Paper OS12

Handle with Care: Study-Specific Assessment of R Package Risk

Shannon Grant, Statistical Center for HIV/AIDS Research and Prevention (SCHARP) at Fred Hutch Cancer Center, Seattle, USA
Blazej (Moni) Neradilek, Statistical Center for HIV/AIDS Research and Prevention (SCHARP) at Fred Hutch Cancer Center, Seattle, USA

ABSTRACT

R packages contain statistical goodies but sometimes also pack explosive accuracy and reproducibility challenges. We developed a practical R package risk assessment methodology that evaluates package characteristics and study-specific needs for high-profile statistical computing in a validated R environment at a nonprofit, quasi-academic medical research institution.

We begin by reviewing challenges of validating open-source software, and risk management approaches from community and industry leaders. We summarize internal discussions rooted in the R Validation Hub White Paper principles, further refining our scope to specific tasks (randomization lists and protocol endpoint analysis) in individual high-profile studies. The assessment is carried out for each study by statistical leads, applies both institutional and study-specific rules, and is documented in an R usage plan. The plan also describes package-specific risk mitigation and measures to maintain reproducibility. Further, we discuss process challenges, including the lead's literacy in interpreting R package characteristics (e.g. `riskmetric` package outputs) in study context, assessment of dependencies, and changing package versions.

INTRODUCTION

BACKGROUND

Software validation is a critical topic in clinical trials work. ICH E9 requires data management and statistical software to be "reliable", with documented testing [1], while validation of computer systems used in clinical trials is required by ICH E6(R3). Validation should include system requirements and specifications, testing plans, and test results, and "should be based on a risk assessment that considers the intended use of the system" [2].

Computer systems validation typically includes Installation, Operational, and Performance Qualification testing (IQ, OQ, and PQ, respectively). IQ and OQ ensure that the software has been installed correctly and is operating as expected, while PQ confirms that user requirements are satisfied. IQ, OQ, and PQ all include testing software behavior against pre-specified tests. In our field, validation of a particular piece of software takes place in a statistical computing environment (SCE) that has itself been validated.

Open-source software such as R presents unique validation challenges. The R Foundation ensures that base R and recommended packages follow a software development lifecycle (SDLC) including code maintenance, release control, testing, and validation against known data and results prior to release, all by qualified people in the R Core Team [3]. To be released on CRAN, other community-contributed packages must pass technical checks to be sure the code examples run, package tests pass, and the package is compatible with other CRAN packages. However, this doesn't mean that results returned by the package are correct. For example, a statistical method could be misunderstood and implemented in ways which are conceptually incorrect, but run smoothly and consistently.

The lengths to which a team goes to formally assess utilized R packages depend on their appetite for risk. To paraphrase Colin S. Gillespie [4], the risk appetite continuum ranges from unconcerned (students) to those determined to thoroughly assess and handle risk (pharma companies), with users who want results to be right but aren't bothered by documentation requirements falling somewhere in the middle. Risk appetite is influenced by both resource levels and strictness of the field in which users work. Pharma companies operate in a highly regulated space, but medical researchers at academic or quasi-academic institutions may have different concerns. Ultimately a range of risks is acceptable within most organizations, with high-impact outputs requiring formal quality procedures, and more flexible requirements for exploratory, less-regulated projects [4].

COMMUNITY STATUS

Validation of R packages is an active topic in the open-source community. The R Validation Hub states that system validation in regulated biomedical research should include accuracy, reproducibility, and traceability [5]. Further, the R Validation Hub suggests that a risk assessment framework for R packages should include these four dimensions.

1. Purpose: will the package use be statistical or non-statistical? Statistical packages are riskier because algorithms are more complex, and errors more difficult to detect.
2. Maintenance good practice: to what extent do the package maintainers follow an SDLC, provide documentation, and make their practices transparent?
3. Community usage: is a package widely used, and incorporated into other packages? Mature packages which are widely used have essentially undergone scattershot informal testing.
4. Testing: does the package include unit tests, and how completely do tests cover functions?

