Paving the Way for Regulatory Submissions Using R: the Risk Assessment Application

Marly Gotti[1] Robert Krajcik[2] Aaron Clark[3] Doug Kelkhoff[4]
Yilong Zhang[5] Juliane Manitz[6] Joseph Rickert[7] Lyn Taylor[8]
Andy Nicholls[9] Paulo R. Bargo[10]

Abstract
In recent years, R has gained significant popularity as a statistical software in the academia mainly because it is free, open source, and vastly used in academic research. This popularity has reached the health sector and other regulated industries. As a consequence, an avalanche of concerns has been raised by statisticians, statistical programmers, quality assurance teams and others within the pharmaceutical industry about the use of R and selected R packages as primary tools for statistical analysis for regulatory submission work. To address such concerns, the R Consortium set in motion the R Validation Hub, a collaboration to support the adoption of R within a bio-pharmaceutical regulatory setting. In this paper, we will introduce the *Risk Assessment Shiny Application*, one of the tools currently in development by the R Validation Hub to aid in the validation of R packages in regulated environments.

Key Words: validation, risk, assessment, metrics, R, shiny, riskmetric

1. Introduction

Through the last decade, R has gained popularity as a statistical software in the academia because it has the advantage of being free and open source. The wide contribution of R packages by researchers, statisticians, and engineers has also propelled the use of R for statistical analysis. Such popularity has reached the health sector and other regulated industries. As a consequence, an avalanche of concerns has been raised by statisticians, statistical programmers, quality assurance teams and others within the pharmaceutical industry about the use of R and selected R packages as primary tools for statistical analysis for regulatory submission work. To address these concerns, the R Consortium set in motion the R Validation Hub, a "collaboration to support the adoption of R within a bio-pharmaceutical regulatory setting" ([6]).

At the beginning of 2020, the R Validation Hub put forth a white paper proposing " a possible risk-based approach for assessing R package accuracy within a validated infrastructure" (see [5]). In the white paper, a framework for validating R packages is proposed with special attention given to package accuracy.

Derived from this white paper, the R Validation Hub started a series of collaborations to create a couple of open source tools to assess package accuracy: the `riskmetric` R

[1]Biogen, 225 Binney Street, Cambridge, MA 02142
[2]Cytel, Inc. Waltham, MA 02451
[3]Biogen, 225 Binney Street, Cambridge, MA 02142
[4]Roche/Genentech, 1 DNA Way, South San Francisco, CA 94080
[5]MSD, 2000 Galloping Hill Road, Kenilworth, NJ 07033
[6]EMD Serono, 45 Middlesex Tpk, Billerica, MA 01821
[7]RStudio, 250 Northern Ave, Boston, MA 02210
[8]PHASTAR, 2D Bollo Ln, Chiswick, London, W4 5LE, UK
[9]GSK, 980 Great West Road, Brentford, Middlesex, TW8 9GS, UK
[10]Janssen R&D, 1400 McKean Rd, Spring House, PA 19477

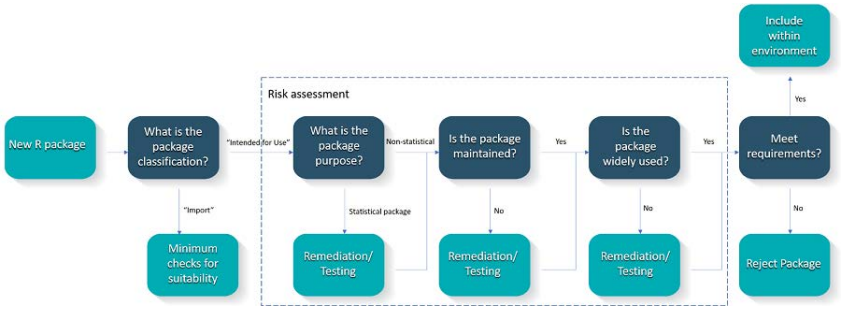

Figure 1: Assessing package accuracy workflow.

package and the *Risk Assessment Shiny Application*. These tools are freely available at [4] and [3].

