PharmaSUG 2022 - Paper SI-057

External R Package Qualification Process in Regulated Environment

Jane Liao, Fansen Kong, and Yilong Zhang, Merck & Co., Inc., Kenilworth, NJ, USA

ABSTRACT

Traceability, reproducibility, and quality are critical components in clinical trial Analysis and Reporting (A&R). It is critical to ensure reproducibility by providing a centralized library of qualified R packages. The centralized library also needs to provide a traceable and consistent environment for members of an organization to use R for both exploratory and regulatory deliverables. R packages are open-source software developed by a community of contributors around the world. R benefits from this de-centralized cohort of developers who have contributed many R packages to the Comprehensive R Archive Network (CRAN) and other code repositories. CRAN and other code repositories empower accelerated innovation but come at a cost of inconsistent quality of R packages. The lack of guaranteed accuracy or standardized testing may be a concern of using R in a regulated environment for clinical development. A centralized R library governed by a thorough qualification process is necessary to ensure process compliance, quality, traceability, and reproducibility for formal A&R deliverables using R for an organization. In this paper, we propose an end-to-end process to qualify and install both internally and externally developed R packages in a regulated R environment. The proposed process aligns with the white paper from the R validation hub by defining a set of pre-specified criteria for the external R package qualification review process.

INTRODUCTION

With the growing trend of using multiple programming languages in clinical trial development, it is challenging to ensure compliance when open-source software is used. Although FDA provided a Statistical Software Clarifying Statement that "*FDA does not require use of any specific software for statistical analyses*" [1], sponsors need to ensure traceability, reproducibility, and quality of deliveries generated using open-source software.

Our journey started after our organization decided to explore using R for regulatory deliverables. In the past few years, we have seen great use cases across the pharmaceutical industry. For example, the R consortium submission working group completed the first open-source pilot submission to the FDA [2]. Zhang et.al released a book to discuss R for clinical study reports and publication [3]. Many use cases assume there is a Good Clinical Practices (GCP) compliant R environment to complete the development lifecycle.

Open-source software is different from commercial software that has a mature customer support model to help sponsors prepare and complete necessary steps to qualify the software. For example, R packages are open-source software developed by a community of contributors around the world. R benefits from this de-centralized cohort of developers who have contributed many R packages to the Comprehensive R Archive Network (CRAN) and other code repositories. CRAN and other code repositories empower accelerated innovation but come at a cost of the inconsistent quality of the R packages in the repository. The lack of guaranteed accuracy or standardized testing may be a concern for using R in a regulated environment for clinical development.

Open-source software is commonly distributed with no guarantee. The software is free, but with additional cost. As discussed in the R-FDA document [4], "*There is a significant obligation on the part of the end-user's organization to define, create, implement and enforce R installation, validation and utilization related Standard Operating Procedures (SOPs) within the end-user's environment.*" In other words, an organization needs to build a support team to review the test cases provided by the package developers and validate open-source software, which is an activity not commonly required when using commercial software. Sponsors should consider the benefits and costs before they adopt an open-source software and ensure alignment with their quality assurance (QA) department.

In this paper, we share our current experience tackling critical tasks to build a GCP-compliant R environment. Specifically, we propose an end-to-end process to qualify and install both internal and

external R packages in a regulated R environment. The proposed process aligns with the white paper from the R validation hub [5] by defining a set of pre-specified criteria for the external R package qualification and review process.

RISK-BASED PACKAGE QUALIFICATION STRATEGY

Our package qualification strategy relies on clear definitions. We reiterate what "validation" is and clearly define our goal. FDA defined validation as "*Establishing documented evidence which provides a high degree of assurance that a specific process will consistently produce a product meeting its predetermined specifications and quality attributes.*" More discussion can be found in Section 5.2 of the R-FDA document [4]. Based on the definition of validation, our goal (as an organization) is to create documentation that describes qualification details of R packages based on pre-specified criteria. Here an organization can be a group of people within a company or research institute who share common use cases for regulatory purposes.

The organization needs to define a proper software development lifecycle (SDLC) for internal R packages. For example, Zhu et.al [6] proposed an SDLC for internally developed R packages. The qualification of internally developed R packages is easier because the organization controls every step in the SDLC.

