PhUSE US Connect 2024
Paper for Presentation OS10

Posit/RStudio Pharma Updates: Reproducibility and Beyond

Phil Bowsher, Posit PBC; Boston, USA,
Cole Arendt, Posit PBC; Boston, USA,

Abstract

Interest in using open source for clinical trials has increased significantly in the last few years. Much of this recent growth has been propelled by major successes in 2022-2023:

1. Increasing Number of Pharmas use R in Submissions:
    a. https://github.com/philbowsher/Open-Source-in-New-Drug-Applications-NDAs-FDA
2. Multiple Successful R Submission Pilots
    a. Various industry contributors are working closely with the FDA to submit R, Shiny and other tools to the FDA. Please read more here: https://posit.co/blog/fda-shiny-r-package-submissions/
    b. Follow the Working group here: https://rconsortium.github.io/submissions-wg/
3. Roche Shifted to an Open-source Backbone in Clinical Trials
    a. Roche has frozen the legacy macros library and R is the default backbone for evidence generation in late-stage. You can learn more about this transition here: https://posit.co/blog/roche-shifting-to-an-open-source-backbone-in-clinical-trials/
4. Novo Nordisk Completes FDA Submission Package
    a. Outputs exclusively written in R
    b. You can learn more about this accomplishment here: https://posit.co/blog/novo-nordisks-journey-to-an-r-based-fda-submission/
5. GSK & Other Pharmas Commit to Open-Source
    a. GSK has stated publicly that by the end of 2025 >=50% of new code should be written in open-source languages
6. Pan-Pharma Cross-Collaboration
    a. Pharmaverse, R Validation Hub and others. You can read more here: https://colorado.posit.co/rsc/open-source-pharma/#/title-slide

Much of the transition is driven by key return on investment (ROI) opportunities such as:

- Partnerships and collaborations with external partners
- Streamlined Workflows
- Talent out of university more likely to know open-source programming languages
- Latest developments more rapidly
- Switch between languages and contexts more easily
- Time savings in creating standard TLGs
- Time savings in creating customized TLGs
- Access to free solutions to support creating a clinical submission

Many smaller to medium pharmas are starting the transition now. This paper will provide strategies focusing on open source for late stage work. This paper is focused on ***small to medium pharmaceutical and biotechnology companies venturing into open source for submissions***.

Introduction

1. Start Simple

Many organizations ask how to transition large teams and complex projects to open source. However, many of the organizations referenced above started many years prior by establishing small wins that helped set the foundation for open source. For example, Andy Nicholls (GSK), recently wrote in a Phuse EU 2023 paper:

“Since 2017, GSK has embarked on various initiatives aimed at creating a ‘multilingual’ Statistics, Programming, and Data Science community”

https://phuse.s3.eu-central-1.amazonaws.com/Archive/2023/Connect/EU/Birmingham/PAP_TT09.pdf

Transiting to open source does not happen overnight. GSK further discusses this journey in a recent posit::conf(2023) talk: **The Need for Speed - AccelerateR-ing R Adoption in GSK**:

https://www.youtube.com/watch?v=VDu2qdpYko8

Ben describes the journey in the abstract:

“How does a risk-averse Pharma Biostatistics organization with 900+ people switch from using proprietary software to using R and other open-source tools for delivering clinical trial submissions? **First slowly, then all at once. GSK started the transition of using R for its clinical trial data analysis in 2020 and now uses R for our regulatory-reviewed outputs.** The AccelerateR Team, an agile pod of R experts and data scientists, rotates through GSK Biostatistics study teams sitting side by side to answer questions and mentor during this transition. We will share our experience from AccelerateR and how other organizations can use our learnings to scale R from pilots to full enterprise adoption and contribute to open source industry R packages.”

When starting, pick a smaller project that is a good use case for open source. Many pharmas started this journey with Shiny in the exploratory space which can lead to adoption of R in late-stage work. For example, many pharmas start with using Shiny in an area of drug development like pharmacokinetics (PK) and pharmacodynamics (PD). Valentin Legras, Senior Data Scientist at Roche recently describes this journey in the talk:

Revolutionizing Data Exploration: An Interactive Journey through R Shiny

https://phuse.s3.eu-central-1.amazonaws.com/Archive/2023/Connect/EU/Birmingham/PRE_DV02.pdf

The slides show how Roche uses Shiny for PK/PD, Safety and Efficacy.

Many organizations will then start to incorporate R for a few examples in stage 1-2 such as in Clinical Pharmacology where scientists use R for statistical and graphical analyses. For example, R is often used for nonlinear mixed effects modeling and subsequent graphical analysis. Below is an example where ggplot2 is used for graphics as well as the R xpose package:

http://www.accessdata.fda.gov/drugsatfda_docs/nda/2016/208573Orig1s000ClinPharmR.pdf

As the organization grows in its use of open source, a next common place in the late stage realm is to use R to generate TABLES, LISTINGS, AND FIGURES (TLFs) in R for stage 3 analysis as seen in the Novo Nordisk webinar above. From there, organizations will start to use R to create SDTM and ADaM datasets as well as add interactivity components to submissions with Shiny, Shinylive, webR, HTML and Javascript.