The `riskmetric` package provides a variety of metrics to evaluate development best practices, code documentation, community engagement, and development sustainability of R packages. For more information about `riskmetric`, see the package's website [4]. On other hand, the *Risk Assessment Shiny Application* is an extension of `riskmetric` that offers a ready-to-use framework to assess the risk of using R packages. The application has the following functionalities:

- inherits the advantages of the `shiny` R package (no R programming knowledge is needed to use the application),
- allows users to provide feedback on the risk calculated by `riskmetric`,
- gathers information on community and maintenance metrics of the package,
- has embedded authorized personnel that can perform risk assessments and modify metric weights depending on the user access level,
- stores historical comments and final decisions,
- contains a reporting tool that allows users to share the assessment insights with other reviewers as either a Word Document or an HTML file.

In this paper we will present the *Risk Assessment Shiny Application* and describe how it can help in the adoption of R within a validating infrastructure.

2. The Risk Assessment Shiny Application

The main goal of the Risk Assessment Shiny Application is to provide a user-friendly framework to assess the risk of using R packages following the philosophy proposed by the white paper [5]. To understand how the application works and how we can set it up in a validated R environment, we will navigate the different functionalities of the application.

2.1 Initial setup

The Risk Assessment Shiny Application should be deployed as a regular shiny application (see "Hosting and deployment" instructions by RStudio at https://shiny.rstudio.com/deploy/). The following packages should be installed prior to deployment:

```
"shiny", "shinyhelper", "shinyjs", "shinydashboard",
"shinyWidgets", "data.table", "DT", "readr", "lubridate",
"RSQLite", "DBI", "rvest", "xml2", "httr", "desc", "dplyr",
"tools", "stringr", "tidyverse", "loggit", "shinycssloaders",
"rAmCharts", "devtools", "plotly", "cranlogs", "formattable",
"rintrojs", "shinymanager", "keyring".
```

The first time the application is run two lightweight `sqlite` databases will be automatically created: `database.sqlite` and `credentials.sqlite`. These databases store package metrics and user credentials. Subsequent runs of the application will not override these databases.

Two users, one *regular* and one *administrator* will be created automatically. The main difference between the two types of users is that *administrators* have access to parts of the application *regular* users do not: *administrators* can add/remove new users and access the different tables on the `database.sqlite` database (refer to Table 1 for default user credentials). After deploying the application, these users should be discarded and new ones should be added.

Table 1: Credentials for the default users created during the application deployment.

User type	Username	Password
regular	`john_doe`	`qwerty`
administrator	`admin`	`qwerty`

2.2 Overview

There are three main regions on the application: the *Package Control Panel* left side panel, the navigation tabs, and the body of the application which changes as different tabs are selected (see marked regions 1, 2, and 3 respectively in Figure 2). In the body of the application, regardless of the tab selected, we will see the *Need help?* button; clicking this button will set the application in help mode and provide a description of the different user interface elements in the tab.

For illustration purposes, we will perform a risk assessment of the well-known R package `dplyr` (see [8]) [1].

2.3 Uploading a package

After logging in to the application, we will be directed to the *Upload Package* tab. To perform the risk assessment of a set of packages, we must upload a `csv` file with the names of the R packages we would like to assess. A template of such `csv` is provided in the application for user's convenience.

A progress bar will indicate the application is calling `riskmetric`, gathering the package metrics, and calculating the risk. The risk of each of the uploaded packages will be calculated and saved on the database. To view the risk of a particular package, select it from the *Package Control Panel*.

[1] Note that `dplyr` and packages developed by RStudio should be very safe to use. Recently, RStudio released a series of articles documenting different approaches to R and R environment validation (see [1] and [7]).

