PharmaSUG 2018 - Paper SI-13

Validating R – Part of the Uphill Battle in the Pharmaceutical Industry

Peter Schaefer and Debra Fontana, VCA-Plus, Inc.

ABSTRACT

R, the open source language for statistical computing, is widely accepted by data scientists across many application areas. However, several myths around R still seem to prevent its acceptance by the pharmaceutical industry. In this paper, we make another attempt to disprove the ‘R can’t be validated’ thesis and show the path to a validated R system. As a bonus, we will describe a concept for a powerful validated system for data analysis and reporting.

Born as beta version infant in 2000, R, the open source programming language for statistical computing, has made an astonishing career: It has grown in popularity by academia and industrial users to surpass other languages. As much as there seems to be this paradigm shift across many industries, the pharmaceutical industry is behind the curve and a number of myths around R and its usability in regulated clinical trial environments are certainly hard to eradicate.

INTRODUCTION

Like it or not – pharmaceutical companies must validate their computerized systems. It’s a burden and one even might argue it’s stifling and slowing innovation. This might be an exaggeration; however, it is evident that the hurdle of computer system validation is costly and often slows the rate of adoption of newer technologies. This validation requirement might be only one reason why a powerful software tool like R is not as widely adapted by the pharmaceutical industry as it is by other industries; however, it certainly is an important one. Of course, the fact that R is an open source system doesn’t help because there is a good level of uncertainty around the use and validation of open source tools in regulated environments.

This paper describes the general context for computer system validation and the specifics around validating an R-based computerized system. You will see, that validating R is not that much different from the accepted approach for validating other systems with some specific changes. A review of some presentations and documents published during recent years rounds up the picture: R can be validated as every other computerized system and the process and effort will not be much different from validating other systems.

THE BIG PICTURE: COMPUTER SYSTEM VALIDATION

FIRST, LET’S START WITH SOME DEFINITIONS

Software (as defined by the American National Standards Institute (ANSI) and as included in the Food and Drug Administration’s (FDA) Glossary (FDA 1995)) is “Programs, procedures, rules, and any associated documentation pertaining to the operation of a system. Contrast with hardware.”

Hardware (as defined by International Organization for Standardization (ISO) and as included in the FDA’s Glossary (FDA 1995)) is **“**Physical equipment, as opposed to programs, procedures, rules, and associated documentation. Contrast with software”.

A **Computerized System** (as defined in the FDA’s Glossary (FDA 1995)) “Includes hardware, software, peripheral devices, personnel, and documentation; e.g., manuals and Standard Operating Procedures. See: computer, computer system”; a Computerized System (as defined by the Organisation for Economic Co-operation and Development (OECD)) is “a set of software and hardware components which together fulfill certain functionalities”.

Software Validation (as defined by the FDA in its guidance document (FDA 2002)) is “confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled."

Computer Validation (as defined by OECD (OECD 2014)) is "demonstration that a computerized system is suitable for its intended purpose".

In short, **computerized system validation** is "documented evidence that a computerized system (software + hardware + people + documentation) does what it is intended to do.

There are many documents available which provide guidance on how to perform computerized system validation. One commonly referenced standard is GAMP®5 from the International Society for Pharmaceutical Engineering (ISPE 2008). The framework defined in GAMP 5 provides a comprehensive approach to computerized system validation that is generally accepted within the industry. Perhaps the most commonly-referred to bit of GAMP 5 is the software categorization scheme. The classifications were changes slightly in GAMP 5 as compared to GAMP 4 and are shown in Table 1.

Category	GAMP 4	GAMP 5
1	Operating System	Infrastructure (OS, DB, MW, etc.)
2	Firmware	No longer used
3	Standard Software Package	Non-configured products
4	Configurable Software Package	Configured products
5	Custom Software	Custom applications

Table 1. GAMP Categories for Computer Systems

A second, important document about required computer system validation is the FDA's 2002 Software Validation Guidance, issued by FDA's Center for Devices and Radiological Health (CDRH) and Center for Biologics Evaluation and Research (CBER). Two remarks about this document:

1. This guidance document was not issued by FDA's Center for Drug Evaluation and Research (CDER). However, CDER has not issued an alternative document and in the absence of other CDER guidance, use of this document for computerized systems used to develop drugs seems prudent.
2. The title of the document refers to "Software Validation" and it is "applicable to the validation of medical device software or the validation of software used to design, develop, or manufacture medical devices" – remember CDRH issued this document. Nevertheless, the concepts are applicable to implemented computer systems (i.e., systems that include software + hardware).

