Big data: survey, technologies, opportunities, and challenges. *The Scientific World Journal* 2014(6077):1–18 DOI 10.1155/2014/712826.

Kishchuk BE, Creed IF, Laurent KL, Nebel S, Kreutzweiser D, Venier L, Webster K. 2018. Assessing the ecological sustainability of a forest management system using the ISO Bowtie risk management assessment tool. *The Forestry Chronicle* 94(1):25–34 DOI 10.5558/tfc2018-005.

Kluyver T, Ragan-Kelley B, Pérez F, Granger BE, Bussonnier M, Frederic J, Kelley K, Hamrick JB, Grout J, Corlay S, Ivanov P. 2016. Jupyter Notebooks—a publishing format for reproducible computational workflows. In: 20th International Conference on Electronic Publishing. 2016:87–90.

Koch C, Jones A. 2016. Big science, team science, and open science for neuroscience. *Neuron* 92(3):612–616 DOI 10.1016/j.neuron.2016.10.019.

Koru G, El Emam K, Neisa A, Umarji M. 2007. A survey of quality assurance practices in biomedical open-source software projects. *Journal of Medical Internet Research* 9:e8 DOI 10.2196/jmir.9.2.e8.

Köster J, Rahmann S. 2012. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics* 28(19):2520–2522 DOI 10.1093/bioinformatics/bts480.

Lapatas V, Stefanidakis M, Jimenez RC, Via A, Schneider MV. 2015. Data integration in biological research: an overview. *Journal of Biological Research-Thessaloniki* 22(1):9 DOI 10.1186/s40709-015-0032-5.

Lawlor B, Sleator RD. 2021. The roles of code in biology. *Science Progress* 104(2):00368504211010570 DOI 10.1177/00368504211010570.
Lawlor B, Walsh P. 2015. Engineering bioinformatics: building reliability, performance and productivity into bioinformatics software. *Bioengineered* **6**(4):193–203 DOI 10.1080/21655979.2015.1050162.

Leimeister CA, Boden M, Horwege S, Lindner S, Morgenstern B. 2014. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics* **30**(14):1991–1999 DOI 10.1093/bioinformatics/btu177.

Leprevost FDV, Barbosa VC, Francisco EL, Perez-Riverol Y, Carvalho PC. 2014. On best practices in the development of bioinformatics software. *Frontiers in Genetics* **5**:199 DOI 10.3389/fgene.2014.00199.

Letondal C, Mackay WE. 2004. Participatory programming and the scope of mutual responsibility: balancing scientific, design and software commitment. In: *Proceedings of the Eighth Conference on Participatory Design: Artful Integration: Interweaving Media, Materials and Practices*. 1:31–41.

List M, Ebert P, Albrecht F. 2017. Ten simple rules for developing usable software in computational biology. *PLoS Computational Biology* **13**(1):e1005265 DOI 10.1371/journal.pcbi.1005265.

Lundgren A, Kanewala U. 2016. Experiences of testing bioinformatics programs for detecting subtle faults. In: *Proceedings of the International Workshop on Software Engineering for Science*. 16–22.

Mangul S, Martin LS, Eskin E, Blekhman R. 2019. Improving the usability and archival stability of bioinformatics software. *Genome Biology* **20**(1):2686 DOI 10.1186/s13059-019-1649-8.

Merali Z. 2010. Error: why scientific programming does not compute. *Nature* **467**(7317):775–777 DOI 10.1038/467775a.

Miyakawa T. 2020. No raw data, no science: another possible source of the reproducibility crisis. *Molecular Brain* **13**(1):712 DOI 10.1186/s13041-020-0552-2.

Mulder N, Schwartz R, Brazas MD, Brooksbank C, Gaeta B, Morgan SL, Pauley MA, Rosenwald A, Rustici G, Sierk M, Warnow T. 2018. The development and application of bioinformatics core competencies to improve bioinformatics training and education. *PLoS Computational Biology* **14**(2):e1005772 DOI 10.1371/journal.pcbi.1005772.

Mulligan C. 2008. Open API standardisation for the NGN platform. In: *2008 First ITU-T Kaleidoscope Academic Conference-Innovations in NGN: Future Network and Services*. Piscataway: IEEE, 25–32.

Noor A. 2019a. The utilization of e-health in the Kingdom of Saudi Arabia. *International Journal of Research in Engineering and Technology* **6**(9):11.

Noor A. 2019b. Discovering gaps in Saudi education for digital health transformation. *International Journal of Advanced Computer Science and Applications* **10**:105–119 DOI 10.14569/issn.2156-5570.

Noor A, Assiri A. 2021. A novel computational drug repurposing approach for Systemic Lupus Erythematosus (SLE) treatment using Semantic Web technologies. *Saudi Journal of Biological Sciences* **28**(7):3886–3892 DOI 10.1016/j.sjbs.2021.03.068.

Noor A, Assiri A, Ayvaz S, Clark C, Dumontier M. 2017. Drug–drug interaction discovery and demystification using Semantic Web technologies. *Journal of the American Medical Informatics Association* **24**(3):556–564 DOI 10.1093/jamia/ocw128.