The `riskmetric` package developed by the R validation hub is a useful tool for gathering information about R packages [6]. Version v0.2.5 (July 23, 2025) returns 19 package characteristics, grouped into the four dimensions described above. The package includes functions to assign a numeric score to each metric, and assign a numeric, custom-weighted package-level risk score. Metrics ranged from simple (presence of a NEWS file) to complex (unit test coverage).

Review of three case studies shows that community and industry leaders have taken different approaches to analyzing and mitigating R package risk [7a-7d]. The four dimensions proposed by the R Validation Hub are included, along with usage of `riskmetric`, though different metrics were chosen to represent each dimension. A common approach is to combine an automated tool collecting and presenting package characteristics with expert review and potentially justification of any gaps or exceptions. Some groups follow a flowchart to assign risk level, while others present package characteristics for overall judgement. Action taken based on risk findings varied, from mandating extra testing based on risk level to prohibiting use of riskier packages for higher-impact deliverables. In some cases, an R governance team was responsible for testing, while in others the project team was responsible.

R VALIDATION AT SCHARP

OUR ENVIRONMENT

SCHARP at Fred Hutch is a nonprofit, quasi-academic medical research institution, working primarily in HIV and other infectious diseases. We contribute to clinical trial design, data management, and analysis, collaborating with the HIV Vaccine Trials Network (HVTN), HIV Prevention Trials Network (HPTN), and other research groups.

Our SCE is managed by a small IT group and contains Unix computing servers with central R installations, some of which are validated. The R server includes a validated package library, but there are no restrictions on users installing and using packages from any source. The validated package library includes select packages from CRAN, Bioconductor, and public GitHub.com repositories. User libraries source packages from these and other repositories (including internally-developed packages from private GitHub.com repositories).

SCHARP biostatisticians and statistical programmers are a mix of SAS and R users, and choice of software is left to each trial's statistical team, in consultation with our principal investigators.

Our team focuses on HIV vaccine research. While there are many licensed products for HIV prevention, an effective vaccine does not yet exist. The HVTN research portfolio contains mostly Phase 1 vaccine trials, with a few active Phase 2 trials. Past work included efficacy trials, and future Phase 3 trials are planned. Though SCHARP has contributed to submissions, and we currently use R for licensure-track products, all submission-related work was done entirely in SAS. The HIV vaccine statistical community strongly prefers R, and continued use of open-source software usage in high-profile trials necessitates an increase in rigor to prepare for closer scrutiny. We developed the R usage plan to serve as a central place to describe how we intend to use R for an individual trial; no such document existed in our group before now.

STREAMLINED R VALIDATION

Our team of R users (biostatisticians and statistical programmers) is small, and we must take practical, risk-based approaches to verification and validation. After extensive discussion and iteration, we decided on a minimalist approach to R PQ with the only test case being if R CMD checks [8] passed or only failed in areas of no current impact. A package that did not pass R CMD checks could still be included in the validated library if, in the PQ team's best judgement, the reason for test failure would not interfere with how we expect to use the package. The specific failures are noted in PQ documentation for potential future considerations (in case of future use and impact).

This limited PQ is justified by later study-specific planning and preparation. Statistical study leads have the best understanding of study conditions, and the R packages and functions they will use, and as such are responsible for risk assessment and mitigation. These details will be recorded in the R usage plan.

R USAGE PLAN

CHOICE OF RISK SCORES AND METRICS

We considered adopting a risk score framework but ultimately decided on a qualitative approach. A consensus-based composite risk index that summarizes across multiple domains is challenging to develop and we did not have sufficient data and expertise (akin to latent variable modeling or psychometrics) to properly justify it as a general process applicable to all our studies.

First, we would ideally have error data that we can train and validate the scores on. Being new to risk assessment, we do not currently have such data, and will not soon have enough data to power reliable model training.

Second, creating an overall score comes with judgement calls, such as several alternative rationales and methodologies for combining across multiple domains and choosing thresholds to categorize into risk categories.

Third, given the lack of data and clear methodology, fully automated risk rules would discourage our team members from developing their own general understanding of R package characteristics and risks and applying them to different study contexts (study as an effect modifier). For example, with too general rules that ignore context, our users may fail to understand that sometimes a package could be rated risky due to characteristics which are irrelevant to a particular study.

