Paper ET07

The changes in the job of statistical programmer in the pharmaceutical industry

James Kim, Pfizer Inc, Cary, USA

ABSTRACT

This paper discusses the evolving roles of statistical programmers in the pharmaceutical industry compared to traditional computer science programmers. The distinction was clear when the main tool of statistical programmers has been proprietary software for the last 30 years. But as the adoption of open-source tools like R is gaining momentum recently, the distinction between them has become unclear and created confusion on what their main responsibilities should be in their day-to-day activities. While there is no correct answer and their roles can certainly be overlapped to a certain extent, understanding the differences will help to plan programming capabilities efficiently, especially for smaller-sized companies with limited resources.

INTRODUCTION

This is a broad topic, and the scope of discussion can vary depending on the level of detail. Also, there are subtopics that can also be discussed as derivatives, such as validation, framework, and regulatory requirements. These topics will be briefly covered, and I will refer to future works where further discussion is proper. This paper will start by curating terms which will be often referred and abbreviated throughout the discussion. Then misconceptions and clarifications will be presented, followed by the current approaches and finally recommend some latest ideas to improve them.

DEFINITIONS

Open-Source or Open-Source Software – The term “Open” has a special meaning when it comes to knowledge. For example, the Open Science movement has the goal of making scientific research, data, code, and publications freely accessible to everyone. Open-Source or Open-Source Software specifically refers to the intellectual property associated with programs or software to be used and changed freely. These terms will be used synonymously throughout the paper. The opposite side of OSS is proprietary or called Closed-Source Software (CSS).

Computer Science Programmers (CSP) – these are traditional programmers who work in software or technology sector. Their main choices of programming are C, C++, Java and Python. They are referred with many different names: *Programmers, Developers, Software Engineers, Coders*, etc. Their platforms are wide such as Operating Systems (e.g., Microsoft Windows ®, Apple MacOS ®, Google Android ®, Linux), Applications (e.g., Microsoft products, Adobe products), Software as Service (SaaS) (e.g., Amazon AWS ® and Microsoft Azure ®). In this paper, they will be collectively referred to as “**CSP**.” The key attribute defining CSP is their use of algorithms or logics to build specific applications to process workflows with optional inputs and outputs to conduct the given tasks.

Data Science Programmers (DSP) – these are a type of professional who use software to analyze data, create reports, and support research projects. And these programmers are the ones who mostly are found in the programming area of pharmaceutical or life science companies. They are ensuring that data is processed and presented accurately, and that it can communicate, including regulatory requirements. They primarily work on data or databases, and use mathematical and statistical methods to collect, organize, interpret, and summarize numerical data. They share common names such as *Clinical Programmers, Statistical Programmers* or more recently, *Data Scientists*. Their choice of popular tools is SQL, SAS, MATLAB, R or Python and their type of programming commonly is called “Scripting,” to differentiate from CSP. In this paper, they will be collectively referred to as “**DSP**.” The key attribute defining DSP is their use of ***data***.

Pharma – In this paper, this term refers to diverse types of life science, medical, and biotechnology companies such as pharmaceutical, medical devices, contract research organization (CRO) or laboratory organizations.

Regulatory Authority (RA) – The FDA is the main RA organization in the United States, but the term also includes other organizations like PMDA (Japan) and MHRA (U.K.).

APPLICATION VS DATA PROGRAMMING

So, what is the one feature that separates CSP and DSP? The key difference lies in the content being programmed. CSP builds blocks of codes that are consisted of logics and algorithms inside an application or framework on platforms that typically produce output (but not necessarily data), whereas DSP focuses on understanding of

underlying data which leads to interpretations and conclusions that usually produce final data with recognizable patterns, visualizations, or logics.

It is interesting to note that there has been an interesting trend of programming job migration recently – in the early days of computing such as in the 70s through early 2000s, when we refer to “programming”, we usually thought of this activity in relation to CSP, except for some explicit areas such as Database programming with high-level language such as SQL that were inherently associated with DSP. CSP’s “Application Driven Programming” focuses on building software, while DSP’s “Data Driven Programming” prioritizes using external data to guide the application’s behavior and decision-making. Now DSP is becoming the dominant power of programming trend – partially thanks to the advancement in technology and manufacturing of cheap storage spaces. With the growth of large datasets, and the Artificial Intelligence (AI) boom that capitalized on, you can only imagine that the trend will continue in the foreseeable future. However, it is important to note that these two types of programming have unique requirements and roles in Pharma and differentiating them will help to prioritize and assemble the right teams for your organization.

