Paper TT05

Getting ready for R

Kieran Martin, Roche, Welwyn Garden City, UK
Second Author, Company, City, Country

ABSTRACT

In 2023 Roche PD Data Sciences is moving to a new open source backbone, with packages co-developed with other companies, and a new focus on a continually evolving set of tools that are owned and contributed to by the users. In this paper, I will discuss what that transition looks like, both the infrastructure we have built to support this new approach, and how we ensure that our talent are ready for this transition. We will look at some of the R packages we have developed, and how we are using open source as a way to drive innovation in what we do, and collaborate with our colleagues across the industry.

1. INTRODUCTION

In Roche PD (Product Development) Data Sciences, we have committed to big changes over the next few years. We will be making R our data science backbone for delivering clinical trials, and moving towards using open source technologies and open sourcing our code base where possible and appropriate.

Making these changes involves a lot of simultaneous efforts. We have built a platform, OCEAN, which is designed to support working in new ways. We have multiple teams developing R packages to ensure that we can deliver high quality results in R. Finally we have a large scale effort in supporting users through all the changes that they are going through.

In this paper I will discuss each of these topics in turn. I will explain why we have chosen to make these changes, look at OCEAN and how we designed it, talk a bit about the R packages we are using, and finish by discussing what the people transition looks like.

2. WHY CHANGE?

The decision to switch to a new framework has not been made suddenly, and is the result of gradual shifts over time. At Roche, we have always had employees who were experts in R, and wanted to use it in clinical reporting. Additionally, there were sometimes requirements that could only be done in R; novel statistical endpoints that had only been implemented in that language.

One big selling point of R was the ability to construct interactive applications in code, using Shiny ®. This capacity was something novel: allowing stakeholders to explore study data without having to request specific views from our programmers. The ability to then turn these views into outputs which could be delivered as part of a reporting event then became a need.

This growth in the need for R led to the development of an internal system, EnableR, to support the delivery of outputs produced in R to health authorities. This system also took advantage of Git, a more modern version control system, which enabled better collaboration and tracking of our work over time.

During this time, we developed internal packages to support both the creation of interactive and static outputs generated in R; the NEST team formed to deliver high quality R code.

All this history informs our decision to move towards open source languages as default. We had clear demonstrations that R could be used to deliver results, and a growing user base interested in making use of R. This switch towards open source technologies was also driven by recruitment; graduates were more likely to have skills in R and Python. Finally, we had a desire to open source our R packages.

2.1 WHY OPEN SOURCE?

We believe that open source languages present a lot of opportunities for both Roche and Pharma in general. Moving towards open source allows:

- Access to the latest developments much more rapidly, as these are often coded in R or Python first.
- The ability to switch languages and contexts more easily; open source tools tend to work together more easily, and avoiding proprietary data formats makes it easy to context switch.
- Transparency of the underlying code, which allows direct risk assessment by users

- Most importantly, the opportunity to collaborate; for more on this, see section 4.

3 MAKING THE OCEAN

The first step towards being able to deliver R end to end reporting was to build suitable infrastructure. However, the goal wasn't simply to deliver reporting in R. Instead we wished to deliver a system which:

- **Was language agnostic**: it would support SAS ®, R, Python and any future software we wished to use. We wanted to avoid programming lock in.
- **Was modular**. We didn't want the system to be tied to one particular solution, and wanted the ability to switch out any component with a new one where necessary.
- **Used cutting edge technology:** we wanted to make a system which would make developers excited to use it.

All this informed the design we arrived at. OCEAN is thus built out of multiple components, which we will discuss briefly in each subsection.

One key design philosophy for OCEAN was to deliver a system where developers could be effective, and work flexibly, but still meet our requirements when it comes to having validated, reproducible and reliable results.

3.1 COMPUTING ENVIRONMENT AND DATA STORAGE

The computing environment is built on top of AWS ®. AWS is used both to manage the compute components, as well as the data storage solution.

Users will interact with OCEAN via containers, created from Docker ® images. These images are strictly controlled, to ensure consistency in how our study teams work.

This solution allows a guarantee of reproducibility as images can be stored and recreated long after a study has delivered its final results. It also allows the progressive adding of different software over time; if a new piece of software is needed to deliver our results then we can simply make an image with that software included. This allows us to be both forwards looking, but meet our responsibilities of supporting study reporting over a long time period.

Data is stored on a file system, backed up by S3. This allows users to operate in a familiar manner, while still maintaining the ability to easily retrieve early copies of our data when needed. Additionally, we have built sophisticated solutions for automatically collecting metadata, to allow users to easily find and understand the clinical data we have collected.