Below are a few simple foundational ways to start your open source journey:

2. Create an Open Source Language Version and Package Plan

This is a critical first step. The main goal of this effort is to identify which R packages are needed. This is usually driven by business demand. When many organizations start down the path of adding open source to the late stage environment, they find that many users are already using open source as in the examples mentioned previously. This is often done on independent laptops and workstations. A **first step is to share a survey or form with the statistical programmers, biostatisticians, and clinical data scientists to discover which packages are currently being used**. This effort will help discover the packages utilized by the various business groups such as pharmacokinetic groups, safety teams etc.

The other information to identify is the R and Python versions in use. Organizations vary on how often they add a new version of R to the controlled frozen environment. Some organizations align to the major upgrades of R (about every 6 months, while others do it once a year or longer. Balancing upgrades is a critical step in aligning with the needs and expectations of programmers.

After the information is collected, IT then can create a plan that identifies the packages (internal and external) used within the late-stage team. This often includes reviewing package licenses as well as standards for quality. This will be discussed further in the next step.

A popular question asked by many organizations new to use open source for submissions is:

“Is there a list of R packages that are currently accepted for regulatory R submissions?”

Usually this question is asked by organizations starting out and looking for advice on how to set up the environment for controlled clinical work. While no specific list exist, there are some important documents and efforts to review:

The R Submission Pilots list the packages used in the FDA Submissions. The submissions were deemed successful by the FDA. The packages used can be seen here:

https://github.com/RConsortium/submissions-pilot1/blob/main/renv.lock

The first submission was focused on generating Tables, listing and figures in R. Pilot 2 added a Shiny app to a submission. Pilot 3 was focused on utilizing R to generate ADaM datasets. Pilot 2 and 3 were also successfully submitted through the FDA eCTD gateway. Pilot 3 was submitted in September of 2023. Comparing the list of R packages used from pilot 1 to 3 reveals new packages such as Admiral. You can see the package list here:

https://github.com/RConsortium/submissions-pilot3-adam/blob/main/renv.lock

Moreover, RStudio in 2020, in collaboration with the community and pharma, released validation guidance documents covering the core packages used for data management and reporting. This includes the tidyverse, tidymodels, r-lib, gt, shiny, rmarkdown and others. The main page and direct links are below:

https://www.rstudio.com/assets/img/validation-tidy.pdf
https://www.rstudio.com/assets/img/validation-shiny-rmd.pdf

There is also a Pan-Pharma Cross-Collaboration called the Pharmaverse. The Pharmaverse is a curated set of open source R packages for clinical reporting. Manoy of these packages adhere to high levels of quality and include test and other documentation.

Given the changing nature of the R ecosystem and more specially the Pharmaserve, having a set package list is unlikely. Rather than referring to a list, organizations will assess the business needs and add packages that meet internally defined criteria for quality which is covered in the next section.

The Regulatory R Package Repository Working Group is working to provide information about packages used for clinical reporting. You can follow this effort here:
https://pharmar.github.io/regulatory-r-repo-wg/

Lastly, it is important to communicate with the regulatory bodies. This includes discussing the availability of using R prior to submission. If applicable, submit sample data and codes if available. You can learn more via the webinar: Using R in Regulatory Review
https://www.youtube.com/watch?v=dtdd_jc1ybw
In it, the FDA discussed a RWD submission with Astellas Pharma US for tacrolimus, which was approved here:
https://www.fda.gov/drugs/news-events-human-drugs/fda-approves-new-use-transplant-drug-based-real-world-evidence

You can see the dialog with the FDA here:
https://www.accessdata.fda.gov/drugsatfda_docs/nda/2022/050708Orig1s053;%20050709Orig1s045;%20210115Orig1s005.pdf

3. Create Standard Operating Procedures (SOP) to "Validate" Packages

Validation and qualification are topics that can vary in the interpretation from organization to organization. While guidance documents do exist, the internal implementation often varies. In addition to the Posit, Pharmaverse and other packages, pharmaceutical organizations will also add other packages (internal and external) to the baseline of packages to be used by statistical programmers. Organizations will often review packages to be included based on the R Validation Hub's Risk-Based Approach guidance here:
https://www.pharmar.org/white-paper/

The riskmetric (Risk Metrics to Evaluating R Packages) package facilitates assessing R packages against a number of metrics to help quantify their robustness.
https://pharmar.github.io/riskmetric/articles/riskmetric.html

There is also a riskassessment Shiny app also provides a tool that helps organizations assess packages. View an example app here:
https://rinpharma.shinyapps.io/riskassessment/

You can learn more about these tools here:
https://www.youtube.com/playlist?list=PL4IzsxWztPdkuta1iqLmZckBdc1N4nZ9c

Some organizations also use the guidance by the Transcelerate Modernization of Statistical Analytics (MSA):
https://www.transceleratebiopharmainc.com/initiatives/modernization-statistical-analytics/

Posit Validation information is here:
https://solutions.posit.co/envs-pkgs/environments/validation/

Many organizations will implement a process for further reviewing packages. There are various R tools to aid in this effort such as the ones listed below:

thevalidatoR - https://github.com/insightsengineering/thevalidatoR
valtools - https://github.com/phuse-org/valtools
testthat - https://testthat.r-lib.org/

Novo Nordisk documents its process of validation with testthat here:
https://phuse.s3.eu-central-1.amazonaws.com/Archive/2023/Connect/EU/Birmingham/PAP_SM08.pdf

Some organizations prefer to buy validation documentation, testing and packages. An option is from Atorus, where they provide the validation tests and documents as well services to support validation. These activities and packages are available via **OpenVal**. OpenVal encompasses many areas of open source validation including a subscription-based repository of nearly 300+ validated R packages and growing.