RECAP OF OPEN-SOURCE SOFTWARE (OSS)

Open-source software (OSS) refers to software whose source code is made freely available for modification and redistribution. This model promotes universal access via an open-source or free license to a product's design or blueprint, and universal redistribution of that design or blueprint. OSS is characterized by a decentralized software development model that encourages open collaboration. The transparency of OSS allows users to inspect, understand, and verify its functionality, which builds trust and can enhance the security of the software. Today, "open source" appoints a broader set of values, including open exchange, collaborative participation, rapid prototyping, transparency, meritocracy, and community-oriented development. Although there are many OSS projects and tools used in Pharma, R has been a solid choice of programming and therefore, it is used synonymously with OSS in this paper.

STRENGTHS

1. One of the primary advantages of OSS is its cost-effectiveness. Since OSS is freely available, organizations can significantly reduce software acquisition costs, making it an appealing choice for businesses and individuals alike.
2. A diverse group of developers worldwide contribute to the improvement and enhancement of the software, often resulting in faster development cycles and innovative solutions.
3. Another key benefit is flexibility and customization. OSS provides users with the freedom to change the source code according to their specific needs, allowing organizations to tailor the software to meet their unique requirements.
4. The transparency of OSS is also a significant advantage. Users have access to the source code, enabling them to inspect, understand, and verify its functionality, which builds trust and can enhance the security of the software.

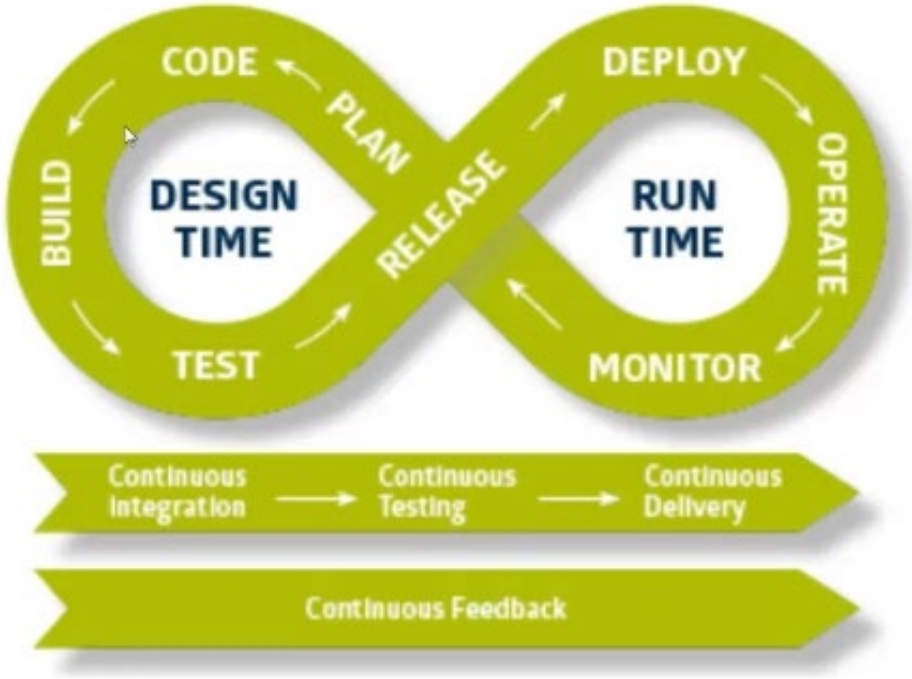
CHALLENGES

1. Lack dedicated customer support – users often rely on community forums and documentation for aid, which may not be as responsive or tailored to specific organizational needs as commercial support.
2. Potential security vulnerabilities – while the transparency of OSS allows for rigorous scrutiny of code, it also means that any vulnerabilities are visible to everyone, including malicious actors.
3. In relation to #1, it may suffer from a lack of consistent updates and maintenance, as they often rely on voluntary contributions from the community.
4. In relation to #1 and #3, the cost-effectiveness advantage mentioned previously diminishes since the integration and management of production environments using these OSS tools is left to the user or organization, giving rise to commercial solutions to solve those problems.
5. There is the risk of intellectual property (IP) issues, as the open nature of OSS can sometimes lead to disputes over code ownership and licensing.