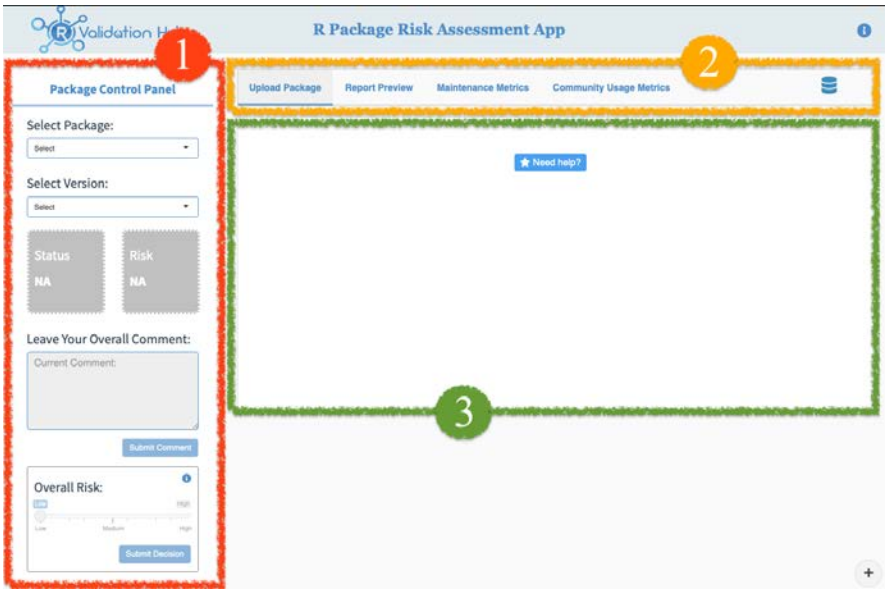

Figure 2: Main regions of the application: region 1 contains the Package Control Panel where users can select a package, review the risk of the selected package, comment, and make a final decision; region 2 displays the different tabs of the application; region 3 contains the main body of the application, which will be updated as different tabs are selected.

Let us upload a `csv` file following the template provided with the `dplyr` package (see Figure 3).

2.4 Reviewing the package metrics

In the left side panel of Figure 3, we can see the risk score of `dplyr` as 0.2. The risk score is a compound value of several metrics; the score ranges from 0 to 1, where 1 represents the highest possible risk a package can have. These metrics are distributed in two tabs on the app: *Maintenance Metrics* and *Community Usage Metrics*. The *Maintenance Metrics* tab contains the following metrics:

- *Presence of Vignettes?*: determines whether the package has vignettes and how many are present.
- *Associated Website Url?*: displays the public url of the package, if one exists.
- *NEWS?* and *NEWS is Current?*: show whether the package has a NEWS file and whether is up-to-date, respectively. As a good maintenance practice, developers should keep this file up-to-date with information describing the latest package releases and the changes incurred on such releases.
- *Bugs Publicly documented?*: displays the url to report bugs, if one exists.
- *Bugs Closure*: shows the percentage of the last 30 bugs closed.
- *Documentation*: displays the percentage of the package objects that have been documented.
- *Source Code Public*: shows the public package url, if one exists.
- *Has a Maintainer*: displays the name of the package maintainer, if one exists.

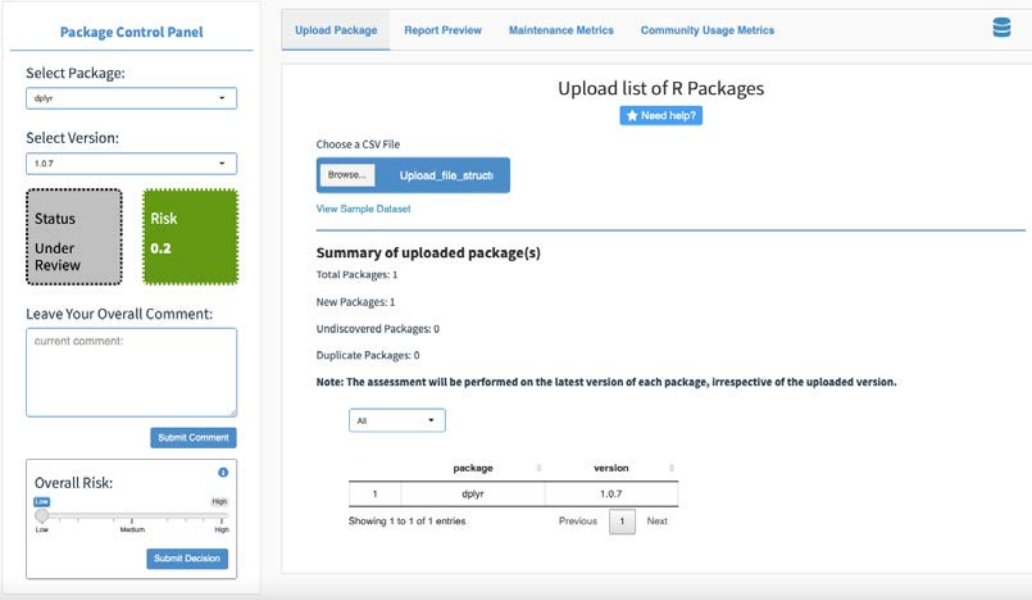

Figure 3: The *Risk Assessment Shiny Application* after uploading `dplyr`.