Below is information on OpenVal for organizations looking for assistance. OpenVal is a subscription based validated package and R environment delivery system in support of the R programming language. OpenVal helps with the assembly of the environment itself and will run rigorous tests to ensure not only the packages are supplied but a properly configured and functioning environment across several different operating systems. A core component of OpenVal is the validation process for accepting packages into the environment as described below:

1. **Relevance and Quality:** Atorus collaborates with customers and leverages knowledge of industry initiatives (R Validation Hub etc.) to include relevant, quality, and stable packages. Impact assessments are also completed for version updates like R releases and package version updates.
2. **Risk Assessment:** Atorus determines the level of testing that is required of a package using open-source tools like riskmetric (R Validation Hub etc.) to provide the baselines. Metrics like package author, number of downloads, number and quality of references, number of vignettes, and package type to determine an overall risk score. Moreover, existing tests and other qualitative checks are included.
3. **Validation Plan:** Atorus establishes the requirements and applies tests to validate packages. Requirements and tests are written to prove packages execute as expected. The risk assessment determines how much testing is completed for packages, with some packages being more thoroughly tested.
4. **Package Validation:** Atorus authors programmatic tests as necessary based on the risk level for packages and adds new tests to the larger testing suite. The test and results of assessing packages are compiled in a validation document showing the requirements and tests that were performed as well as the results.

Using the installation and test execution scripts, all R packages are automatically installed and on-system testing is executed to produce the environment Package Validation Summary. The summary is used as documentation that the packages have passed validation checks and can be supplied to Q/A teams.

4. Freeze Packages in Posit Package Manager

The documents and tools above, and the implemented risk-based approach, allow organizations to establish a baseline of pharma specific packages that are deemed controlled for internal use. Organizations can use Posit Package Manager mentioned above to create an internal repository of packages selected specifically by Quality and IT for clinical trials. Validated packages, either from internal review or bought via OpenVal, can be surfaced via a repository (Validated) in Posit Package Manager while the system dependencies and environment configurations happen externally.

CRAN, out of the box, uses the latest of everything. Therefore many organizations use Posit Package Manager to create frozen internal repositories of R, Bioconductor and Python packages. Below are resources from some of large pharmas speaking about this effort:

- Johnson & Johnson: https://youtu.be/VYKShbR-pd8?list=PL9HYL-VRX0oTu3bUoyYknD-vpR7Uq6bsR&t=647
- Roche: https://www.youtube.com/watch?v=nqJsLSLd39A&t=905s
- Merck: https://www.youtube.com/watch?v=RBVqKi3FV30&t=1769s
- Novo Nordisk: https://youtu.be/t33dS17QHuA?t=600

Internal Repository of Packages for Clinical Trials

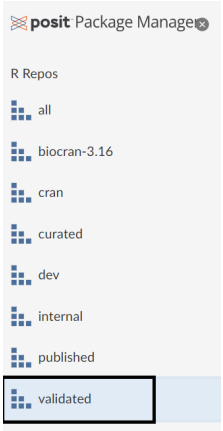
The documents above, and the implemented risk-based approach, allow organizations to establish a baseline of existing pharma specific packages that are aligned to internal validated efforts. Posit Package Manager can also aid IT in vulnerability scanning based on the osv.dev database, as well as the ability to easily block all packages with known vulnerabilities or problematic licenses.

Organizations can use Posit Package Manager mentioned above to create an internal repository of packages selected specifically by Quality and IT for clinical trials. Technical steps to do this are outlined here:

https://solutions.posit.co/envs-pkgs/environments/validated/

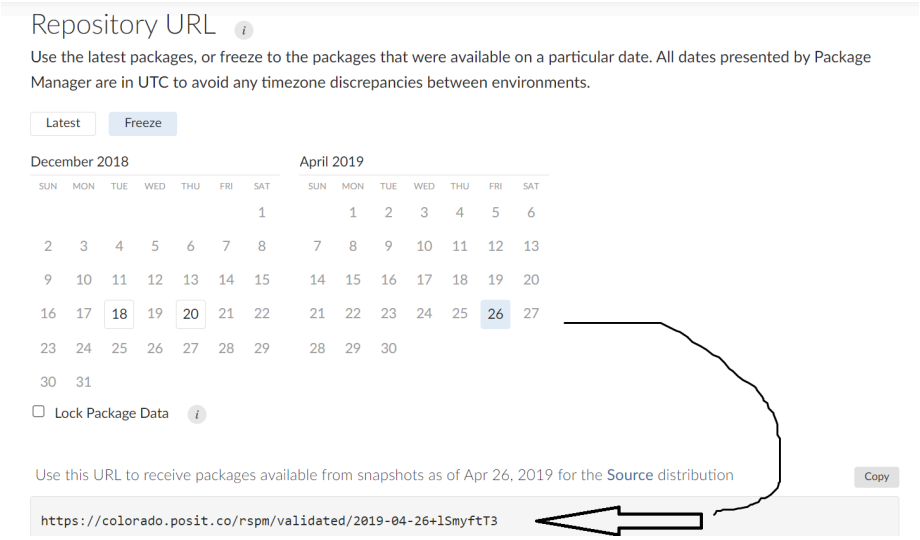
https://github.com/philbowsher/Creating-an-Internal-Repository-of-Validated-R-Packages

Posit Package Manager can host many repositories within the organization such as a Validated repo:



The Validated repository within your organization is often referred to as the “Shared Baseline” or Curated subset.

On the clinical side, Posit Package Manager allows for multiple repos for sharing one Validated source with snapshots or Multiple Validated repos for each version of R. Posit Package Manager also provides versioned package repositories (Snapshots) which are a copy of CRAN from a specific date. This enables time-traveling across different package versions so IT and users can recreate environments used on an historic date. IT will often upgrade the packages once or twice a year based on major releases of R.



Some organizations prefer to have a separate Validated Repo per R version rather than using Snapshots.

Posit Package Manager regularly syncs PYPI, CRAN and Bioconductor. It then builds linux binary packages for CRAN and offers them via the RSPM Sync service when ready. The RSPM sync service not only offers the package but also metadata on where to find the packages that are not on CRAN or Bioconductor. Each Posit Package manager instance regularly syncs precisely this metadata and once it is there, Posit Package Manager offers all the newly added packages in the user interface and to the user. This process takes time and package manager typically does not sync every day as seen here: https://packagemanager.posit.co/client/#/repos/cran/activity

Also package manager does not provide a time-based snapshot every day:

https://packagemanager.posit.co/client/#/repos/cran/setup?snapshot=2024-01-09

There is a bit of lag between a package being released on CRAN and the same appearing on package manager (1-31+ days).

In addition to R, organizations can create curated repositories for Python packages from PyPI. More on managing open source packages can be found here:

https://solutions.posit.co/envs-pkgs/manage_pkgs/

5. Deploy Frozen/Controlled Packages into Site/System Library

Within RStudio, and for other controlled R executions (batch), IT Admins will surface the Pharma Specific packages identified by the organization into a frozen environment. In a qualified and validated system, Posit strongly recommends to pre-install all “validated” packages in a site library. That way a user has the reassurance that any package they can load without installing something is approved. It also reduces the complexity of package validation. For example, for GAMP level, if allowing the installation of “validated” packages on-the-fly, is often regarded as a modular software install and requires more testing etc. If software components are pre-installed and hence immutable, the organization will often test/validate once. Below is information on installing base package sets:

https://solutions.posit.co/envs-pkgs/manage_pkgs/#installing-base-package-sets

Posit is a proponent of setting repository locations by the IT Admin in Rprofile.site. It can be challenging for reproducibility to allow users to update and change repositories in the RStudio IDE via Global options or similar actions. If a user requires setting repos, it is advised to do that in the .Rprofile in the project folder and then also add this .Rprofile to a version controlled process.

The packages that have been reviewed will be surfaced as pre-installed packages into the System library. For packages managed centrally with a site library, IT will often configure the user-interface of the RStudio IDE to discourage end-user package installation. In RStudio Workbench use allow-package-installation=0 in /etc/rstudio/rsession.conf.

Repositories can allow various use-cases within Posit Package Manager. These packages can also be surfaced to Windows desktop users as well as Linux Servers. Some organizations will allow users to install packages from the various supported repositories within Posit Package Manager. This is often true for exploratory environments. IT can also configure these environments so that packages are installed automatically by users from the validated environment if required as explained above.

As organizations accumulate different R versions, IT can also create a process to set the appropriate repositories depending on the active version of R selected by users in RStudio. The frozen environments usually have a locked R version and corresponding R packages. You can read more about best practices here:

https://github.com/philbowsher/Managing-Multiple-Versions-of-R-in-a-Clinical-Environment

6. Images/HPC/Docker

When reproducibility requirements extend deeper than packages, repositories, and language versions, we begin to approach reproducing entire computer systems. Immediately concerns arise for operating systems, low level building blocks like C compilers and linear algebra libraries, and system components written in domain specific languages like Fortran. These thoughts can also begin from a desire for increased computational resources; if the CPU and Memory requirements outpace what a single computer or server can provide. We must then think about clustered machines and having a uniform way to build tens, hundreds, or thousands of identical computers.