WHY RISK IS YOUR FRIEND

While the GAMP 5 categorization scheme is interesting, its utility seems limited. The bottom line regarding these categories is that as the category number increases, the risk associated with the computerized system incorporating the software and the length of the qualification/validation documents likely increases. Ans because typically the effort for validating the computerized system increase as well, you will be looking for way to control and limit the overall effort. This is where the risk assessment comes into play.

Risk is a key consideration in GAMP 5, and GAMP 5 provides a useful tool for assessing risk. In a two-stage process, priorities for a validation process are determined. In the first step, you depict severity of a problem against the probability for the problem to occur (see Figure 1).

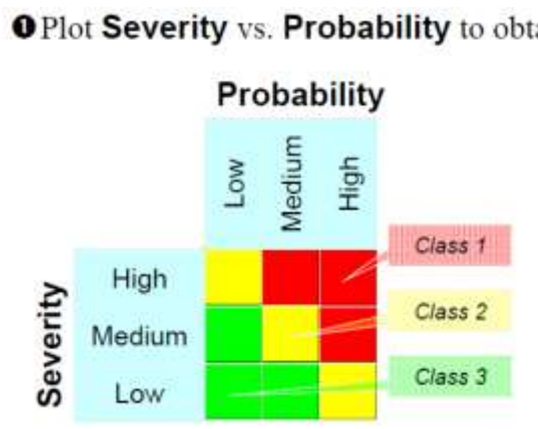

Figure 1. Probability versus Severity to Determine Risk Classes

Then in the second step you depict the risk class against the detectability of a problem to determine the priority for the validation process (see Figure 2.).

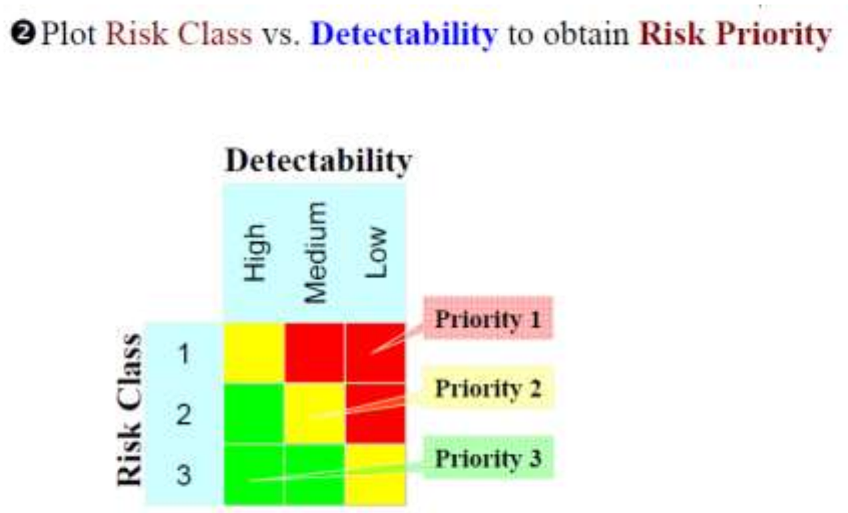

Figure 2. Detectability versus Risk Class to Determine Priority

During the validation process, the focus should be placed on the risks with the highest Risk Priority, with possible risk mitigation strategies including changing the process with the risk, changing the design of the computerized system, adding new features to the computerized system, or applying external procedures (i.e., procedural controls, SOPs). Note that the risks to be considered include business risks (e.g., what happens to the business when the computerized system is unavailable for whatever reason) and compliance risks (i.e., what regulatory authority action could be imposed on the organization or what data integrity issues could arise if the computerized system does not function as intended). And most importantly, the risks to be considered include risks to patient health and safety if the computerized system does not function as intended and decisions are made based on the output which adversely affect patients. After all, to limit or even eliminate this risk is the whole purpose of the validation effort, right?