For external R packages, the organization serves as a customer to review the products (external R packages) provided by vendors (R package developers). As a customer, the organization has limited control over the source code of external R packages. The main responsibility of an organization is to review available materials to qualify the product for regulatory purposes following the standard operating procedure (SOP) of the organization. The organization may need to develop additional test cases and maintain validation code for external R packages; however, this is only part of the overall qualification efforts. This same process is followed when an organization purchases and uses other commercial software for regulatory purposes. Importantly, the organization is not a developer or maintainer of the external R packages, and the goal is not to write testing R code for every function from external R packages an organization uses.

We follow the R validation hub white paper [5] to apply a risk-based strategy to qualify both internal and external R packages based on the type of Analysis and Reporting deliverables. Table 1 provides one way to categorize the type of deliverables to align R package risk.

Type of Deliverables	Example	R Package Risk
External (electronic Common Technical Document (eCTD))	Clinical Study Report (CSR) and submission package Drug labeling Agency request	Low
External (non-eCTD) Internal (Outside Department)	Data monitoring committee Manuscript & publication (using clinical data) Internal committee review or presentation	Moderate or Low
Exploration/Within Department	Data exploration Data quality checks Exploratory analysis	Open, Moderate, or Low

Table 1. Examples of deliverables and respective risk categories of the R packages

We define three R package risk categories: low, moderate, and open.

- Low risk R packages are high-quality R packages that can be used to generate deliverables used for regulatory submission purposes.
- Moderate risk R packages are high-quality R packages that are suitable for most of the deliverables except eCTD.
- Open risk R packages are R packages with unknown quality ready for exploratory purposes.

To qualify external R packages in each risk category, we define five criteria.

- c1: Package is developed and maintained by a trusted vendor.
- c2: Package is user-facing with sufficient SDLC evidence equivalent to internal SDLC requirement.
- c3: Package is not user-facing and all packages dependent on this R package are qualified.
- c4: Package is user-facing with additional internal work to complete necessary steps following internal SDLC requirements.
- c5: Package maintained by a trusted person or organization.

For low risk, we propose using the first four criteria to qualify external R packages. For moderate risk, we propose using all criteria to qualify external R packages. As long as an R package meets one of the criteria, the R package can be qualified in the proper risk category. An example is provided in the appendix of an R package qualified with the c2 criterion. As the open risk category is for exploratory purposes, the goal is to allow users from the organization to use any R packages on CRAN or other repositories.

For the c1 criterion, the organization needs to review the SDLC document provided by vendors. With great support from the R community, several developer teams have provided SDLC documents. For examples,

- The R foundation: https://www.r-project.org/doc/R-FDA.pdf
- RStudio: https://resources.rstudio.com/assets/img/validation-tidy.pdf
- Stan: https://mc-stan.org/docs/sdlc.html

The organization should carefully review the SDLC documentation to determine if they can be classified as trusted vendors. One approach is to evaluate if the SDLC provided by a vendor covers minimal or equivalent requirements of an internal SDLC. R validation hub recognizes R foundation by providing it a trusted status which renders a collection of Base R and recommended packages “low risk” as a result [7]. Note that the presence of an SDLC document alone is insufficient to provide documented evidence as to why an organization trusts the software vendor/creator. As with a physical audit, it remains the responsibility of the organization to establish whether such SDLC is actually being followed. The list of trusted vendors is still being discussed in our organization.

For the criteria c2 and c3, it depends on the use cases of an organization to determine if an R package is user-facing. A user-facing package is a package that the user interacts with directly. It contrasts with the non-user-facing package that runs in the background as sub-routines on the back-end of another package. The key deliverable is a qualification report to summarize key information to support package qualification. In the appendix, we provide an example package qualification report. For criterion c4, the organization may need to put additional efforts into maintaining validation code if necessary. For criterion c5, the list of trusted persons or organizations should be properly reviewed and maintained by an organization.

PACKAGE QUALIFICATION

The process to qualify external R packages and build a regulated R environment can be cumbersome. Testing a statistical package can be challenging and time-consuming. We suggest that an organization create an internal R package to streamline and automate the process to reduce manual steps in qualifying external R packages. A high-level Global R Library update and package qualification workflow is summarized in Figure 1.

