PharmaSUG 2024 - Paper SS- 376

Exploring New Technologies to Package R/Shiny Applications for Submissions to FDA

André Veríssimo, Ismael Rodriguez, Appsilon

Abstract

Pilot 4 of the R Consortium Submission Working Group marks a pioneering step in using R for FDA clinical trial submissions, introducing novel technologies like WebAssembly and Containers to elevate submission efficiency and effectiveness. This initiative, in collaboration with Appsilon, is revolutionizing the Shiny application packaging and transfer process, establishing a new benchmark that complements the static and extensive documentation.

At the heart of Pilot 4's innovation are WebAssembly and Containers. WebAssembly enables R applications to run almost at native speed in web browsers, bypassing complex local installations and greatly enhancing access. This advancement is crucial for more efficient and secure operation of R-based applications, especially under the strict regulations of clinical trials.

Containers, notably Podman, allow for the same benefits of WebAssembly in terms of portability with the advantage of being a well-established technology. Podman, in particular, is an open-source tool that places a strong emphasis on security, making it an ideal choice to comply with stringent FDA requirements.

Pilot 4 is set to split into two separate submissions, focusing on WebAssembly and Containers, respectively. This approach facilitates a thorough assessment of each technology's capabilities in refining clinical trial submissions.

As the R Consortium awaits feedback on previous pilots and moves forward with Pilot 4, this initiative is expected to significantly expedite clinical trial submission practices. Integrating these advanced technologies underlines the Consortium's dedication to deploying R-based methods for more effective and technologically sophisticated submissions in the future.

Introduction

The first pilot, a pioneering effort submitted in November 2021, was the first of its kind to be publicly available. It included an R package, scripts, and an R-based Analysis Data Reviewed Guide (ADRG), setting a precedent in the submission process. The FDA's response, received in March 2022, marked a crucial step in this groundbreaking journey.

The second pilot furthered this innovation by incorporating a Shiny application, the first to be included in such a submission package. Utilizing the datasets and analysis from the first pilot, this submission was a testament to the evolving use of R in clinical trials. The application is accessible online, with alternative versions also available.

What's next?

Pilot 3

This pilot was successfully submitted to the FDA on Aug 28, 2023. This was the first publicly available R submission that included R scripts to produce ADaM datasets and TLFs. Both the ADaMs (SDTM .xpt sources from the CDISC Pilot study) and the TLFs (ADaMs .xpt sourced from the ADaMs generated in R by the Pilot 3 team) were created using R. The next step for this pilot is to await FDA's review and approval, which may take several months to complete.

Pilot 4

This pilot aims to explore using technologies such as containers and WebAssembly software to package a Shiny application into a self-contained unit, streamlining the transfer and execution process for enhanced efficiency (Perkel, 2024).

This pilot is expected to be divided into two parallel submissions:

a. will investigate WebAssembly and;
b. will investigate containers.

The Journey with WebAssembly and Containers

Our team at Appsilon teamed up with the dynamic Pilot 4 crew to explore WebAssembly technology and containers. George Stagg and Winston Chang also joined the working group to discuss the web-assembly portion of Pilot 4. This partnership brought together our engineering prowess to contribute to these tools, injecting fresh perspectives into the ongoing pilot project.

Some of the outcomes of the collaboration:

1. We were able to set up a robust container environment for this pilot project;
2. We aided the progress made on the use of both experimental technologies: containers and WebAssembly;
3. We developed a working prototype submission using Podman container technology;
4. We developed a working early-stage prototype for wrapping a small Shiny application using WebAssembly.

WebAssembly

Figure 1. webR logo.

WebAssembly allows languages like R to be executed at near-native speed directly within web browsers, providing users with the ability to run R code without having R installed locally. WebR is essentially the R programming language adapted to run in a web browser environment using WebAssembly. This project is under active development.

The Pilot 4 Shiny App Up and Running on webR!