Ultimately the process of reviewing package characteristics to make a qualitative decision serves our goal of improving team literacy in understanding and assessing R package risk. Collectively, we are still learning about package characteristics, and our chosen approach is designed to encourage further learning. In the future, as we gain experience both as a team and as part of the R community, a quantitative approach may be more compelling.

R USAGE PLAN OVERVIEW

An R usage plan is required only for high-profile, high-risk trials, an internal SCHARP designation which generally translates to collaboration with an industry developer, a licensure-track product, or a Phase 2 or higher trial. Packages used for randomization lists and primary endpoint analyses or data manipulation must be included, while package dependencies and those used for secondary endpoint analyses are currently recommended but optional.

Our usage, impact, and risk considerations are structured like those described in the R Validation Hub white paper, adapted to study needs. For each package, our template R usage plan presents select package characteristics collected using `riskmetric` and custom code, along with study-specific usage, purpose, and validation status. A semi-automated initial risk assessment quickly off-ramps packages with low impact or low risk (based on our institutional criteria) and identifies a smaller subset of packages needing further, more time-consuming study-specific risk assessment. Prompts and examples lead staff through conducting a qualitative study-specific risk assessment and developing a plan to mitigate risk, if needed. Finally, the R usage plan ends with a description of steps the statistical team will take to maintain stability and continuity of R usage over the life of the trial.

PACKAGE CHARACTERISTICS

The template R usage plan presents the following package characteristics for externally-developed packages:

- Purpose
- Passing R CMD checks
- Trusted source (base R, recommended, Tidyverse)
- Number of reverse dependencies
- Number of annual downloads
- Bug report URL exists
- Help files present

Of these, the first four are required by the R usage plan, while the remaining three may be dropped or replaced by other characteristics at the discretion of the statistical team.

These characteristics were chosen as straightforward metrics of the four components of a risk assessment framework as described in the R Validation Hub white paper [4]. Passing R CMD checks is a basic quality measure, while existence of a bug report URL and completeness of help files are indicators of maintenance good practice. Community usage is represented by number of reverse dependencies and number of annual downloads. Package development by a trusted source encompasses good maintenance practices, community usage, and testing, as these packages are widely used and set the standard for package development practices and testing. The section on study-specific risk assessment discusses potential impacts of gaps in these characteristics.

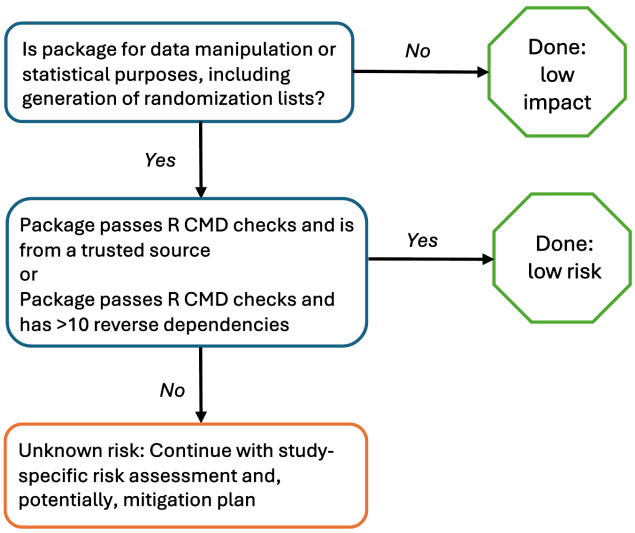
Package purpose is study-specific, and the assessment of category is limited only to functions that perform activities within the scope of the R usage plan. The assessment should also include other functions in the same package that are either directly or recursively called by these functions. Other functions in the package may be ignored. Purpose categorization does not account for dependencies on other packages. Instead, if dependencies are included in the R usage plan, purpose is documented separately for each assessed dependency package.

Possible purpose categories are statistical analysis (high impact), data manipulation (medium impact) and other non-statistical purpose (low impact). If a package is assessed into multiple categories, the package is assigned the category with the highest impact. For example, a package with functions for both data manipulation and statistical analysis is assessed as a statistical package. Note that the assessment depends on R usage specific to each study, so category for the same package can vary across studies or even for the same study over time (if the study's R usage changes).