As evidenced by this risk mitigation process within GAMP 5, the guidelines apply not only to validation activities, but also to a variety of other processes within an organization, such as supplier relations and system maintenance. And while GAMP 5 describes a flexible, risk-based approach to compliant GxP regulated computerized systems, its history is based in equipment qualification and process validation (note that GAMP is an acronym for “Good Automated Manufacturing Practice”). For these reasons, GAMP 5 can seem a bit complicated and overwhelming when you just want to validate a computerized system which doesn’t control manufacturing equipment. And as a final remark, consider that because the GAMP 5 framework are guidelines, not regulations, it is not mandatory to follow this methodology.

YOU NEED TO GO BEYOND GAMP

Abbreviation …	… stands for
IQ	Installation Qualification
OQ	Operational Qualification
PQ	Process Performance Qualification or Product Performance Qualification

There is a very specific issue with Gamp 5. As the name indicates (Good Automated Manufacturing Practice) GAMP 5 is focused on control manufacturing equipment. And some of the terminology used in GAMP 5 is either not applicable to a computerized system or is outdated. This specifically applies to the terms IQ, OQ, and PQ that are not used anymore in the FDA's 2002 Software Validation Guidance. Though these terms are defined in the FDA Glossary (FDA 1995) as follows:

- **IQ** (Installation Qualification) is "establishing confidence that process equipment and ancillary systems are compliant with appropriate codes and approved design intentions, and that manufacturer's recommendations are suitably considered."
- **OQ** (Operational Qualification) is "establishing confidence that process equipment and sub-systems are capable of consistently operating within established limits and tolerances."
- **PQ** (**Process** Performance Qualification) is "establishing confidence that the process is effective and reproducible" or **PQ** (**Product** Performance Qualification) is "Establishing confidence through appropriate testing that the finished product produced by a specified process meets all release requirements for functionality and safety."

Over the years, people have attempted to apply these IQ, OQ, PQ terms and definitions to computerized systems, even though based on the above definitions, for the most part they clearly don't apply. For example, some people have stated that IQ is used to verify the Design Specification requirements, while OQ is used to verify both the Design Specification and the Functional Specification requirements, and PQ is used to verify the User Requirements Specification requirements. But this construct is overly complicated for Commercial Off-the-Shelf (COTS) software and Software-as-a-Service (SaaS) implemented computer systems because the Design Specification is a document that the software vendor likely wrote when developing the software (which doesn't need to be (and in many cases cannot be) tested by the end user, and because if functionality is defined in both the Functional Specification (typically stated as being written by the software vendor) and the User Requirements Specification (typically stated as being written by the end user), then should that functionality be tested in OQ or PQ? Others have stated that IQ is "Configuration Testing", OQ is "Functional Testing", and PQ is "Requirements Testing". But this construct doesn't work either because there can be both configuration requirements and functional requirements, so should these requirements be tested in IQ (configuration testing) or OQ (functional testing) or in PQ (requirements testing)?

The FDA doesn't try to use the IQ/OQ/PQ terms in its 2002 Software Validation Guidance. This guidance document states "For many years, both FDA and regulated industry have attempted to understand and define software validation within the context of process validation terminology. For example, industry documents and other FDA validation guidance sometimes describe user site software validation in terms of installation qualification (IQ), operational qualification (OQ) and performance qualification (PQ). While IQ/OQ/PQ terminology has served its purpose well and is one of many legitimate ways to organize software validation tasks at the user site, this terminology may not be well understood among many software professionals, and it is not used elsewhere in this document." This guidance further states "Terminology regarding user site testing can be confusing. Terms such as beta test, site validation, user acceptance test, installation verification, and installation testing have all been used to describe user site testing. For purposes of this guidance, the term "user site testing" encompasses all of these and any other testing that takes place outside of the controlled environment used by the developers. This testing should take place at a user's site with the actual hardware and software that will be part of the installed system configuration." So the FDA has made the testing part of computerized system validation easy to understand and organize; if the end user is doing testing of the computerized system, then use the actual hardware and software (or cloud-based implementation) that make up the computerized system and call it "user site testing", without worrying about where IQ, OQ, and PQ start and end. It is all just "testing".