PROGRAMMING PLATFORMS IN PHARMA

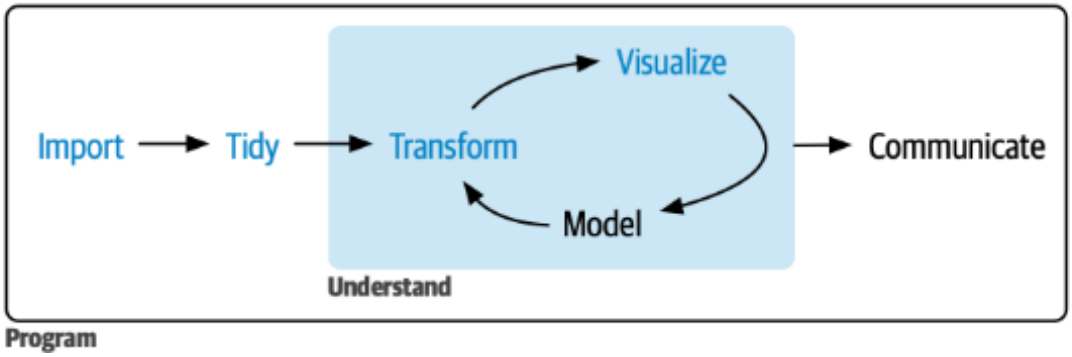
WORKFLOW FOR CSP (APPLICATION DEVELOPMENT)

Below is a typical workflow diagram of the application development cycle (DevOps) with CI/CD (Continuous Integration and Continuous Delivery).

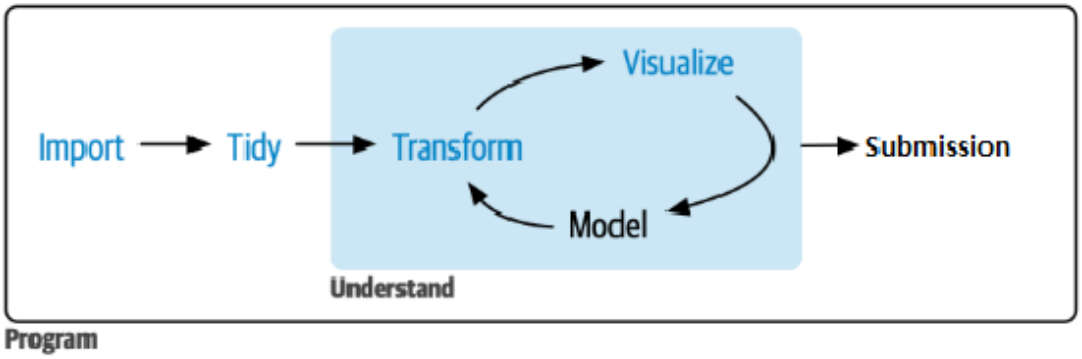


WORKFLOW FOR DSP (DATA SCIENCE PROGRAMMING)

In “R for Data Science (https://r4ds.hadley.nz/),” Hadley Wickham gave the overview of data science workflow with the following diagram.



For Pharma, we have a special requirement – all works, including any data science types, must be done with the submission step in mind for the regulatory approval process. This also means you are bound to work with the CFR21 Part 11 rule which can have different meanings depending on your type of works and environments. So, the last step in the above diagram can be replaced like this.