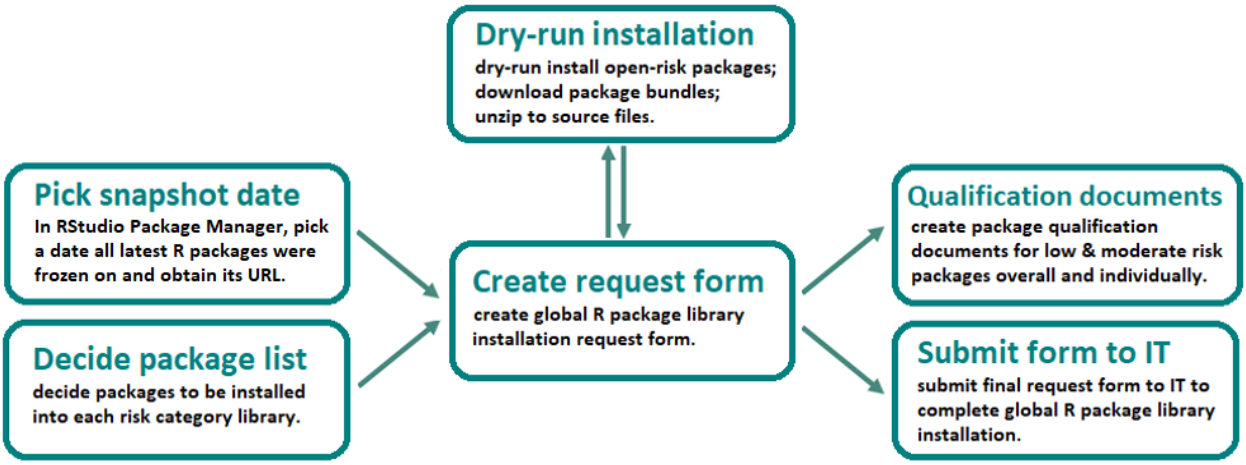

Figure 1. Global R library update and package qualification workflow diagram

GLOBAL R LIBRARIES

We follow a shared baseline strategy [8] recommended by RStudio to manage a reproducible R environment. Users can easily share and re-run work in this environment; the R package is available through a site-wide library installation and library updates are made on a scheduled basis.

The benefit of the shared baseline approach is that everyone uses the same installed packages. We build and update our libraries from a frozen repository (Figure 2). A frozen repository allows us to always get a consistent set of packages to ensure reproducibility.

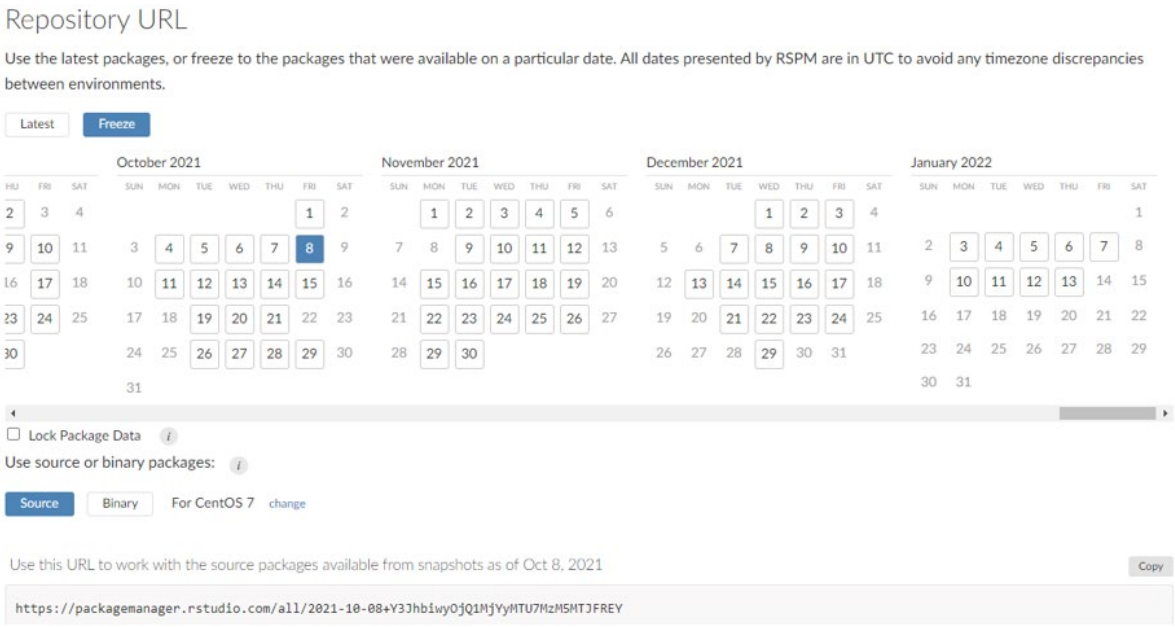

Figure 2. Frozen repository (Freeze on October 8th, 2021)