One item of note regarding IQ: The "establishing confidence …… that manufacturer's recommendations are suitably considered" bit of the definition of IQ does apply to computerized systems. To ensure that the computerized system functions as the software vendor expected, and to ensure that the testing that the vendor did is not invalidated, when implementing a computerized system, consideration should be given to the software vendor’s requirements or recommendations regarding hardware (e.g., processing speed requirements, RAM requirements) and other software (e.g., operating system version, database version, Microsoft Office version). An Installation Qualification protocol for the computerized system which documents the vendor’s requirements/recommendations and the actual state of the computerized system provides the confidence that is mentioned in the IQ definition. This IQ which documents the components of the system also is useful as a disaster recovery tool, in terms of knowing what hardware/software you need to acquire to rebuild the computer system.

So, now that we know that we can just do “user site testing” rather than worrying about OQ and PQ (and FAT, UAT, system testing or whatever else it has been called), what do we test during a validation project? Because the purpose of validation is to demonstrate that the computerized system does what it is intended to do, the functionality of the system that is actually used should be tested, using a risk-based approach, meaning that not every bit of functionality must necessarily be tested. The functionality of the system should be documented in a Requirements Specification. Note that there is really only the need for one Requirements Specification; there is no need for a User Requirements Specification, and a Performance Requirements Specification, and a Configuration Specification, and a Functional Requirements Specification, and any other type of requirements specification. Put all of the requirements, whether they be for security or performance or functionality or whatever in one document and move on. Keep it simple! Also note that the Requirements Specification should be written by the organization that will use the computerized system. Because validation relates to the intended use of the computerized system, and only the organization that will use it knows how they intend to use it, they must document the requirements. Do not use whatever requirements specification has been provided by a software or SaaS vendor for this purpose; this document will be insufficiently specific (e.g., it will state that the system provides configurable workflows, but not the specific workflows that will be used) and will likely include all of the functionality in the system, even the functionality that will not be used. Of course, use any documentation provided by the vendor to “jump start” your Requirements Specification, but ensure that the specification documents how you intend to use the system and ensure that it is approved by appropriate personnel from your organization.

Any validation project is very much about creating and collecting evidence of what has been done. Table 2 shows the artifacts that should be created during a typical computerized system validation project.

Document	**Content and Purpose**
Validation Plan	Documents • **When** you are doing the system validation (hint: it should be prospectively before the system is used for “real work”; the time for retrospective validation is long gone.) • **Who** will be involved as the Validation Project Team (include users, management, IT, QA, and a validation professional (either internal or contracted)) • **What** deliverables will be produced by the validation project (i.e., this list of artifacts)
Requirements Specification	Documents the intended use of the system and the resulting requirements that need to be tested. Note that this specification will have a focus on usage of the system and therefore will typically be different from a design document or the code specification that would be created by the vendor of a software program.
Risk Assessment	Documents the risk assessment and provides background for the priorities and focus during the validation process.

Document	Content and Purpose
Installation Qualification	Includes all components of the computerized system. A good structure would be to have one section for the R installation (including operating system version for the computer, the version of R, what R packages are installed, what other software is installed), another section for the data storage location, another section for the script library location, and another section for the output storage location.
Test Plan	Outlines the “scope and approach” of the testing, the order in which test script(s) will be executed, who will execute the testing, and who will review the executed test cases
Drafted Test Cases	Test cases which test the requirements documented in the Requirements Specification
Executed Test Cases	The test cases which have been executed by the testers and which include objective evidence (e.g., screen shots, reference and test output)
Testing Summary Report	Summarizes the overall testing effort, including how many test cases were executed, how many passed/failed, what investigations revealed about the failures, and how users should be notified of processes that are changed due to the test failure (hint: in the SOP for using the R-based computer system)
Traceability Document	Traces each requirement in the Requirements Specification to the test case(s) which tested the requirement, and the status of the requirement, i.e., “Passed” if all of the related test cases passed, “Failed” if any of the related test cases failed.
Validation Summary Report	Summarizes the activities that were completed during the validation project, documents any deviations from the Validation Plan, and states that the system is acceptable for routine use and is “released to production”.