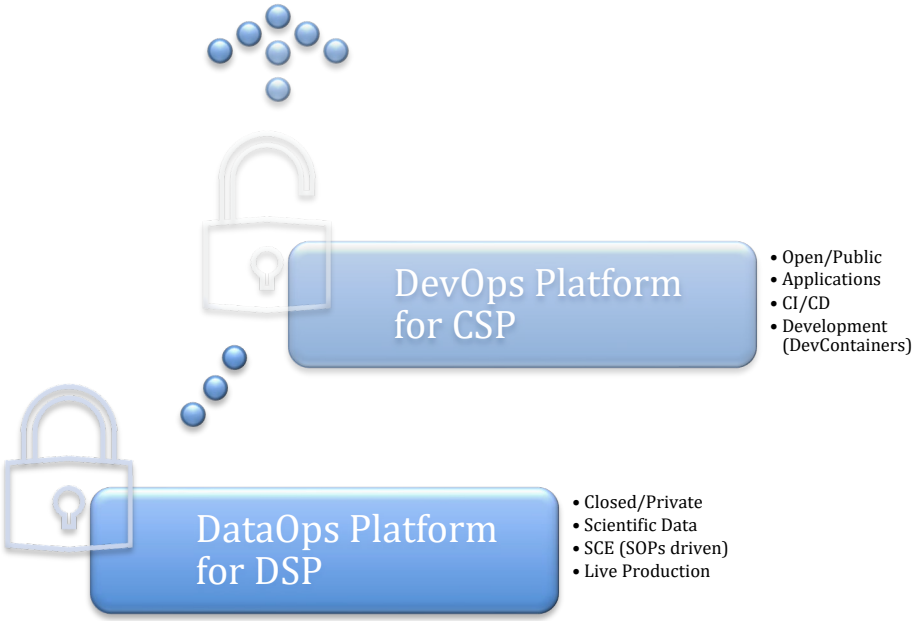


And this last step is the pinnacle of all other works – there are not many points to generate data or results if they cannot be used for submission and this has overall implications on how we generate and process data in the upstream processes.

PLATFORM REQUIREMENTS

Pharma companies must support a controlled environment to meet the RA requirements, and this should remain as the DSP's main place of business. This is usually satisfied by having a system referred as Statistical Computing Environment (SCE). DSP work must remain in this space, separately from CSP work which should remain in the platform like GitHub ®. This separation can be summarized as the following.

1. Production environment (SCE) – this will keep stable packages installed and should be used for all regulatory submission works. Any modified or extra packages used for projects (studies) must be separately qualified at the proper level and the users will manage their derivations.
2. Development environment (GitHub or any other development platforms) – this will provide sandbox playground where users can develop and test new packages. There can be multiple of this type of environment – it is relatively easy to create these with the "Container" technology such as Docker – and any works done here should be considered draft and proof of concept (POC) only (i.e., they should not be used for any regulatory submissions).



NOTES ON GITHUB (OR OTHER GIT PLATFORMS SUCH AS GITLAB)

GitHub is the de facto standard of OSS platform for collaborating projects across industries. It is no doubt that it has features that attract OSS programmers all over the world. However, the materials previously discussed so far leads to this question – is GitHub an ideal platform for supporting businesses like Pharma who focuses on the generation of data and its science, especially with their nature of being proprietary? Matching GitHub's openness with Pharma's sensitive data content obviously presents a huge challenge.

1. GitHub is not designed to be a platform for data scientist. It is a platform for CSP, not DSP. The learning curve for Git is also a limiting factor for DSP as the workflow is different and its technicalities.

2. The openness of OSS and closedness of Pharma is not compatible. However, with the clear distinction between CSP vs. DSP, CSP part of Pharma can use GitHub as the main platform for OSS and tools (CI/CD).
3. This means DSP can use GitHub for OSS that does not violate data privacy.
4. There are use cases where this is proper. One is sharing the metadata. E.G., repository – this should be a requirement than a recommendation due to its usefulness of sharing the repository/package information with the RAs. This will simplify the reproducibility of programming environments.

PRAGMATIC OSS IN PHARMA

DOUBLE-EDGED SWORD OF OSS

As a DSP, it was an easy path to work with commercial proprietary software. They were bought and installed by IT department with optional components, and it required no user intervention to get them installed on a personal computer or server. Any qualification steps would have been taken care by the established Standard Operating Procedures (SOPs). Any programs/macros written will fall on the user to do QC individually according to the SOPs. (e.g., Double-Programming, Peer-Review).

The main challenge of OSS which has double sides is its capability to diversify without limitation. Anyone can download, change, and enhance OSS – meaning it can generate infinite number of changes and versions of similar programs. Tidyverse, which is a popular OSS package for data science works in R has a broad number of users and developers constantly contributing improvements. This volatile nature makes it hard for the administrators who manage the R programming environments because the work of pharmaceutical data science needs to be regulated and controlled as per the GXP requirements (i.e., The latest and greatest features are not necessarily beneficial in this environment).

THE R DILLEMA

Although R has its roots in statistics, it is still a general programming language. What does this mean? It means it has both CSP (logic), and DSP (data) aspects embedded in the language. This is the major misunderstanding when it comes to the responsibilities of programmers in Pharma – we are unknowingly asking a DSP to do the job of CSP because the line between them became unclear with OSS. This is also a reason there are big debates in the R community about “Base R vs. Tidyverse” – which is the best way to train a new R programmer. For more on this, read https://github.com/matloff/TidyverseSkeptic. The same analogy applies to the Python vs. Pandas © argument.