Orengo C, Velankar S, Wodak S, Zoete V, Bonvin AM, Elofsson A, Feenstra KA, Gerloff DL, Hamelryck T, Hancock JM, Helmer-Citterich M, Hospital A, Orozco M, Perrakis A, Rarey M, Soares C, Sussman JL, Thornton JM, Tuffery P, Tusnady G, Wierenga R, Salminen T, Schneider B. 2020. A community proposal to integrate structural bioinformatics
activities in ELIXIR (3D-Bioinfo Community). *F1000Research* 9(ELIXIR):278 DOI 10.12688/f1000research.20559.1.

Oulas A, Minadakis G, Zachariou M, Sokratous K, Bourdakou MM, Spyrou GM. 2019. Systems bioinformatics: increasing precision of computational diagnostics and therapeutics through network-based approaches. *Briefings in Bioinformatics* 20(3):806–824 DOI 10.1093/bib/bbx151.

O’Connor BD, Day A, Cain S, Arnaiz O, Sperling L, Stein LD. 2008. GMODWeb: a web framework for the generic model organism database. *Genome Biology* 9(6):1–11 DOI 10.1186/gb-2008-9-6-r102.

Patton R. 2006. *Software testing*. Noida: Pearson Education India.

Ranganathan S. 2005. Bioinformatics education—perspectives and challenges. *PLoS Computational Biology* 1(6):e52 DOI 10.1371/journal.pcbi.0010052.

Rother K, Potrzebowski W, Puton T, Rother M, Wywial E, Bujnicki JM. 2012. A toolbox for developing bioinformatics software. *Briefings in Bioinformatics* 13(2):244–257 DOI 10.1093/bib/bbr035.

Sanders R, Kelly D. 2008. Dealing with risk in scientific software development. *IEEE Software* 25(4):21–28 DOI 10.1109/MS.2008.84.

Segal J. 2007. Some problems of professional end user developers. In: *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007)*. Piscataway: IEEE, 111–118.

Segal J. 2008. Scientists and software engineers: a tale of two cultures. In: *Proceedings of the Psychology of Programming Interest Group (PPIG 08)*. 44–51.

Segal J, Morris C. 2008. Developing scientific software. *IEEE Software* 25(4):18–20 DOI 10.1109/MS.2008.85.

Selby P, Abbeloos R, Backlund JE, Basterrechea Salido M, Bauchet G, Benites-Alfaro OE, Birkett C, Calaminos VC, Carceller P, Cornut G, Vasques Costa B. 2019. BrAPI—an application programming interface for plant breeding applications. *Bioinformatics* 35(20):4147–4155 DOI 10.1093/bioinformatics/btz190.

Soergel DA. 2014. Rampant software errors may undermine scientific results. *F1000Research* 3:303 DOI 10.12688/f1000research.

Steinberg DH. 2003. Stein gives bioinformatics ten years to live. Available at [https://courses.cs.duke.edu/cps006g/fall04/papers/bioinformaticsdead.pdf](https://courses.cs.duke.edu/cps006g/fall04/papers/bioinformaticsdead.pdf).

Stodden V, Borwein J, Bailey DH. 2013. Setting the default to reproducible: computational science research. *SIAM News* 46:4–6.

Taschuk M, Wilson G. 2017. Ten simple rules for making research software more robust. *PLOS Computational Biology* 13(4):e1005412 DOI 10.1371/journal.pcbi.1005412.

Tenenbaum JD, Shah NH, Altman RB. 2021. Translational bioinformatics. In: *Biomedical Informatics*. Cham: Springer, 867–911.

Trapnell C, Roberts A, Goff I, Pertea G, Kim D, Kelley DR, Pimentel H, Salzberg SL, Rinn JL, Pachter L. 2012. Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nature Protocols* 7(3):562–578 DOI 10.1038/nprot.2012.016.

Troup M, Yang A, Kamali AH, Giannoulatou E, Chen TY, Ho JW. 2016. A cloud-based framework for applying metamorphic testing to a bioinformatics pipeline. In: *Proceedings of the 1st International Workshop on Metamorphic Testing*. 33–36.

Umarji M, Seaman C, Koru AG, Liu H. 2009. Software engineering education for bioinformatics. In: 2009 22nd Conference on Software Engineering Education and Training. Piscataway: IEEE, 216–223.
Venigalla ASM, Chimalakonda S. 2021. Understanding emotions of developer community towards software documentation. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society. Piscataway: IEEE, 87–91.

Verma D, Gesell J, Siy H, Zand M. 2013. Lack of software engineering practices in the development of bioinformatics software. ICCEGI 2013:57–62.

Weston P. 2004. Bioinformatics software engineering: delivering effective applications. Hoboken: John Wiley & Sons.

Wilson G. 2014. Software carpentry: lessons learned. F1000Research 3:62 DOI 10.12688/f1000research.3-62.v1.

Wilson G, Aruliah DA, Brown CT, Hong NPC, Davis M, Guy RT, Haddock SH, Huff KD, Mitchell IM, Plumbley MD, Waugh B. 2014. Best practices for scientific computing. PLoS Biology 12(1):e1001745 DOI 10.1371/journal.pbio.1001745.

Yang A, Troup M, Ho JW. 2017. Scalability and validation of big data bioinformatics software. Computational and Structural Biotechnology Journal 15(4):379–386 DOI 10.1016/j.csbj.2017.07.002.