INITIAL RISK ASSESSMENT

The initial risk assessment is designed to quickly determine which packages can be accepted without further consideration, freeing the risk assessor to devote more effort to reviewing packages with potentially concerning characteristics. The same rules are applied across the organization, though statistical teams are allowed to make the rules stricter. In this assessment, packages are sorted into mutually exclusive categories of low impact, low risk, or unknown risk. See Figure 1 for a flowchart.

Figure 1: Initial risk assessment flowchart showing criteria applied to each package.



First, impact is checked. Packages are *low impact* if they are not used for data manipulation, statistical analysis, or generating randomization lists. For example, packages such as `ggplot2` and `knitr` which create visualizations and output results, but do not generate results, are low impact. Our definition of low impact is a variation of statistical versus non-statistical purpose described in the R Validation Hub white paper. We decided not to label non-statistical data manipulation packages as low impact because errors in analysis datasets can be catastrophic; an analysis is only as accurate as the underlying dataset.

Packages which are not low impact go on to the risk check. Packages are *low risk* if they pass R CMD checks and are developed by a trusted source, or pass R CMD checks and have high community uptake. We define trusted source packages as base R, CRAN-recommended packages, and the Tidyverse. Developers of these packages are leaders in maintenance good practice, including following an SDLC with testing. High community uptake refers to those packages widely used by the developer community, with more than 10 reverse dependencies. We consider

high developer uptake to be a better measure of community usage than number of downloads, as developers are more closely engaged with the functions and likely to spot problems than the typical user. We arrived at 10 as the threshold for the number reverse dependencies by reviewing characteristics of statistical packages in our validated library; admittedly 10 is a somewhat arbitrary number, but it neatly aligned with packages we felt to be of high quality, based on our experience. Lastly, R CMD checks are a foundational measure of meeting package norms.

Packages which are not low impact or low risk are said to have *unknown risk*.

The above criteria are applied to externally-developed publicly-available packages. Internally-developed packages which do not have low-impact purpose are assessed as unknown risk.

The initial risk assessment is implemented programmatically in a template Rmarkdown report which runs R CMD checks and uses `riskmetric` to return package characteristics. Purpose and trusted source are added into inputs manually. Packages which are low impact or low risk need not be considered further, while packages of unknown risk are considered in the study-specific assessment. Refer to Figure 2 for an example of the initial risk assessment table.

Figure 2: Template risk assessment table showing select package characteristics and initial risk assessment.

Table 3: Initial risk assessment for external packages. Sorted by initial impact/risk, purpose and package name. *(continued)*

Package			Deliverable and objectives		Low impact criteria	Low risk criteria			Potential study-specific risk assessment characteristics, if applicable			
Name	Version	Validated	Deliverable(s)	Objective(s)	Purpose	CRAN/Bioc check error	Source	Reverse dependencies	Annual downloads (in 1000s)	Bug report URL exists	Help files present (%)	Initial impact/risk assessment
purrr	1.0.2	Yes	PT report, pdata	P2, S1	Data manip.	No	Tidyverse	2099	11492	Yes	100	Low risk
survival	3.8.3	Yes	PT report	P2, S1	Statistical	No	Recommended	1086	1549	No	100	Low risk
Exact	3.3	No	PT report	P2, S1	Statistical	No	Other	5	525	No	100	**Unknown risk**
MIMOSA	1.29.1	Yes	pdata	S1	Statistical	No	Other	0	<1	No	67	**Unknown risk**
PropCIs	0.3.0	Yes	PT report	P2, S1	Statistical	No	Other	3	14	Yes	100	**Unknown risk**
randomizeR	3.0.2	Yes	Randomization list	NA	Statistical	No	Other	1	6	No	78	**Unknown risk**

P2 means Primary objective 2, S1 Secondary objective 1. *Other indicates packages with other non-statistical, non-data manipulation purposes.

STUDY-SPECIFIC RISK ASSESSMENT

Study-specific risk assessment is flexible and tailored to the needs of the trial. As in the initial risk assessment, the assessed scope for each package includes only relevant functions, and any other functions in the same package that are either directly or recursively called by the used functions.