Table 2. Typical Artefacts of a Validation Project

While this list of documents may seem long, most of these documents are relatively short: Maybe 2 pages each for the Validation Plan, Test Plan, and Validation Summary Report for the shortest document, but typically some more pages for the Requirements Specification as the longest document, certainly depending on the complexity of the computer system.

NOW WHAT ABOUT VALIDATING R?

This paper has so far only discussed validation of computerized system in general, so now let’s look at validation for a computerized system including R and R scripts as the software. The previously described concepts translate into the following consideration for validating R:

- Because R does not operate in a vacuum, but requires input and produces output, it is required to consider not only where R resides and what it does, but also what data it uses for input and where that resides, what output is produced and where that is saved, and, if scripts are re-used, where the script library is located. All of these bits could be considered as part of the organization’s “analysis and reporting computer system” and this whole system could be validated with one computerized system validation activity.
- Regarding requirements: The intended functionality of R is to provide a scripting environment, which is used to develop scripts which can be executed to perform specific functions. So, there are likely few requirements that relate to the R software itself. But consider the other parts of the computerized system: What requirements are there related to security of the data storage location and the output storage location? Are there specific requirements around existing scripts? For example, who can write and execute scripts or how are existing scripts protected against modification? What

requirements are there related to number of concomitant users and processing time? What requirements are there related to input and output file types?

- Regarding user site testing: Testing should demonstrate that the documented requirements are satisfied by the implemented computerized system, so draft and execute test cases which demonstrate the requirements. Use “business process workflows” which test how a user would interact with the system from end-to-end; test that only users who meet the security requirements can get to the input data, test that the system allows the user to program a simple script (if that’s a requirement), test that the script can be saved to the script library and retrieved from the library for execution, and test that output produced by the script can be saved to the output location in a way that downstream processes can use the output.

The described steps and artefacts already provide a solid framework for validating an R-based system, but there are still some additional issues to consider:

- The validation process and the deliverables defined above are appropriate for any computerized system which does not control manufacturing equipment or laboratory equipment, not only for a system including R. The difference for other types of computerized systems is that the documents may be longer, especially the Requirements Specification and associated test cases for systems with more functionality or with higher risk. The validation process that you follow, and the required deliverables should be documented in an SOP.
- Concerns that a computerized system containing Open Source software cannot be validated are unfounded. Open Source software such as R can be incorporated into a computerized system as long as the software and version (and associated packages) are documented in the IQ documentation for the system, as long as the R version is not changed in the system without an appropriate change control process, and as long as user site testing demonstrates that the Open Source software fulfills the documented requirements of the system. While Open Source software may not come with all of the documentation that often accompanies single-vendor software resulting is less assistance from the vendor when drafting your own documentation, simply being Open Source does not disqualify software form being used as part of a validated computerized system.
- When using a scripting software like R (or SAS or S+), validating the computerized system is only half of the story. You must also consider how you will validate and control the scripts that are created to perform intended functions. For example, each script that is created to perform a particular function, especially if you intend to re-use the scripts across drug compounds or client projects, should have documentation around what is does (i.e., a description of the script, a “mini-requirements specification” regarding what the script does and with what types of input data it is appropriate for use), and how it was tested (i.e., with only one dataset for one compound/project or with a variety of datasets for multiple compounds/projects, by being QC reviewed by a second person to verify that it will do what is intended). Each script should also be version controlled, and only changed with an appropriate change control process.

  And in this context an additional observations: Given that with server- or cloud-based systems, it is easier than ever to make applications (including scripts) available to non-programmers (or “script writers”) it seems a logical step to deploy re-usable scripts as hosted applications, validate and version-control the underlying scripts and the hosting platform and thereby reduce the overall validation effort significantly.
- The validation process and deliverables described above exclude many other activities and deliverables that many computerized system validation guidelines/SOPs require. This is because this process assumes that computerized system validation is merely one part of your overall Quality Management System (QMS), and that the QMS will require completion of other activities/deliverables. Further, this process assumes that you have a defined process for “implementing computerized systems” and that validating the computerized system is one component of the implementation project. For example, other QMS elements in your organization are your “SOP on SOPs” which requires that SOPs exist and defines how they are drafted/reviewed/approved, and your SOP on personnel training which requires that personnel be trained and defines how training is documented. Therefore, it is not necessary that SOPs and training be part of the computerized system validation