The most naive way to do this is to use Virtual Machines or Virtual Machine Images, oftentimes with other names like Amazon Machine Images (AMIs) in respective cloud providers. These are large and unwieldy reproductions of an entire machine, including all of its components and the linux kernel measured in the 10s to 100s Gigabytes. What is worse, they are holistic, hard to maintain, and expensive / time consuming to build. These challenges spurred the software industry to create a solution in what is commonly known as “Docker Images” or “Containers,” but more generically as the Open Container Initiative:
https://opencontainers.org/

These containers and container images are a mechanism for reproducing *most* of a system in a layered and easily reusable format that excludes the linux kernel so that sizes can be measured in Megabytes. Then these containers are built with specific and narrow purposes, so that the maintenance and reproducibility is scoped tighter and therefore easier to reason about and faster to compute. Some common related topics concern turning monolithic applications into microservices; essentially creating purpose-built solutions to a single problem instead of a large and tangled web of interdependencies.

These containers then become building blocks for more complex processes, or processes that need to be run across hundreds or thousands of computers. Instead of moving tens or hundreds of Gigabytes to each computer, or altering the purpose of the entire cluster, you move hundreds of megabytes of purpose-built environment to a generic computer interface that can then execute that workload. The process for distributing this workload across many computers is often called “High Performance Computing (HPC)” in the scientific world or, in the software world, distributed computing using technologies like Kubernetes or “Serverless.”

Containers using tools like Docker or Singularity are essential to this process because it simplifies the operation by which you can package up, ship, and reproduce entire systems in an efficient manner. Far cheaper in both time and storage than is possible with shipping entire systems.

Although it can be tempting to get started with a massive distributed system, this introduces many challenging steps. Instead, it can be ideal to get started on your local computer! Install docker or singularity and start trying to reproduce some of your work using a container instead of your local development environment. This will present plenty of growing opportunities and prepare you for the mechanics of debugging in distributed systems. The next step is to get plugged into communities of folks who are on the same journey! There are many industry-specific communities like R in Pharma, language-specific communities like ROpenSci and Posit Community, and tech-specific communities like the Docker Community and Singularity slack channels.

Especially if you do not have support within your respective organizations, these communities can be an invaluable source of inspiration, connection, and collaboration. Just keep in mind that most contributions in these types of communities are volunteers. As a result, finding ways that you can provide value to others is essential to being a good community member and getting the outcomes you want. For instance, by helping others who are a bit behind you or doing as much pre-work as possible when asking for help, you raise the bar for the community and help provide value to others!

Getting started in even simple ways and making progress daily can provide a tremendous amount of growth over time. Although these technologies can be complicated, they are useful tools for reproducibility and provide a valuable toolset to the practitioner who is willing to learn!

Before we move on to a scaling architecture, it is important to note that the previous layers of reproducibility are still included when using tools like containers. Although containers provide isolation and reproducibility at the system level, you will still need to use tools like renv (library management) and repository management to ensure that your reproducible system is using reproducible language environments.

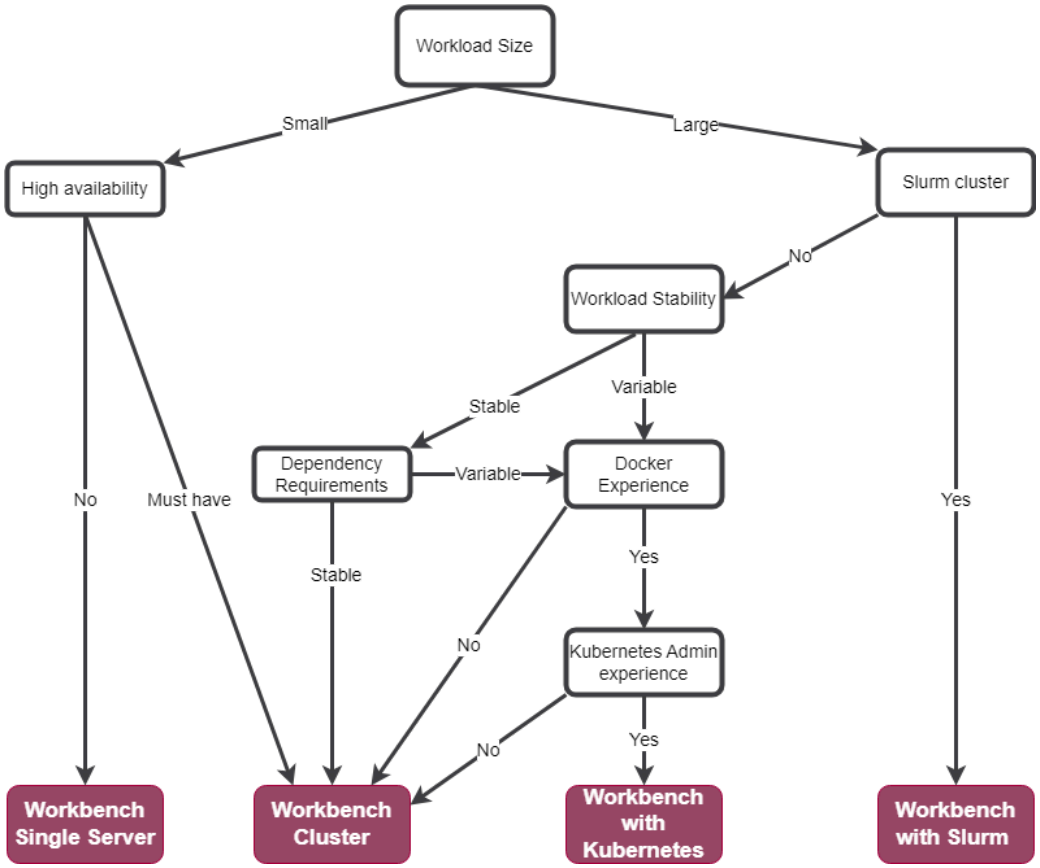
When installing packages, Posit Package Manager expects you to know which operating system to use. It is not always clear to users which operating system Posit Workbench is running on for the user. In order to tackle this the IT Admin needs to know the operating system and be able to properly set up the repository for the appropriate Linux environment to make use of package binaries. Below is script that helps define this information and other key actions such as pre-installing some packages and setting up Bioconductor etc.:

https://github.com/sol-eng/singularity-rstudio/blob/main/data/r-session-complete/centos7/scripts/run.R

Sample Images/HPC/Docker Architecture

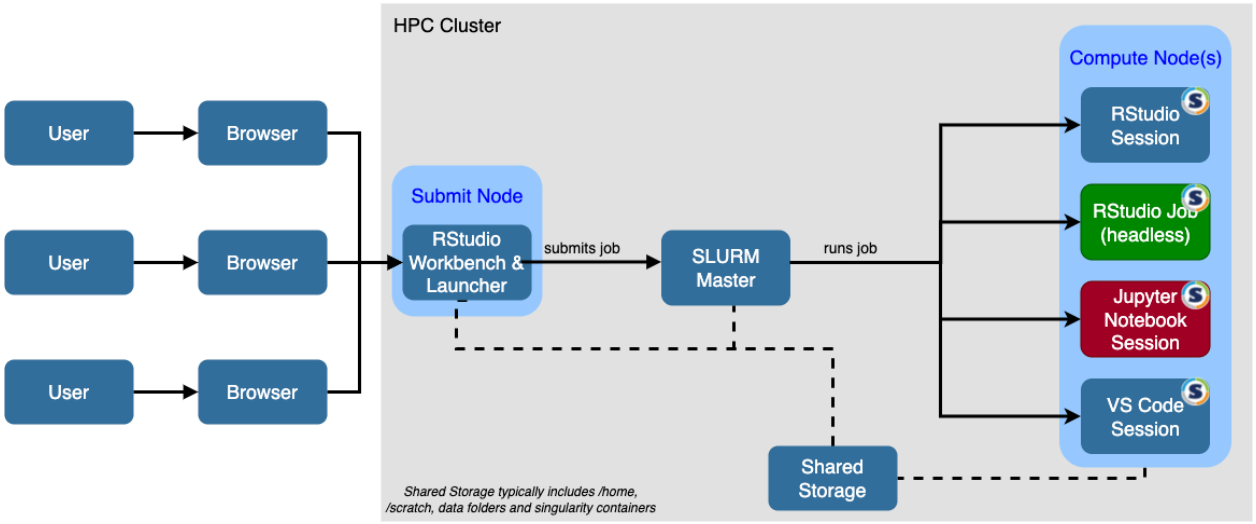
While a simplest possible architecture is recommended, often complexity is needed as scale grows. You can read more about these options here:
https://solutions.posit.co/admin-training/courses/workbench/07_scaling.html

Below are examples of scaling options:



More on these Architectures can be found here:
https://solutions.posit.co/architecting/
https://solutions.posit.co/architecting/architectures/pwb/

Slurm has been an appealing platform for clinical trial work because the functionality provides good support for clinical workflows. For example, a bulk of work can be dropped into a queue, and the work can wait in line and process as resources become available. Similar to Kubernetes, Slurm also provides container support in the form of Singularity, or its successor Apptainer as explained here:
https://github.com/sol-eng/singularity-rstudio



Another option is the Posit Workbench Launcher Plugin for Altair Grid Engine (AGE). Posit Workbench enables sessions (RStudio, VS Code, Jupyter) with large scaling needs to run on a scalable compute backend. Those backends as discussed are usually Kubernetes and/or SLURM. Other HPC schedulers supported are Altair Grid Engine (AGE) Clusters (formally SGE/UGE) as explained here:
https://posit.co/blog/integrate-posit-workbench-with-your-altair-grid-engine/

These containers are simple to create directly from an existing docker image. Environments like OpenVal can be easily created as Singularity/Apptainer containers and this is in fact the environment used for OpenVal development at Atorus. You can read more about the Slurm Launcher here:
https://docs.posit.co/ide/server-pro/job_launcher/slurm_plugin.html

Organizations can create/convert SIF (Singularity Image Format) container images via the compatible r-session-complete docker images or recipes here:
https://github.com/sol-eng/singularity-rstudio/blob/main/README.md#singularity-recipe