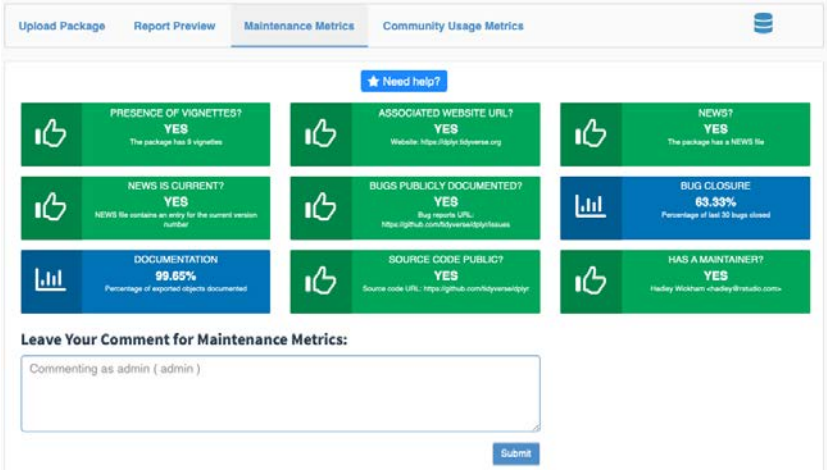

Figure 4: Maintenance metrics for `dplyr`.

For `dplyr` we can see that the main metric driving the risk score above 0 is the *Bugs Closure* rate, which is a slightly above 50% (see Figure 4).

In addition, we have the *Community Usage Metrics* tab. As the label indicates, this tab will dive into the interaction of users with the package, i.e., the number of downloads the package has had in a selected period of time. Figure 5 offers an insight into the number of downloads `dplyr` has had in the last two years. For instance, we can gather that, after release 1.0.1, the package has gained more attention from the community; this is shown by the steady increase in its number of downloads. There are multiple functionalities in this tab that allows us to go back two years and analyze whether the package has improved over time, or focus on a shorter time interval to analyze the impact a recent version has made on the overall view of the package by the community.

These metrics are being extended and improved frequently. For more information about the current metrics, refer to the `riskmetric` and *Risk Assessment Shiny Application* websites ([4] and [3]).

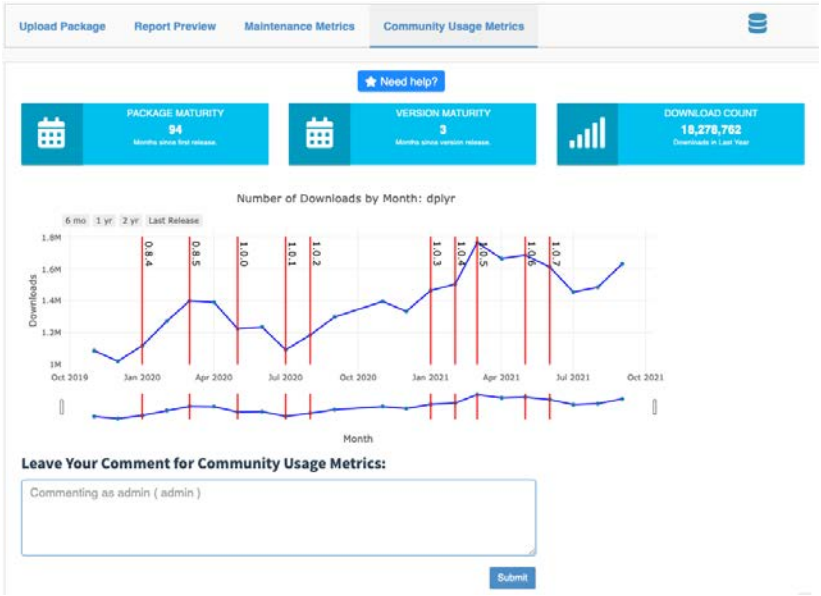

Figure 5: Community usage metrics for `dplyr`.