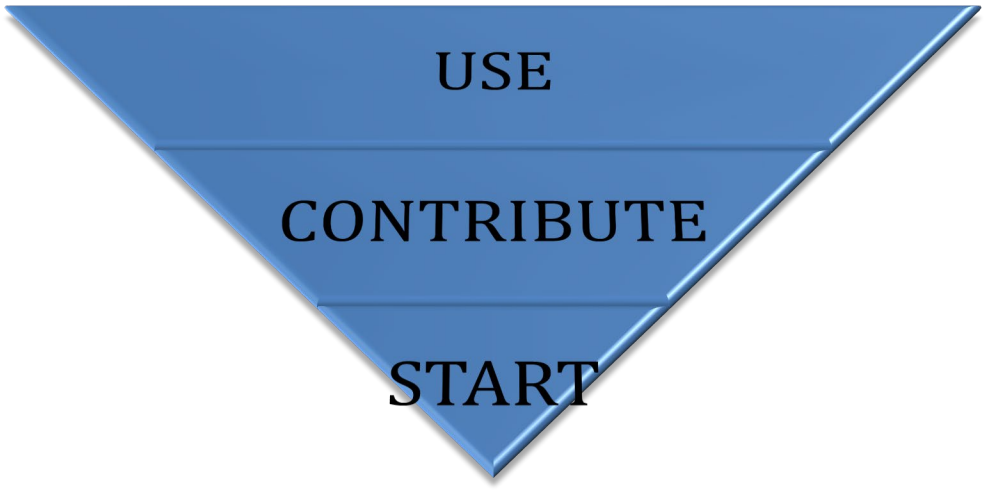
Tidyverse is a data science package (i.e., implemented on the top of Base R) that make DSP’s job easier. It is perfectly acceptable to do the DSP work with the base R; however, it is harder to learn and use. You will lose the benefit of understanding the underlying complexity of R when you are trained in Tidyverse, but it will help new learners of R to get themselves up and running much faster – this is an invaluable advantage in training and migrating hundreds of programmers to R. Being a DSP with R means you will be the *USER*. You will be the expert of using the existing packages including Tidyverse to produce *data* output.

In contrast, if you are coming from CSP backgrounds, you are better equipped to be the base R programmer and be the *DEVELOPER* of R. You will manage writing functions and packages to support the DSP who will use them. For example, you don’t have to be the expert in the data outputs such as Tables, Listings and Figures (TLFs) but you will have a better understanding of how to optimize the data structure to speed up the process of millions of rows of such data outputs.

Note that these two distinct roles should always be in communication and complement each other to understand what’s required in the whole organization.

MIX AND MATCH OF ROLES

I have proposed the following ratio for the use, contribute and start of all Pharma-related OSS projects in my earlier presentations/talks. I do not oppose the idea of someone performing both CSP and DSP roles in general; however, this should be avoided unless there is a shortage of programming resources, or the programming group is small. Basically, asking someone to master in both roles can be demanding and create more complication.



1. USE – The majority will only "use" the OSS products as part of conducting their other projects/studies. This would be about 90-95% of the entire programming group, mostly consisted of DSP.
2. CONTRIBUTE (or DEVELOP) – People will choose to improve the existing OSS. They should have the technical skills to program in CSP topics yet have the good understanding of the work of DSP. They should account for 5-10% of the programming group, mostly consisted of CSP.
3. START (or LEAD) – A miniscule number of people should be in this category, capable of starting or leading a Pharma OSS project. The person is not only technically strong but must possess the leadership, enthusiasm and vision to make sure that the project takes off and maintains the stable status, which means that the project must be able to deliver milestone products on a planned schedule while maintaining a healthy community of users and developers. This type of programmers is usually accounted for 0-1% of the entire group.

TEST AND VALIDATION (QUALITY CONTROL OR QUALIFICATION)

Testing a program is not the same as validating it – this is an important distinction that can save time and prevent headaches. People use these two terms interchangeably and I strongly recommend keeping them separate by the role of programmers. Tests should be kept with CSP using a reliable workflow such as GitHub Actions ® (CI/CD), while validation should be done with DSP using data-oriented SOPs to verify the accuracy and reproducibility.

1. 100% bug-free or "perfect" outputs are not possible. "Fit-for-purpose" or "Risk-based" approach is needed to balance the risk and success.
2. A set of standard checks are useful to raise the overall quality of programs and data and scheduled workflows to run, support, and release them are necessary.
3. Qualifying the outputs by duplicating them (i.e., double-programming) does not make sense for CSP-type works. Trying to double-program R Shiny applications or their outputs is a clear sign that you are mixing up the works of CSP and DSP.
4. The ultimate purpose of quality control for CSP and DSP are simply different. CSP may use test data to verify that program logics and algorithms are working as intended. This process optionally uses test data and the expected results, depending on the data contents. DSP often use the comparison of two data outputs to verify that they are identical in their contents and metadata.
5. Use of OSS in validation presents a unique opportunity. While the main tool for comparing outputs in proprietary software has only been limited, there are more options available in OSS that can perform the similar job at various levels of details.