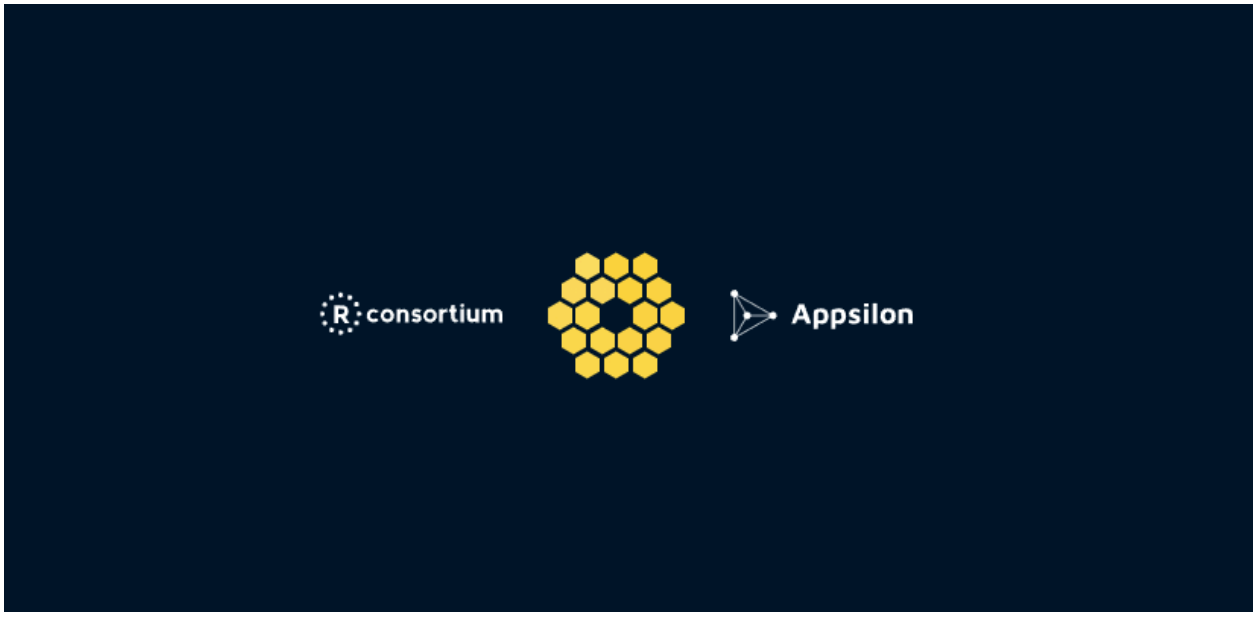

Figure 2. Initial Screen of Shiny App While It Is Loading.

The deployed example of the Shiny app running on webR is publicly available. Check out the video of the application running.

During this pilot, engineers at Appsilon developed a prototype of a Shiny application running on webR. The application reuses most of the code from the previous pilot apps with some tweaks and a couple of hacks/changes to get around non CRAN dependencies, specially for data loading, WebR compatibilities, and shimming some of the functionality from {teal} and other packages that are (for now) not available on CRAN.

webR Shiny App

During the **second** iteration, which was recently held, Pedro Silva shared the process of developing this Shiny app running on webR.

The Process

1. Leverage the last 2 iterations of the application
    - Reuse as much code as possible
    - Avoid touching the logic part
2. Restrict the number of dependencies to packages on CRAN
    - Replace/shim functionality that was lost from removing dependencies

Here is the list of dependencies to packages on CRAN (see Table 1); those that worked are colored green, and those that were removed are marked in orange. We ended up with just 3 problematic dependencies (bold).

library(config)	library(reactable)
library(cowplot)	library(rhino)
library(dplyr)	library(rtables)

library(emmeans)	library(shiny)
library(ggplot2)	library(stringr)
library(glue)	**library(teal)**
library(haven)	**library(teal.data)**
library(htmltools)	library(tibble)
library(huxtable)	library(tidyr)
library(magrittr)	library(tippy)
library(markdown)	library(Tplyr)
library(pkgload)	library(utils)
library(purrr)	library(visR)

Table 1. List of Dependencies to Packages on CRAN.

Issues with library(cowplot):

- Some issues with low-level dependencies when deployed.

Solution:

- Replace functionality with HTML.

Issues with library(teal):

- Uses {shiny.widgets} (not working for webR).

Solution:

- Redo the UI;
- Load modules directly;
- Recreate filter functionality.

Issues with library(teal.data):

- Use rds exports.

Solution:

- Shim functionality, load data directly;
- Leverage shinylive and {httpuv} to export and serve the application;
    a. Shinylive can help streamline the export process;
        i. Problems:
            1. shiny.live won't let us have non-R files in the application directory - this is an outstanding bug that George asked us to raise an issue for.