2.5 Final decision and comments

As performing risk assessment is a holistic process, we should carefully evaluate not only the final risk provided by `riskmetric` but also how the interconnected metrics may affect this score. When we navigate through the different metrics, we have the option to leave comments; such comments will be saved permanently on the database, allowing inspectors or other reviewers to understand the justifications behind a particular decision. In the case of *dplyr*, for instance, we should be inclined, before making our final decision, to look further at the bugs/issues of the package given that the bug closure rate is a slightly above 50%. In addition, since the version we are analyzing, i.e., 1.0.7, was recently released, we may deter from using this version in favor of a previous, more stable one. Doing so will give the package more time to marinate in the community and decrease the risk of having unforeseen issues that can easily be avoided with more community exposure. By virtue of these reasons, we can make the final decision declaring the package medium risk, leave a comment on that regard, and proceed to do a risk assessment of an older version of `dplyr`. The final comment/decision sections can be found on the left side panel of the application.

2.6 Creating a report

At any moment during the review process, we may create a report. This report will contain general information on the package, the package risk score, the values of each metric, comments, and the final decision (if one has been made). There are two available formats: `html` or `docx`.

2.7 Admin role and advanced settings

Administrators have access to the advanced settings view of the application, allowing them to add/remove users. It is recommended that administrators change the credentials of the default users right after the application is deployed (see Table 1 for a list of the default users).

In addition, administrators can modify the default weight of each metric. It is not recommended to change these weights at any point other than right after deployment, as this may lead into audits down the line and can render previous risk assessments invalid. However, if administrators need to change these weights, then they may do so in the database view of the application. After changing the weights, administrators will have the option to re-calculate the risk of each package in the databse based on these new weights. Notice that this may render previous final decisions and comments invalid. To this end, final decisions will be voided and a new comment will appear in each metric stating the change of weights and prompting the reviewer to re-evaluate each package. If administrators change the weights but do not tell the application to re-calculate the risk of each package, then only future uploaded packages will have their risks calculated based on these new weights.

3. Further Resources

The increasing use of R within the bio-pharmaceutical community has resulted in numerous initiatives and approaches to validation (see [5], [7], [1], [2]). As these ideas mature, a common approach to validation should start to emerge. In the meantime, the interested reader can share share ideas and keep up-to-date by joining the mailing list for the R Validation Hub https://www.pharmar.org/contact/. We also welcome contributions to `riskmetric` and the *Risk Assessment Shiny Application*; the reader can do so by reaching out to our team at [4] and [3].

4. Acknowledgments

The authors would like to thank the R Consortium, the R Validation Hub, and all the contributors of `riskmetric` and the *Risk Assessment Shiny Application*.

References

[1] P. Bowsher and S. Lopp. R Validation: Approaches and Considerations, 2021. PharmaSUG 2021 Proceedings. `https://www.pharmasug.org/proceedings/2021/SI/PharmaSUG-2021-SI-203.pdf`, Last accessed on 2021-09-29.

[2] M. Ginnaram, S. Ye, Y. Zhu, and Y. Zhang. A Process to Validate Internal Developed R Package under Regulatory Environment, 2021. PharmaSUG 2021 Proceedings. `https://www.pharmasug.org/proceedings/2021/SI/PharmaSUG-2021-SI-084.pdf`, Last accessed on 2021-09-27.

[3] M. Gotti, A. Clark, M. Gans, R. Krajcik, and R Validation Hub. *Risk Assessment Shiny Application*, 2020. A shiny application to aid in the validation of R packages. `https://github.com/pharmaR/risk_assessment`.

[4] D. Kelkhoff, M. Gotti, E. Miller, K. Kevin, Y. Zhang, E. Milliman, J. Manitz, and R Validation Hub. *riskmetric: A collection of risk metrics to evaluate the quality of R packages*, 2019. R package version 0.1.1. `https://github.com/pharmaR/riskmetric`.

[5] A. Nicholls, P. R. Bargo, J. Sims, and R Validation Hub. A Risk-based Approach for Assessing R package Accuracy within a Validated Infrastructure, 01 2020. `https://www.pharmar.org/white-paper/`, Last accessed on 2021-09-28.

[6] R Validation Hub. R Validation Hub: Mission, 2021. https://www.pharmar.org/about/, Last accessed on 2021-09-25.

[7] RStudio. Environments and Validation: Using R for Validated Work, 2021. https://environments.rstudio.com/validation.html, Last accessed on 2021-10-01.

[8] H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2018. R package version 1.0.7.