During the study-specific risk assessment, the risk assessor must decide if the final risk level is low, high, or unknown. Packages which are not low risk go on to risk mitigation planning. Risk level definitions:

- A package is justifiably *low risk* if the assessor expects it to be reliable, so that problems with the relevant functions are rare or easy to detect and address.
- A package may be *high risk* if the assessor is not confident in its reliability or suitability for the study. There may be known or expected issues with relevant functions or documentation, function operating characteristics may not have been assessed by the statistical team so it's unclear if the package will meet the needs of the study, or the version history may indicate inconsistent maintenance.
- A package is of *unknown risk* if it is not clearly low or high risk.

Characteristics to consider for study-specific risk assessment

While our process does not mandate specific use of package characteristics in study-specific risk assessment, we provide the following recommendations or considerations to help statistical teams that are new to package risk assessment.

Statistical teams should consider the *likelihood of encountering problems* when using the package. Less-used packages such as those with version < 1.0.0 (typically development versions), fewer downloads, and fewer reverse dependencies have higher risk of having problems.

Ease of troubleshooting should also be considered. Missing help files (for used functions) mean the user doesn't know well what a function does and can often indicate high risk. In contrast, availability of a relevant

vignette may increase the assessor's confidence in a function. A bug reporting URL and recent bugs being addressed can indicate low risk, as together they facilitate identification of errors by users and subsequent fixes by developers. Bug reporting used to be less common for older packages. For these packages, the lack of bug reporting URL can be outweighed by their long-term use by many users and strong development over time. Statistical teams are encouraged to check the number of downloads, number of reverse dependencies, the NEWS file and number of releases to assess developer activity on issues and package improvement.

Suitability of the package for the *study context* should also be assessed. The statistical team can review the literature or test statistical functions for their operating characteristics under relevant scenarios, such as different rates of missingness or censoring, ties, or a range of event rates. In some studies, such testing may have been done during the trial design phase, and can be documented in the risk assessment.

If units tests are available in the package source, the risk assessor can check *unit test coverage*. This process is potentially time-consuming as it requires forking the source, running it on our system, and interpreting the results. If coverage is not 100%, the risk assessor should consider whether the available unit tests cover relevant functions we plan to use, and are adequate. This effort could be worthwhile if risk cannot be determined from other characteristics and the package impact warrants a more thorough risk assessment.

Writing the study-specific risk assessment

The statistical team must make their own study-specific decisions, and document their reasoning. The goal is to write a thorough qualitative composite assessment, considering the balance of package strengths and weaknesses. There are no prescribed criteria, but the template Rmarkdown report includes the number of annual downloads, existence of a bug report URL, and help file coverage (see Figure 2). The template code collects other `riskmetric` characteristics, and study-specific R usage plans may report these as well. It is possible that no single characteristic may well support the final risk assessment, but the total information from several characteristics may more robustly point the assessor to a final risk determination. While the rationale for the determination may be subjective and dependent on the assessor's risk tolerance, or be only vaguely related to computational risks such as output errors, the assessor should document the rationale clearly in the R usage plan.

To help our team learn about package risk assessment, the template Rmarkdown report and process documentation provide recommendations and examples for packages we regularly use, many of which mirror considerations in the R Validation Hub white paper and community approaches. We provide this example for the package `randomizeR`:

> The assessor for study A is evaluating the package `randomizeR` for producing randomization lists. The assessor notes a small number of reverse dependencies and downloads as well as incomplete help file documentation as weaknesses. They also note non-developmental version, the long history of the package, a robust JSS paper (Uschner 2018), and available help documentation for functions needed for the study as strengths. For context when determining the availability of help documentation, the assessor also notes that while the `riskmetric` export_help metric < 1 indicated some missing help files, help files were in fact missing only for internal and hidden objects and all documentation for user-facing functions needed for the study was available. During a discussion with the lead statistician about the pros and cons, they consider low versus uncertain risk as two competing possibilities. Although of potentially low risk, to err on the side of caution, they decide to categorize the package as uncertain risk. (Note that a more risk-tolerant study team may have gone with low risk!)

RISK MITIGATION PLAN

Like the prior risk assessment step, risk mitigation is study-specific and limited to only relevant functions. Packages which are not both low impact and low risk per the study-specific risk assessment require a mitigation plan. When planning how to handle risk, the assessor should consider package characteristics, impact of errors on the study, the study team's risk tolerance, and availability of additional resources for mitigation.