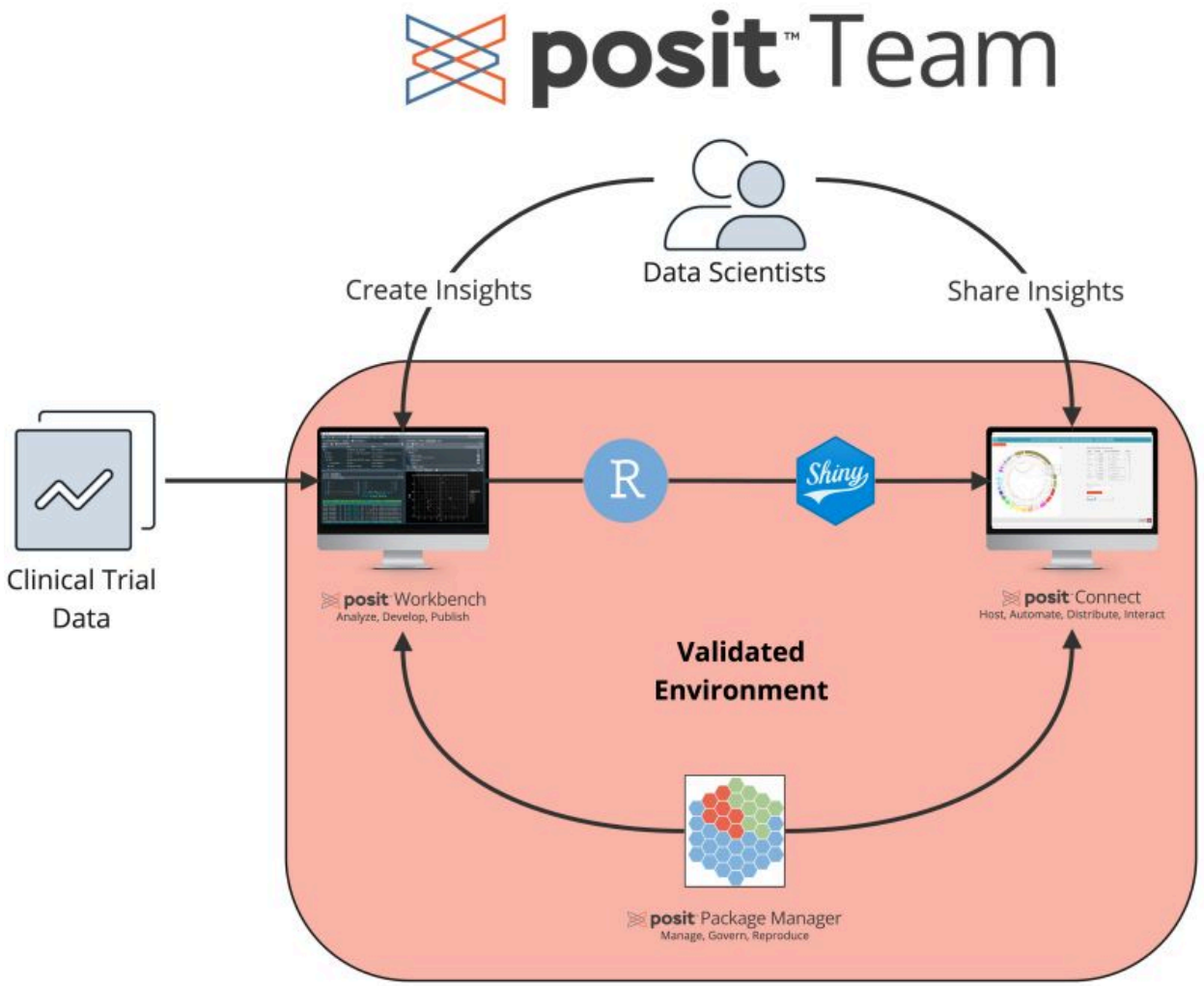
Below is a sample architecture for pharma for a mixed use case (PK/PD, clinical, data science etc.) with Microsoft Azure. An organization could easily swap this for AWS etc. While file storage can be chosen based on need, one option is Lustre, a file system for high-performance computing (HPC) and AI workloads. For large real world data, Arrow and DuckDB (https://arrow.apache.org/blog/2021/12/03/arrow-duckdb/) can be used. Sample architecture is below:

Programming Development: Posit Workbench with SLURM Launcher and HPC via Azure CycleCloud (https://learn.microsoft.com/en-us/azure/cyclecloud/overview?view=cyclecloud-8). Support for modeling and PK/PD requirements. Scale-up and down based on user need and leverage elasticity of the cloud environment in Azure.

Production Deployment: Posit Connect either in high available/load balanced Azure virtual machine setup or a AKS (Kubernetes) setup if needed to scale beyond 3-4 Azure VMs.

Package Governance: Posit Package Manager instance for GxP related work, also with high availability and hosted on AKS. Can be two Azure VMs if a smaller deployment. Surface the Validated packages from internal assessment or via OpenVal.