2. We wouldn't be able to run the application as a traditional shiny app.

ii. Solution:

1. Custom build script.

b. {httpuv} can help serve the application.

i. {httpuv} would run natively on a machine to serve the Shiny app

Application Structure

The figure below shows an overview of what we ended with:

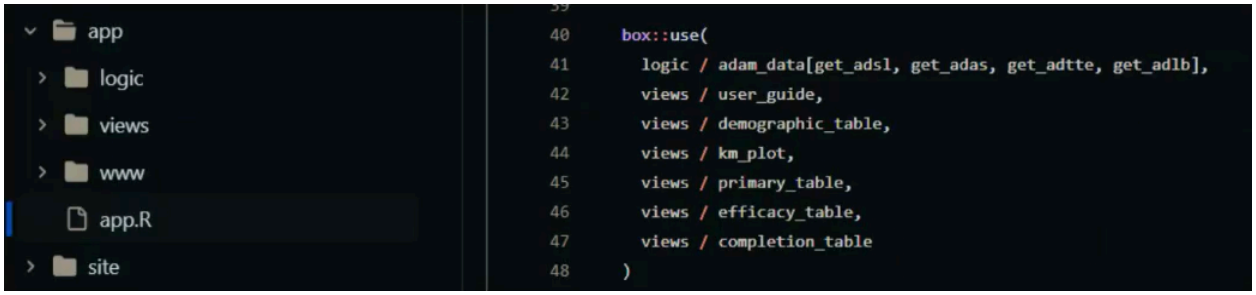

Figure 3. Overview of the Application Structure

Some of the issues and solutions found at the very beginning:

1. The previous applications were built using golem and another one in Rhino; the support for these frameworks is not great in webR up to now.
    1. Solution
        1. {box} works out of the box (reuse the rhino version modules)
        2. Simplify the structure and use a simple shiny modular structure
2. Shinylive does not like non-R files when generating the bundle
    1. Solution
        1. Keep the app folder as clean as possible for now (www folder only)
3. {teal} and {teal.data} are not on CRAN
    1. Solution
        1. Shim and used functionality
        2. Use a simple tab system for the UI structure

The FDA was previously told that the shiny application being prepared for the Pilot 4 submission would not be a 1 to 1 mapping from the previous one submitted for the Pilot 2 due to certain constraints such as {teal} not being on CRAN; however, this didn't represent a problem for them since they would mainly like to test the technology.

Pedro Silva, one of the engineers working on the development of this app, mentioned "While WebR is still in development, it shows tremendous promise! The loading is definitely still a pain point (over 100mb to set up the environment!) but it will only get better moving forward."

Containers

Figure 4. Containers

Containerization, particularly through technologies like Docker, Podman or Singularity, offers several advantages for deploying Shiny apps.

Choosing the Right Container

Choosing the right container was a question that arose in this project. Although Docker is the most popular, we decided to move forward with Podman.

In our exploration of containerization tools for deploying Shiny applications, we've identified key distinctions between Docker and Podman that influenced our choice.

Podman stands out for its daemonless architecture, enhancing security by eliminating the need for a central daemon process. Unlike Docker, Podman supports running containers as non-root users, a critical feature for meeting FDA reviewer requirements. Developed by Red Hat and maintained as an open-source project, Podman prioritizes security with its rootless container support, offering a robust solution for security-conscious users.

Goals

A Container-based method to deploy Pilot 2 Shiny App.

What we did

1. Configurable Podman Dockerfile / docker-compose.yml
    - R version
    - Registry / organization name / image name (differences between docker.io and ghcr.io)
2. Documentation on creating the container
3. CI: Automated build on amd64 and arm64 platforms

