PhUSE US Connect 2019

Paper XXnn

Modernizing the Safety Review Toolkit Using Open-Source Software

Daniel Choi, IBM, Arlington, VA, USA
Austin Taylor, IBM, Arlington, VA, USA
Derek Ahneman, IBM, Arlington, VA, USA
Joy Li, US Food and Drug Administration, Silver Spring, MD, USA
Bobbie Witzczak, US Food and Drug Administration, Silver Spring, MD, USA

Abstract

The Office of Computational Science (OCS) is currently modernizing software tools used in the safety review process by transitioning to open-source technologies. Previously, SAS scripts modified by medical reviewers were used to generate static reports. This strategy had the following disadvantages: (1) the requirement for expensive licenses; (2) the need for SAS to be installed locally; and (3) reviewers were responsible for editing scripts themselves, despite varying levels of technical ability. We found that moving to an open-source, web-based framework solves these problems and allows for the development of interactive and intuitive tools. OCS has created a collection of web applications related to safety signals that generate downloadable reports. These tools have drastically improved the user experience. Notably, the need for coding expertise has not been eliminated, but rather shifted from medical reviewers to OCS. This paper discusses the details of this transition.

Introduction

The mission of the Office of Computational Science (OCS) is to provide medical reviewers in the Center for Drug Evaluation and Research (CDER) with innovative and reliable solutions to make the scientific review process more thorough and efficient. To this end, OCS creates software tools that can automate common exploratory and safety analyses. These tools generate a collection of reports to assist clinicians and statisticians in the regulatory review process. Two examples of these outputs include a demographic baseline report and a drug-induced liver injury report. In the past, these tools largely consisted of a collection of SAS® scripts. Recently, OCS has begun transitioning this toolbox from SAS scripts to R Shiny web applications. Herein, we describe the resulting benefits in cost, distribution, and usability.

Attempted Solutions within a SAS-Based Framework

Although effective in generating static outputs, using SAS scripts carries some disadvantages. One obvious drawback is cost, as the proprietary software requires an expensive annual subscription for each user. Distribution is also inelegant; the scripts are shared via email or through a repository, which
requires the user to download the script, tailor the code to their application, and run it. This process also requires that the user have the software installed locally, a process which often causes undue delay. Finally, this procedure places the burden of modifying the scripts on medical reviewers who have varying levels of programming experience. OCS sought solutions to overcome these challenges – cost, the need for local software installation, and the need for reviewers to modify computer code.

OCS first explored ways to improve distribution and usability within a SAS-based framework. The scripts were initially deployed in various ways, including being run by analysts at OCS as well as being sent to users for their own modifications. In some cases, scripts were modified to run on remote workstations to obviate the need to install SAS locally. However, this strategy still required reviewers to manually modify computer code, an acute problem if the user was not familiar with SAS programming. Ultimately, this solution was short-lived, as remote workstations proved slow and unreliable due to congestion. To improve usability, OCS also attempted to minimize the necessity for medical reviewers to modify code by creating an Excel form which passed parameters to the SAS script. However, this solution was suboptimal because reviewers were often unsure of which variables to use and typing in variable names manually resulted in frequent spelling errors. OCS therefore became interested in using a technology that was more interactive and could provide options based on the variables present in the clinical trial dataset.

Transition to Open-source Web Applications

In open-source software (OSS), source code is released under a license where the copyright holder grants users the rights to study, change, and distribute the software to anyone for any purpose.\(^1\) R is a programming language which is widely used among statisticians and data miners for statistical computing.\(^2\) Unlike SAS, there is no cost associated with purchasing licenses to develop tools in R. There also exist myriad well-maintained libraries in R which can cut development time significantly. R can also interface with SAS and vice versa, allowing OCS to minimize disruptions in service during a transition. Finally, the use of object-oriented programming (OOP) in R can assist developers modularize, organize, and reuse code. OOP is a powerful paradigm to tackle complex problems and the flexibility of R’s available data structures relative to SAS accelerates development timetables and avoids the accrual of technical debt.

Early efforts by OCS to transition to an open-source environment focused on revamping the Demographic Subgroup Analysis Tool, a tool which helps reviewers fulfill the FDASIA requirement mandated by Congress\(^3\). The previous version consisted of a SAS script populated via an Excel form. Instead, we proposed using R Shiny, an R library which creates an adaptable web-based front end for analysis scripts. The creation of web applications using R Shiny obviates the need to install software and streamlines the deployment of these tools. This strategy also removes the requirement for medical reviewers to interact directly with computer code. Furthermore, as the front end is prepopulated with the data elements corresponding to the dataset uploaded by the user, there are fewer input-related errors. It should be noted that although the analysis is prepopulated with relevant variables based on data standards, the reviewer still inevitably needs some level of familiarity with the study datasets. Moreover, the interactive nature of web applications facilitates exploratory work by the reviewers. Finally, R Shiny web applications can generate downloadable reports in Word and Excel, a feature important to medical reviewers.

Creating web applications also requires coding a user interface (UI). Therefore, making web applications instead of simple scripts generally lengthens overall development time and requires more maintenance. However, in our experience, the improvements in facile distribution and enhanced usability have justified this time investment. Another inherent hurdle with web applications is the requirement for dedicated servers, the number of which will depend on the size of the userbase. Dedicated servers have not yet been procured and we have resorted to alternatives in the interim. One option considered was for the Office of Information Management and Technology (OIMT) to push R to all local machines like other
commonly pushed software (e.g., Microsoft Office Suite), but (1) versioning and package management would be difficult to manage at scale and (2) this would be similar to the current requirement for SAS installation. We therefore leveraged shared work folders to store all necessary components of the web application and wrote a batch script to load them on the user’s local machine each session. OCS is in the process of working with OIMT to move the web applications to dedicated servers.

Conclusion

The Office of Computational Science strives to make the regulatory review process more efficient and robust. This paper describes some recent strategies for accomplishing this goal by moving toward open-source technology and using a web-based framework for tool delivery. The creation of web applications allows for more intuitive and interactive safety review tools; recent user acceptance testing (UAT) revealed positive feedback from medical reviewers. We anticipate this process will continue to shift technical work in generating safety analyses from medical reviewers to OCS.

References