Below is a high level overview of of the architecture:



7. Quarto

As teams incorporate more open source languages into the clinical trials processes, support for various tools and environments are important. Quarto is an open-source scientific and technical publishing system and supports various languages like Python, R, Julia, and Observable. Quarto 1.4 was released in Jan 2024, and includes many new features that include interactive dashboards, cross-references, manuscripts and more. You can learn more about Quarto here:
https://quarto.org/

As many organizations approach interactivity into the clinical study report (CSR) and submissions, Quarto will be valuable as a reporting format that will enable reports and dashboards. These reports can be HTML, Javascript enabled and or Shiny based. You can learn more about Quarto for Pharma in the **R in Pharma** presentation “Quarto All the Things!”:
https://www.youtube.com/watch?v=k-dQ36sx4Rk

The instructors, Devin Pastoor (A2-AI), Mutaz Jaber (Gilead Sciences), and Priyanka Gagneja (OnPoint Insights) discuss various applications of Quarto in pharma like documentation, extensions, formats and reporting.

Devin Pastoor is focused on Quarto for pharma and is working on these areas:

- Word Support – Support for cross references for tables, managing changes and other pharma specific solutions solved through a combination of process and tooling.
- Reproducibility of Execution Environment Over Time – Tools such as quarto version manager (qvm) to manage multiple versions of quarto on the system for both admins and users in highly controlled environments. Tracking and launching projects, extensions, and the executing R or Python so the environment works together. Supporting projects that were started with different versions of Quarto and tracking the version of quarto over time.
- Distribution of Quarto Extensions – Tooling to consolidate and share quarto extensions traceably and centrally rather than from directly from GitHub.
- Quarto Extensions - Development to support complex report structures needed for regulatory submissions

8. Common questions / Other Notes / renv & pak & Posit Package Manager

For GxP work and submissions, organizations often use locked down and controlled sys/site/batch processes as explained above. There are tools that are also used in the clinical space like renv, RStudio projects, version control and pak that users use to support workflows and reproducibility but may also change organizational requirements and complexity for validation.

Hye Soo Cho (FDA) recently discussed using renv for reviewers as part of a FDA submission process and can be watched here:
https://youtu.be/wKG8VyZg6V0?t=3380

In the video, she raised many questions about renv. Below are important notes about renv and pak.

Renv will detect package dependency problems as if a user wants to add an old package that would break another package in the same project library. If a user starts a project with renv, the respective environment and the renv.lock file will not have package version conflicts.

Users can change the renv lock file either via direct editing or the use of function renv::lockfile_modify(). It is usually recommended not to modify or edit the lock file but rather use appropriate parameters when restoring to override some settings in the lock file (e.g. repositories). renv will use the repo defined in renv.lock but users can override those settings when restoring as seen in particular the repos parameter here:
https://rstudio.github.io/renv/reference/restore.html

Having the repo in renv to point to the latest is advised as renv documents package name and version for each package. renv then attempts to reinstall each package precisely with the same version. For historic projects, pointing to the latest for an aged project may lead to challenges if CRAN has removed the package and/or archived it. renv for archived packages will discover them. For removed packages, in order to avoid this issue, immutable Posit Package Manager snapshots are highly recommended.

Many people wonder when to use pak or renv or if they should be used together. You can learn more about pak here:
https://pak.r-lib.org/index.html

pak and renv are used for different tasks and can even be used together. pak is focused on the installation of R packages and many related tasks (like parallelization and system dependencies). renv is focused on creating isolated environments for R projects. pak can install packages from CRAN, Bioconductor, GitHub, URLs, git repos, and local files with the command: pkg_install(). The simplicity and power of pak comes from the installation plan created before

downloading packages requiring dependencies to be installed. You can specify pak when using renv via the option in the config to enable pak: https://rstudio.github.io/renv/reference/config.html

pak is a useful tool to efficiently prepare locked down environments like when making use of its parallel package installation feature when bootstrapping docker containers. Admins and users can also specify the Validated Posit Package Manager when using pak via the repo_add() function explained here: https://pak.r-lib.org/reference/repo_add.html

Below is a list of other tools that are coming up more and more in pharma within the late-stage space that would be good to be aware of:

- gt - The gt package, used for crafting attractive, customized tables in R and popular for TLFs https://gt.rstudio.com/
- Great Tables - Great Tables package, anyone can make great-looking display tables in Python: https://github.com/posit-dev/great-tables
- Shinylive - A tool for running Shiny for R and Python directly in the browser: https://posit-dev.github.io/r-shinylive/
- webR - allows you to use R in the browser – no server needed. This and Shinylive are being used in the Pilot 4 Submiion pilot here: https://github.com/RConsortium/submissions-pilot4-webR and https://docs.r-wasm.org/webr/latest/

Conclusion

This paper was to highlight strategies for managing open source reporting in late-stage environments. As interest in R for clinical trials increases by clinical statistical programmers, it is important to understand how this tooling can be used. The information outlined in this paper highlights practical approaches for IT and organizations to use when establishing a GXP environment.

CONTACT INFORMATION

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

Phil Bowsher
phil@posit.co