In our organization, we build a centralized library of R packages to provide a uniform environment to use R for both exploratory and regulatory work. The Global R Libraries are a set of directories containing installed R packages and their dependencies. There are three risk levels: low, moderate, and open. The global R libraries are nested and independent. All low-risk packages are included in the moderate-risk library and all moderate-risk packages are included in the open-risk library. The Global R Libraries are updated approximately every three months with a pre-defined snapshot date for the two latest RStudio versions available on the internal server (Figure 3). All previous releases of global R libraries are available on the server to ensure traceability and reproducibility.

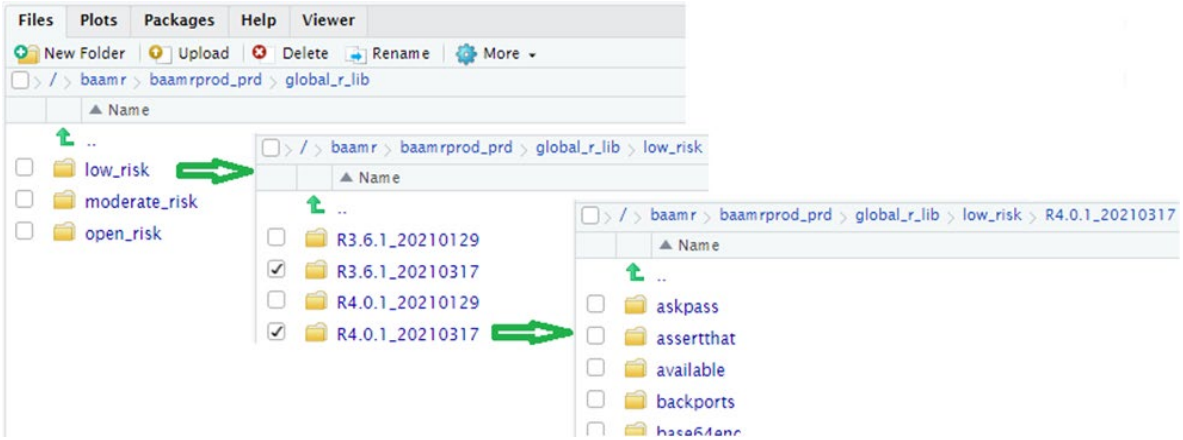

Figure 3. Global R library Folder Structure

GLOBAL R LIBRARIES UPDATE

To start the library update, a snapshot date needs to be selected first. RStudio Package Manager (RSPM) is the R package repository server we use to host source code to install packages in global R libraries. Other repository servers such as CRAN, etc. could be alternatives. A daily freeze is created on RSPM to lock the repository for a specific date. This date can be selected as the snapshot date to create new Global R Package Libraries. The unique URL associated with the snapshot repository is used to download and install R packages to the Global R Package Libraries.

Project Folder Structure

Once the snapshot date is chosen, we create a standard folder structure for package qualification [9]. Here is an example of the Global R Libraries installation view in RStudio with a snapshot date of October 8th, 2021, (Figure 4). The project folder “mkqualify-2021-10-08” is the location used to prepare the Global R Library update and package qualification. Within the folder, we have a sub-folder “mkqualify20211008” that contains an R package folder structure to save programs following the structure proposed by Wu et.al [9].