project and be produced as deliverables. Rather, these items should be part of the overall "implementing the computerized system" project, which also includes project approval, project management, software product identification and selection, vendor assessment, system implementation, computerized system validation, backup/restoration/disaster recovery process and SOP implementation, user SOP implementation, and performance of training. Further, your QMS should include elements which assist in maintaining the validated status of the computerized system once it is validated, specifically change control and periodic review processes and SOPs.

COMMENTS ON SOME PUBLICATIONS ON VALIDATING R

Over the last years, an increasing number of publications is dealing with use cases for R and some touch on how to validate R and R-based systems. In this section, some of these documents are summarized to give you an idea about what others have done and experienced.

- Most prominently, there is the guidance document about the use of R in a regulated environment published by the R Foundation (R Foundation, 2014). The name of the document, "R-FDA.pdf", sounds very promising and someone might believe that this is simply the required reference to show that R is validated. That is certainly not correct. The paper provides the background for validation in general and what is more important insight into the software development lifecycle that the R Development Core Team is following. This is important, but it is only a small bit of what you need for validating R, at this point, you have not the "documented evidence" that this is what they actually did. And for sure, there is no evidence that the system does what you intend to use it. A second point to note is that you really need to verify that the scope of this document matches your system: The document only applies to the core R distribution, many of the packages that you might use in a powerful R based system are not covered. Clearly, this doesn't mean that additional packages can't be part of your validated R system, but you can't rely on this document for their validation.

  One section of the document might turn out specifically useful when planning your validation project, the section on 21 CFR Part 11 compliance. The document discusses several of the requirements and explains the R Foundation's position with respect to these requirements. Whether you follow their thought or take another position, it certainly is valuable input.

- Of course, the pharmaceutical industry is always looking at the FDA for guidance – literally and figuratively. However, sometimes the FDA's message doesn't seem to get through: Nowhere in the guidance documents is suggested that open source software – such as R – can't be used. In fact, the FDA issued specifically a statement that clarifies that it does not require the use of any specific program for statistical analysis (FDA, 2015). The statement clarifies that the appropriate validation of the computer system that is used by sponsors is all that is required. Other publications by FDA staff (with the usual disclaimers that it's not the FDA's position) describe about how R is already used at the FDA (for example, Soukup, 2007; Brodsky, 2012). Mat Soukup summarizes the validation requirements and goes on to describes the then current status of R usage at the FDA: Since 2004, R has been approved as a non-standard software at the FDA, and since 2007, this extends to all versions of R. In his presentation, he already predicts that the flexibility and power of R would sell itself. A series of publications since then confirms this prediction (Brodsky, 2012; Kyun-Seop, 2017; Wong, 2017; Taylor, 2018). These posters describe a broadening basis of R usage at the agency, the co-existence of R and SAS, and that the FDA has accepted R as a tool as any other tool to be used where appropriate.

- An interesting presentation on using open source software in a regulated environment by a pharmaceutical company is the one by Anthony Rossini and David James. They nicely capture the requirements for statistical software systems and the mental reservations – including the typical red herrings and FUDs when dealing with open source software. What makes this presentation worth reading is the section on the real issues with open source software – and specifically R – in the context of a pharma company: Issues such as potential uncertainty about future development, concerns about the release cycles, when there is not one vendor controlling consistency of releases, and the potential cultural clash between a vibrant open source community and internal processes that might be slower.

- Finally, we cannot conclude a paper on validating R without mentioning that there is a validated version of R offered by a vendor (Mango Solutions, 2018). This vendor has created an open-source-based product bundling the R distribution and tested R packages into a commercial product. This certainly helps companies to get a head start, but as with every other COTS product, there will be the validation requirements as described earlier in this paper.