Some possible mitigation steps include:

- For statistical packages, compare to known results or results generated using other statistical software.
- Use an alternative function, R package, or other software system with lower risk, or identify an alternative to use if problems arise.
- Plan to verify results by independent programming using a different R package, SAS, or other statistical software.
- Write custom tests that show high impact function(s) work as expected.
- No mitigation. In this case the R usage plan must acknowledge the willingness to tolerate the risk.

We provide this example of a risk mitigation plan for the package `randomizeR`:

The risk assessor concludes no mitigation is needed as our standard practices include a second statistician checking the randomization list to verify that the number of treatment groups, strata, blocks, block size(s), and number of treatments within a block are as specified. If any unresolvable errors arise, the study team can use SAS PROC PLAN to generate the randomization list.

REPRODUCIBILITY

This section of the R usage plan describes how the statistical team ensures reproducibility of outputs generated by assessed packages throughout the life of the study. We mandate the following internal reproducibility measures.

Package libraries

For packages and study objectives in scope, the team only loads packages from static R libraries that contain package versions assessed by the currently effective version of the R usage plan. If this package version is available in the validated R library, the team must load it from there, even if the same version is available in another library. If this package version is not available in the validated library, the team will use another package version control approach to load the appropriate version. For example, the team can create a study-specific library in the study folder and/or use the `renv` package manager. The team should not load packages from personal libraries as these frequently differ in version across users and over time. Any deviation from these requirements must be documented and justified in the R usage plan, including how reproducibility risks will be managed.

Package versions

When new versions of included packages become available and are needed, the assessor evaluates risk of reproducibility failure and, if needed, revises the reproducibility plan. The assessor should first review release notes and news for the package to assess the extent of recent updates, and potential impact on their protocol. If the risk assessment or mitigation changes, the risk assessor updates the plan.

If the assessor finds risks to reproducibility such as new errors or different structures or results, they describe the risks and corresponding measures to maintain workflow and result stability, such as testing new versions before using them in production. They also describe any changes to the package management approach. For example, if the statistical team was originally able to use only packages from the validated library but they decide to up-version a package, they could create a new static study-specific library to load their new version from.

Widely used, well-maintained packages (such as base R, CRAN-recommended and Tidyverse packages) have less risk of reproducibility failure over time, as new versions of these packages typically maintain backwards-compatibility and are updated regularly with new releases of R. However, this is not guaranteed, and careful review of changes as described above is still needed. In contrast, new versions of other packages or changes to their R and non-R dependencies have a higher risk of being destabilizing, and should be scrutinized. If code needs to transition across multiple versions, developmental packages with version <1.0 may be especially prone to reproducibility failures.

Limitations of the current approach to reproducibility

The scope of our reproducibility planning is internal, and the R usage plan does not discuss how to ensure individuals outside our group can reproduce results. External reproducibility is an area for future work.

Recall that our SCE has no guardrails to prevent users from installing and using packages outside the validated library; thus we currently lack an enforcement mechanism for ensuring the approach described in the R usage plan is followed consistently by study team members. This is also a topic for future development.

EFFICIENCIES AND TEAM LEARNING

We, the authors, are jointly responsible for the development of the R usage plan, including the process and training materials. The R usage plan process is governed by a standard operating procedure, which includes many examples of how to conduct a study-specific risk assessment and consider plans for mitigation.

We designed an R usage plan template to provide structure and minimize repetitive effort. The plan template is managed and distributed through a github.com repository, including extensive instructions and examples. Users update configuration files with any additional packages for their study, and generate the first version of the report by knitting an Rmarkdown document. The first run of the plan contains an automated table which implements the standard initial risk assessment, and sample study-specific risk assessment and mitigation plans for our most

commonly used packages. Users edit the samples as needed for their risk preferences and study context, or use as models for other packages.

The R usage plan process is integrated with R PQ. The template R usage plan reuses code and, for validated packages, incorporates package characteristics collected by the R PQ team. Preliminary package purpose (not accounting for study context) was determined by the R PQ team.

CONCLUSION

This is a new process for our organization. We plan to evaluate, re-assess and modify the process, template, and training materials in the future.