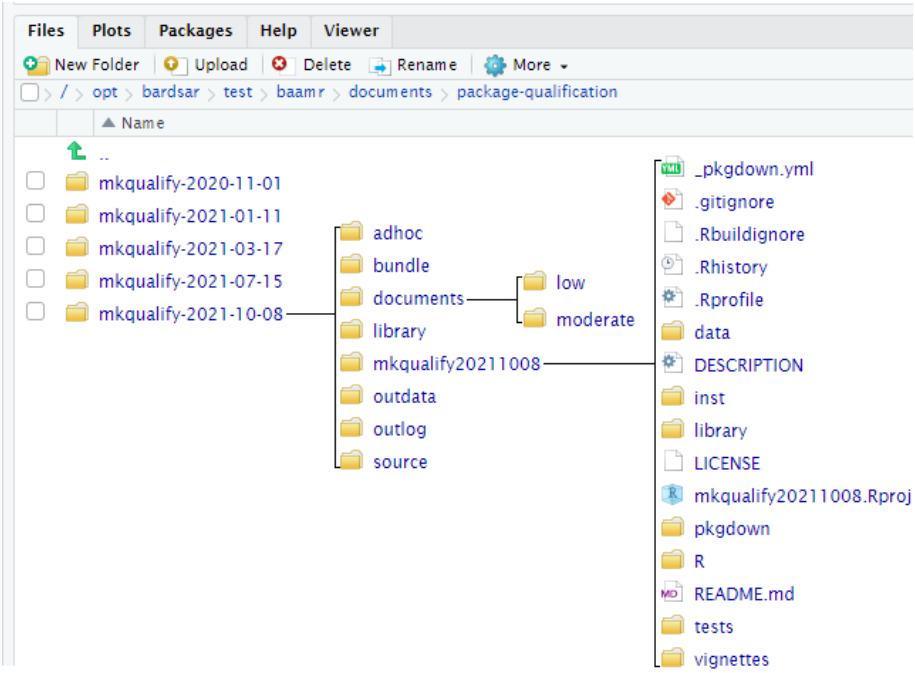

Figure 4. Project Folder Structure Example

Request Form

Request form is a csv file with a list of R packages to be installed (Table 2). Subject matter experts (SME) review and decide which packages to install and the associated risk level in the Global R Libraries. Below is sample code to obtain package dependencies, which is an essential part to create request form programmatically. The request form will be saved to the **documents** folder.

```
repo <- “https://packagemanager.rstudio.com/cran/2021-10-08”
pkg_avail <- utils::available.packages(contriburl = contrib.url(repo))
all_pkg   <- miniCRAN::pkgDep(pkg = c("broom", "cli", "crayon", "dbplyr"),
                              availPkgs = pkg_avail,
                              repos = repo,
                              type = "source",
                              suggests = FALSE,
                              includeBasePkgs = FALSE)
```

The request form contains the following information:

- Package: R package name
- Version: version of the R package
- Risk: package risk level (low, moderate, or open)
- R version: R version number.
- Snapshot date: date picked to freeze RSPM R package repository
- User-facing: Whether the package is to be used by the user directly. FALSE means the package is another user-facing package’s dependency

Package	Version	Risk	R version	Snapshot date	User-facing
available	1.0.4	low	4.0.1	10/8/2021	TRUE
boot	1.3-28	low	4.0.1	10/8/2021	TRUE
broom	0.7.9	low	4.0.1	10/8/2021	TRUE
class	7.3-19	low	4.0.1	10/8/2021	TRUE
cli	3.0.1	low	4.0.1	10/8/2021	TRUE

Table 2. Example of Global R Library Package Installation Request

PACKAGE DRY-RUN INSTALLATION

A Dry-run installation is conducted to examine whether packages can be installed successfully on the target server before a formal installation in the production area. It also helps to identify potential issues early. Only open-risk packages are needed for dry-run installations because all moderate-risk and low-risk packages are included in the open-risk level.

If a package cannot be installed on the server successfully, support will be needed from the Information Technology (IT) team to resolve the issue or remove the package from the request form after approval from the SME. R packages often depend on system libraries or other software external to R. These dependencies are not automatically installed. Installing system dependencies for a package can be challenging, especially when the instructions differ by the operating system. Missing system dependency is a typical reason that results in installation issues.

Below is sample code to install packages to the project **library** folder and download package bundles (.tar.gz files) to the **bundle** folder by assuming variables are properly defined.

```
tmp <- subset(request_form, risk == "open")
        for(i in 1:nrow(tmp)){
```

```
try(install.packages(tmp$package[i],
    lib = path_to_library,
    destdir = path_to_binary,
    repo = repo)}
```

Besides conducting a dry-run installation, we also unzip package bundles to extract package source files to the **source** folder so that we can run testing cases and create a code coverage report in the package qualification step later. Below is sample code to unzip package bundle files to the **source** folder.

```
lapply(list.files(path$bundle,
                  pattern = ".tar.gz",
                  full.names = TRUE),
                  untar, exdir = path$source)
```

All low-risk and moderate-risk packages to be included in Global R Package Libraries need to be qualified by pre-defined criteria so that they can be used for formal A&R deliverables. For each newly added external R package in the low or moderate risk category, SME should review the package and determine which criteria are used to qualify the packages.