SPECIAL NOTES ON INTERACTIVITY

Adding interactivity to static content such as TLFs is another aspect of data visualization. For R, R Shiny ® and R Markdown ® generate creative, dynamic contents in traditional static reports. However, moving your works from the R scripts to R Shiny crosses the line between the job of DSP into CSP. Also, it is still in early phase of development for submission purpose so it should only be considered for augmenting the existing solution at this point. Things to consider when this happens.

1. All your R scripts will be "shadowed" behind Graphical User Interfaces (GUI) and embedded algorithms, which means they will not be visible to the R Shiny users. Now you are no longer generating data outputs as DSP – you are building an application which should be managed by CSP. A separate process and practices are needed to verify the performance of the Shiny app including testing (CSP) and validation (DSP).
2. Pfizer has open-sourced our internal R Shiny framework for reproducible content at Pfizer Open-Source area at https://github.com/pfizer-opensource/shiny-framework.
3. R Markdown is recommended if you want to keep the project in DSP. R Markdown (or Quarto ®) is a document, and it will only require validation of the output, not the testing (or full-development cycle of CSP). All codes/scripts are accessible, and any Quality Control (QC) works can be addressed directly on the document.
4. For production use, there must be a standard on the smallest requirements that all interactive applications (R Shiny) and documents (R Markdown/Quarto) must meet, something equivalent for static contents. And the application components must pass *both* predefined tests (for the app) and quality checks (for the document) components (such as HTML, PDF, or MS Word ® files).

CONCLUSION

1. Both DSP (users of OSS) and CSP (developers of OSS) are important and require unique skills in their roles. It is important to understand their different working domains to set up their working parameters and responsibilities. Mixing up the terms like testing & validation can introduce confusions and delays in implementing OSS solutions.
2. The ecosystems needed by CSP and DSP must be separated but harmonized.
    a. CSP's space needs to be with the OSS community, and it is not recommended to create company-specific packages or solutions as this will complicate the submission process in the downstream. Starting a new OSS project needs a careful consideration to meet the minimum userbase to sustain it – it is easier and more productive to work with existing, stable projects to improve upon the common goals and avoid redundancies.
    b. DSP's work needs to be in the closed-system and no OSS projects should be started in this space. The system (SCE) needs the controlled environment where DSP will remain as only ***users*** of OSS – no new packages should be created or developed. Functions are recommended and needs the study/project level validation, separate from the system-level validation.
    c. The communication between CSP and DSP are essential to succeed. Information Technology (IT) and a dedicated group's work is needed to coordinate the efforts with clear goals and missions to loop feedback between the groups.
    d. Relevant training materials need to be developed and delivered to specific programming groups to improve the learning experience – I recommend creating two different career paths for each CSP and DSP programming groups to maximize efficiency and speed up the process. Pfizer has developed our own training program using R Markdown and open-sourced the content here - https://github.com/pfizer-opensource/pharma-hands-on-exercises.

3. Senior management's commitment to OSS is a key factor in implementing successful OSS solutions. You need dedicated resources to understand and support the constantly innovating space (OSS with CSP) and the controlled moderate-paced space (SCE with DSP).

FUTURE WORKS

There are selective topics where more research and discussion can benefit to improve the overall OSS implementation strategy for Pharma in relation to the roles of programmers.

1. OSS-based data science programming platform for handling proprietary data
2. Modern Quality Control and Validation opportunities with OSS
3. Use cases of open-sourcing proprietary metadata such as packages or ML models for regulatory submissions

DISCLAIMER

This paper has information or directions that my employer is taking, while some are not – All information is provided "as-is" and opinionated; no assurance or promise is given by myself or my employer. Contents were reviewed and aided by AI (Microsoft Copilot ®).

ACKNOWLEDGMENTS

I like to thank my colleagues at Pfizer's SWAT (Scientific Workflows and Analytical Tools) team for their inspirations and contributions. They are Mike Smith, Natalia Andriychuk, Samir Parmar, and Narayan Iyer.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: James J. Kim
Company: Pfizer Inc.
Email: james.j.kim@pfizer.com
Website: https://github.com/pfizer-opensource