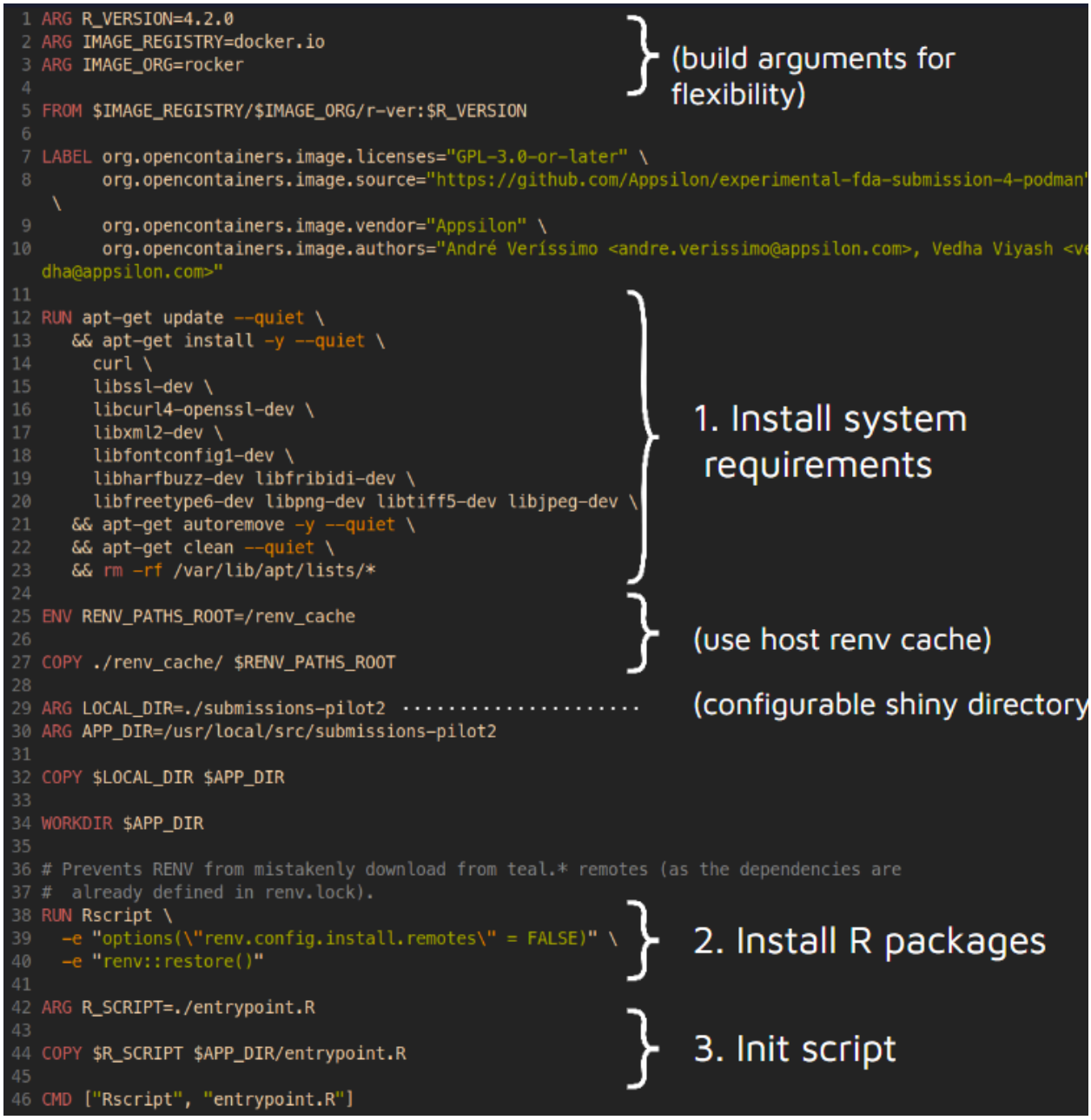

Figure 5. Dockerfile (Recipe) for the Container

Next Steps

The immediate priorities include awaiting the FDA's review of Pilot 3 and submitting the two segments that investigate new technologies to regulatory bodies. The collaboration between the R Submission Working Group and various partners has yielded a functional prototype of a Shiny application similar to {teal}, operating on webR, with ongoing developments in Podman integration.

Figure 6. Rhino R Package Logo.

Regarding Rhino Compatibility, Appsilon is concurrently focusing on Rhino compatibility. In the future, this framework might seamlessly integrate into the Pilot 4 application.

Concerning {teal} and other similar packages, they may soon be available on CRAN. Post this, we could integrate them, replacing the temporary solutions implemented in this version.

As for Boot Time, enhancing the boot time is crucial. This involves removing dependencies and continuing advancements in webR.

Conclusion

Pilot 4 of the R Consortium Submission Working Group, in collaboration with Appsilon, signifies a groundbreaking endeavor in the utilization of R for FDA clinical trial submissions. By integrating innovative technologies such as WebAssembly and Containers, specifically Podman, this pilot sets a new standard for submission efficiency and effectiveness.

The adoption of WebAssembly facilitates R applications to operate at near-native speeds directly within web browsers, streamlining access and operation without the need for complex installations. This feature is particularly beneficial for the stringent regulatory environment of clinical trials, ensuring secure and efficient handling of R-based applications. Containers, with a focus on Podman, offer similar advantages in terms of portability and security, making them ideally suited to meet FDA's rigorous requirements.

As the consortium anticipates feedback and progresses with Pilot 4, the integration of these technologies is expected to revolutionize clinical trial submissions, paving the way for more efficient, secure, and technologically advanced methods in the future.

References

Perkel, J.M. (2024) ‘No installation required: How webassembly is Changing Scientific Computing’, Nature, 627(8003), pp. 455–456. doi:10.1038/d41586-024-00725-1.

Contact Information

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

André Veríssimo
Appsilon
andre.verissimo@appsilon.com

Ismael Rodriguez
Appsilon
ismael.rodriguez@appsilon.com
www.appsilon.bio