Supplementary Information

Besides the important information of criteria a package meets, we also provide other supporting information in qualification documents, including but not limited to lock date, R version, package version, risk-level, user-facing status, source repository, dependency, author, maintainer, license, description, URL, bug reports, code coverage, and details of qualification criteria the package meets.

Sample code to obtain package information from each package's DESCRIPTION file:

```
pkg_path <- list.dirs(path$library, recursive = FALSE)
desc_path <- file.path(pkg_path, "DESCRIPTION")
pkg_desc <- lapply(desc_path[file.exists(desc_path)], function(x){
            try(as.data.frame(read.dcf(x),
                              stringsAsFactors = FALSE),
              silent = TRUE)})
```

Creation of Qualification Documents

We generate summary documentation for low-risk and moderate-risk R package qualification as well as individual documentation for each low and moderate risk R package qualification. They are saved in the **documents** folder. SME should review the qualification documents and confirm the accuracy before moving on to the next step.

- Overview document (Table 3): List of all packages in the respective risk category with the information about the R version, source repository, qualification criteria, and a table of Listing of Qualified Packages.

package	user_facing	origin	version	criteria
askpass	TRUE	RSPM	1.1	c1, c5
assertthat	TRUE	RSPM	0.2.1	c2, c5
available	TRUE	RSPM	1.0.4	c1, c5
backports	TRUE	RSPM	1.2.1	c1, c5
base64enc	FALSE	RSPM	0.1-3	c3, c5
bbmle	FALSE	RSPM	1.0.24	c3
bdsmatrix	FALSE	RSPM	1.3-4	c3

package	user_facing	origin	version	criteria
BH	FALSE	RSPM	1.75.0-0	c3, c5
bit	FALSE	RSPM	4.0.4	c3
bit64	FALSE	RSPM	4.0.5	c3

Table 3. Overview Document Example

- Individual document: Each low-risk and moderate-risk package in the snapshot library should have an individual qualification document created. This document includes an overview, package information, package dependency, and qualification details such as criteria qualified for the package and details about the criteria. Refer to the appendix for an example of an individual qualification document for the R package: selectr.

REQUEST FORMAL INSTALLATION

After SME review to confirm that qualification documents are accurately created, the final request form is used for formal Global R Package installation. Notifications will be sent to users when new Global R libraries are successfully installed. Qualification documents will be saved to the production area and the preparation work for this library update snapshot will be frozen.

CONCLUSION

R packages are open-source software. As an organization, we benefit from using R packages to accelerate innovation through the advanced analytical methods they offer. While open-source software significantly enhances our toolbox to support A&R deliverables, they come with the added cost of qualification to meet the regulatory guidance. The external R package qualification process ensures high quality and compliance in a regulated environment. The R validation hub is also developing an open-source R package “riskmetric” (https://github.com/pharmaR/riskmetric) to help an organization collect metrics that can be useful for the R package qualification documentation.

REFERENCES

[1] U.S. Food & Drug Administration “Statistical Software Clarifying Statement” May 6, 2015. https://www.fda.gov/files/about%20fda/published/Statistical-Software-Clarifying-Statement-PDF.pdf

[2] R consortium submission working group “Successful R-based Test Package Submitted to FDA” December 8, 2021. https://www.r-consortium.org/blog/2021/12/08/successful-r-based-test-package-submitted-to-fda

[3] Yilong Zhang, Nan Xiao, Keaven Anderson “R for clinical study reports and submission” Available at https://r4csr.org

[4] The R Foundation for Statistical Computing c/o Institute for Statistics and Mathematics “R: Regulatory Compliance and Validation Issues, A Guidance Document for the Use of R in Regulated Clinical Trial Environments” October 18, 2021. https://www.r-project.org/doc/R-FDA.pdf.

[5] Andy Nicholls, Paulo R. Bargo, John Sims on behalf of the R Validation Hub "A risk-based approach for assessing r package accuracy within a validated infrastructure" January 23, 2020. Available at https://www.pharmar.org/white-paper/.