Our intent is to force assessors and stakeholders to think hard about all relevant study- and package-specific risks. With most of the team new to package risk assessment, exposure to and implementation of this approach will lead to wider institutional knowledge of the breadth of risks and greater literacy in package risk assessment and inform further development of the process. Guidance and training are provided by a template R usage plan repository and detailed work instruction.

Relative to industry and community case studies we reviewed, our assessment is more customized to study context and requires greater assessor effort and judgement. This may seem like a step backwards when compared to efforts in other organizations to automate this process as much as possible. However, in our context, this approach is manageable as the number of studies is limited. We felt it is important for all staff to build deep awareness of R package risks, and handle packages with extra care!

REFERENCES

1. International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use. *Good Statistical Principles for Clinical Trials* (ICH E9, 1998). https://database.ich.org/sites/default/files/E9_Guideline.pdf, accessed 2 January 2026.
2. International Council for Harmonisation of Technical Requirements for Pharmaceuticals for Human Use. *Guideline for Good Clinical Practice* (ICH E6(R3), 2025). https://database.ich.org/sites/default/files/ICH_E6%28R3%29_Step4_FinalGuideline_2025_0106.pdf, accessed 2 January 2026.
3. The R Foundation for Statistical Computing (2021). *R: Regulatory Compliance and Validation Issues A Guidance Document for the Use of R in Regulated Clinical Trial Environments*. https://www.r-project.org/doc/R-FDA.pdf, accessed 2 February 2025.
4. Gillespie CS. *Can I trust that package?* (posit::conf 2025). https://www.youtube.com/watch?v=g6QU16jOt0g
5. R Validation Hub, Nicholls A, Bargo PR, Sims J. *A Risk-based Approach for Assessing R package Accuracy within a Validated Infrastructure* (2020). https://pharmar.org/white-paper/, accessed 2 February 2025.
6. R Validation Hub, Kelkhoff D, Gotti M, Miller E, K K, Zhang Y, Milliman E, Manitz J (2026). *riskmetric: Risk Metrics to Evaluating R Packages*. R package version 0.2.6, https://CRAN.R-project.org/package=riskmetric.
7. Case studies from collection at https://pharmar.org/categories/case-studies/.
    a. Roche R Package Validation Team. *Automated R package validation*. https://github.com/pharmaR/case_studies/blob/fa5e4c82098266fe7516adaa8671a64c9796a0e3/Roche_Rpkg_validation_2022.pdf
    b. Stachniuk A, Wallet P, Fidler M, Ochsenbein D, Schmidt S, Illis W. Case study: *R Package Risk Assessment at Novartis* (2022). https://github.com/pharmaR/case_studies/blob/fa5e4c82098266fe7516adaa8671a64c9796a0e3/Novartis_Rpkg_validation_20220509.pdf
    c. Palukuru UP, Bernecki P, Liao J, Zhang Y. *External R Package Qualification Implementation at Merck: Case study using GGally* (2022). https://github.com/pharmaR/case_studies/blob/fa5e4c82098266fe7516adaa8671a64c9796a0e3/External%20R%20Package%20Qualification%20Implementation%20at%20Merck.pdf
    d. Manitz J, Pinkert S, Gregory M, Beckers F. *Case Study: Risk Assessment of R Packages and Merck KGaA/EMD Serono* (2022). https://github.com/pharmaR/case_studies/blob/fa5e4c82098266fe7516adaa8671a64c9796a0e3/Merck_KGaA_Rpkg_validation_220202.pdf
8. Wickham H, Bryan J. *R Packages* edition 2, Appendix A. https://r-pkgs.org/R-CMD-check.html, accessed 20 January 2026.

ACKNOWLEDGMENTS

Thanks to our colleagues from the SCHARP R PQ team, whose work and insight lay the foundation for this paper: Radhika Etikala, Valeria Duran, Dave Slager, Xuehan (Emily) Zhang, and Amy Harper. Thanks also to Jiani Hu for programming the first draft of the R usage plan template.

CONTACT INFORMATION

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

Author Name: Shannon Grant
Company: Statistical Center for HIV/AIDS Research and Prevention at Fred Hutch Cancer Center
Email: sgrant@fredhutch.org