CONCLUSION

Of course, validation of computerized systems is a burden, it costs money and time, it might delay adapting newer systems and technologies. Sometimes, it's hard to get people to admit that there is value in knowing that your systems do what they are supposed to do and that system failures in drug development must be avoided because they might harm people. This paper shows – not the first time – how validation of systems – including systems that include open source software like R – can be done efficiently and should not prevent companies and users to incorporate R in their suite of data management, analysis, and reporting environment. More than ever before, interoperability between programs and tools enables users to pick the best tool for the job. The old paradigm, that 'if you only have a hammer, every problem is a nail' doesn't hold anymore,

REFERENCES

The R Foundation for Statistical Computing, The R Project for Statistical Computing, Accessed at https://www.r-project.org/

Federal Drug Administration (FDA), Glossary of Computer System Software Development Terminology, FDA, August 1995. Accessed at https://www.fda.gov/iceci/inspections/inspectionguides/ucm074875.htm

Federal Drug Administration (FDA), General Principles of Software Validation; Final Guidance for Industry and FDA Staff, January 2002. Accessed at https://www.fda.gov/downloads/MedicalDevices/.../ucm085371.pdf

Organisation for Economic Co-operation and Development (OECD), Draft Advisory Document 16 – The Application of GLP Principles to Computerised Systems, September 2014. Accessed at http://www.oecd.org/chemicalsafety/testing/Draft-OECD-GLP-Guidance-Document-computerised-systems.pdf

International Society for Pharmaceutical Engineering (ISPE), GAMP 5: A Risk-Based Approach to Compliant GxP Computerized Systems, February 2008. Available for purchase at https://www.ispe.org/publications/guidance-documents/gamp-5

The R Foundation for Statistical Computing, 2014. "R: Regulatory Compliance and Validation Issues. A Guidance Document for the Use of R in Regulated Clinical Trial Environments", Accessed at https://www.r-project.org/doc/R-FDA.pdf

Federal Drug Administration (FDA), 2015. "Statistical Software Clarifying Statement" Accessed at https://www.fda.gov/downloads/forindustry/datastandards/studydatastandards/ucm587506.pdf

Soukup, M., 2007. "Using R: Perspectives of a FDA Statistical RevieweR" Accessed at http://user2007.org/program/presentations/soukup.pdf

Brodsky, J. 2012, "Some Challenges of Using R in a Regulatory Environment" Accessed at http://blog.revolutionanalytics.com/downloads/FDA-Janice-Brodsky-UseR-2012.pdf

Kyun-Seop, B., Lee, J. E., 2017, "Development & Use of R Package for Non-compartmental Analysis", Accessed at http://www.phusewiki.org/docs/2017_CSS_US/PP02_Final.pdf

Wong, J., 2017, "Developing Standardized Clinical Review Tools Using Shiny in R", Accessed at http://www.phusewiki.org/docs/2017_CSS_US/PP34_Final.pdf

Taylor, A., Choi, D., 2018, "Utilizing Visualizations Developed in R Shiny for Exploratory Safety Analysis", Accessed at https://www.phuse.eu/css-2018-presentations-posters/p29.pdf

Rossini, A., James, D.A., 2007. “Open Source Statistical Software (OS3) in Pharma Development: A case study with R” Accessed at http://user2007.org/program/presentations/rossini.pdf

Mango Solutions, 2018, “ValidR”, Accessed at https://www.mango-solutions.com/data-science/products/valid-r/

Olszewski, A., 2016. “SAS and R Team in Clinical Research”, Accessed at http://www.kcrcro.com/uploads/attachments/news_pdf/EPC%20November%202016%20p18-21.pdf

Olszewski, A., 2017. “Another attempt to fight against decades of myths and FUD about using R in pharmaceutical industry. 13 resources worth reading”. Accessed at https://www.linkedin.com/groups/77616/77616-6339672314442977280

ACKNOWLEDGMENTS

The idea for this kind of white paper was originally born when VCA-Plus was founded, and we started to sell the concept of a validated cloud-based system supporting R and other scripting languages to the pharmaceutical industry. Recently, a post on LinkedIn by Adrian Olszewski (Olszewski, 2017) finally convinced us to execute on this idea, so he certainly deserves some of the credit.

CONTACT INFORMATION

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

Peter Schaefer
VCA-Plus, Inc.
pschaefer@vca-plus.com