[6] Yalin Zhu, Rinki Jajoo, Clare Bai, Sarad Nepal, Daniel Woodie, Keaven Anderson, Yilong Zhang “R Package Oriented Software Development Life Cycle in Regulated Clinical Trial Environments” 2020. https://www.lexjansen.com/phuse-us/2020/tt/TT12.pdf

[7] The R validation hub blog on trusted resources: https://preview--condescending-lewin-a2fc51.netlify.app/blog/2022/01/07/2021-12-10-trusted-resources/

[8] RStudio “Shared Baselines”. https://environments.rstudio.com/shared

[9] Peikun Wu, Uday Preetham Palukuru, Yiwen Luo, Sarad Nepal, Yilong Zhang 'Analysis and Reporting in Regulated Clinical Trial Environment using R'
https://www.pharmasug.org/proceedings/2021/AD/PharmaSUG-2021-AD-079.pdf

APPENDIX

Below is an example of the individual qualification document for R package: selectr

PACKAGE QUALIFICATION – "SELECTR"

QUALIFICATION OVERVIEW

The purpose of this document is to demonstrate that selectr when used in a qualified fashion, can support the appropriate regulatory requirements for validated systems, thus ensuring that resulting electronic records are "trustworthy, reliable and generally equivalent to paper records."

Package	selectr
Risk level	low
Qualification date	2021-10-19
Qualification criteria	c2

PACKAGE INFORMATION

package	selectr
version	0.4-2
author	Simon Potter [aut, trl, cre], Simon Sapin [aut], Ian Bicking [aut]
maintainer	Simon Potter <simon@sjp.co.nz>;
license	BSD_3_clause + file LICENCE
description	Translates a CSS3 selector into an equivalent XPath expression. This allows us to use CSS selectors when working with the XML package as it can only evaluate XPath expressions. Also provided are convenience functions useful for using CSS selectors on XML nodes. This package is a port of the Python package 'cssselect' (<https://cssselect.readthedocs.io/>;).
url	https://sjp.co.nz/projects/selectr
bugreports	https://github.com/sjp/selectr/issues

PACKAGE DEPENDENCY

package	version	user_facing	criteria
glue	1.4.2	FALSE	c1
R6	2.5.1	FALSE	c1

package	version	user_facing	criteria
selectr	0.4-2	FALSE	c2
stringi	1.7.5	FALSE	c3
magrittr	2.0.1	TRUE	c1
stringr	1.4.0	TRUE	c1

We listed all R package dependencies including the current package, except base packages below. Because they are installed with R and qualified based on the document: R: Regulatory Compliance and Validation Issues A Guidance Document for the Use of R in Regulated Clinical Trial Environments.

QUALIFICATION DETAILS

CRITERION C2

Criterion c2: There is sufficient evidence of publicly available software development lifecycle information, including authors, source code, testing, release notes, and user guides.
To qualify selectr, we reviewed and confirmed the R package selectr follows a proper software development lifecycle.

- Each exported (user-facing) function contains a documentation.
- The released version has a unique version number on CRAN.
- Proper testing has been provided with source code. The code coverage is 98%.
- The R package passed a series compliance check through CRAN.
- The R package has a maintainer, Simon Potter simon@sjp.co.nz
- The R package has a bug report approach using https://github.com/sjp/selectr/issues
- The R package can be properly built and installed in the system.
- Source code archive files (“tarballs”) are made available via the CRAN mirror infrastructure.
- All current and historical released versions of this R package are available from the main CRAN server (http://cran.r-project.org/src/base/) and its worldwide mirrors (http://cran.r-project.org/mirrors.html).

ACKNOWLEDGMENTS

The authors would like to thank management teams from Merck & Co., Inc., Kenilworth, NJ, USA, for their advice on this paper/presentation and want to thank Suhas Ramesh Sanjee from Merck & Co., Inc., Kenilworth, NJ, USA, for valuable inputs on the paper.

CONTACT INFORMATION

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

Jane Liao
Merck & Co., Inc., Kenilworth, NJ, USA
Jane.liu2@merck.com

Fansen Kong
Merck & Co., Inc., Kenilworth, NJ, USA
fansen.kong@merck.com

Yilong Zhang, Ph.D.
Merck & Co., Inc., Kenilworth, NJ, USA
yilong.zhang@merck.com