3.2 CODE STORAGE

Previous solutions used in PD Data Sciences embedded code with data. This had advantages; it meant code could be directly referred to in the file system. However, it also imposed costs. It was hard to share code, and often meant that code, which was normally not required to be restricted, was treated with the same level of sensitivity as clinical data. While there are circumstances where this is necessary, it is often the case that code can be shared more widely (internally to the company) than data can.

This motivated the decision to use Git as the code versioning tool on OCEAN, with code being stored on an internal instance of Gitlab ®. All code on OCEAN will be stored on Gitlab, and then "cloned" down to a user's home repository for use. This does mark a large change for our users, possibly larger than shifting software language! I will discuss in Section 5 some of the strategies that we have used to help users with this change.

This change we feel is vital as it allows collaboration and sharing on a level not formerly possible. Another advantage of using Gitlab is that we can make use of the project management tools embedded into Gitlab, making tracking the progress of individual studies much easier than previously.

3.3 CI/CD AND DELIVERY

One other advantage of using Gitlab is that CI/CD tools can be used. For OCEAN, a customized solution has been built, on top of the tools available on Gitlab, which allows users easy access to controlled production runs, without having to worry too much about the technical implementation.

This solution currently uses Snakemake as a tool to determine how to execute user code, but we may switch to other orchestration tools in future. By providing a wrapper around Snakemake, users have a great deal of flexibility in how they deliver for a particular reporting event, but by using CI/CD we can also ensure that when users do deliver production runs they use the controlled validated tools, and that their work is reproducible.

3.4 R PACKAGES AND VALIDATION

One key consideration when designing OCEAN has been how to properly manage R packages. R packages, being collections of code, usually released open source, provide R much of its power; most of the most popular functionality in R is only there because it has been released in an R package.

This, however, presents a challenge: how can we determine if a particular R package is trustworthy? Even if we can assess an R package in some way, we probably cannot realistically do so for all possible R packages. Fortunately, we have managed to find an approach which walks the line between having a validated, stable, environment, and allowing users the flexibility to get the R packages they need to do their work.

This was solved using an in-house solution called Autovalidate. This is a framework which assesses R packages based on a number of different criteria. If a package passes all of these *and its dependencies* also pass, then it can be added to our suite of validated packages. Autovalidate also automatically generates all the documentation we need to demonstrate that we have performed due diligence for an individual R package.

If an R package fails the automated test, then it can be manually assessed, and a decision can be made if we can justify accepting the R package, or should reject it until it reaches a higher quality level. One of the benefits of this approach is that we can approach internally developed and externally developed R packages in the same way, with the only difference being that with internally developed R packages we have control to update them to meet Autovalidate standards. With external packages we can and do pass feedback to external development teams, and have seen changes based on these, but can't expect developers to necessarily have the time or inclination to make the changes we might propose.

This process being quick and easy means that new packages can be requested as and when a user might need them, and provided the R package is of sufficient quality, it can be made available within a week.

Packages are hosted on Posit ® Package Manager (PPM), which allows us to meet several needs: we can host multiple different "repos" of packages, including distinct repos for different versions of R, and unvalidated repos for non-GxP work. PPM takes snapshots over time, which allows users to refer to a particular suite of packages, which aligns with the Autovalidate process.

By hosting copies of all these packages internally, we increase our confidence in the reproducibility of our work; we are not reliant on packages remaining publically available. This is another advantage of open source software, as we are free to take long term copies of a tool without any commercial consequences.

One final piece was needed for managing R packages; we have an internal source of R packages, but now we need to be able to replicate work done over periods of time, when package versions may change. To support this we use renv, an open source R package which makes managing R packages much simpler. The renv package takes a record of all packages used and their versions, making it much easier to reconstruct what a particular analysis used.

This record provided by renv also makes it easier when it comes to submission, as we now have a clear record of which packages were used in a provided analysis.

3.5 OCEAN CONCLUSION

This section aimed to give you an overview of the components used in OCEAN. I have deliberately not gone into full detail here on each component, as the technical details of how each solution was achieved could fill a whole paper. I have also avoided discussing some of the tools and ways of working which were created to make developing in OCEAN easier, and more efficient.

With each component, we have tried to strike a balance between ease of use and providing a validated and high quality workflow. Historically validation has often gotten in the way and hindered the way developers work, and what we have managed to create with OCEAN is a solution which embodies the principles of validation, by integrating them into the way we work.

4 R PACKAGE DEVELOPMENT AND OPEN SOURCE

In this section I will discuss the three main products required for clinical reporting, SDTM, ADaM and TLGs (Tables/Listings/Graphs), as well as additional packages which meet needs outside of those products.

4.1 OPEN SOURCE PHILOSOPHY

Our goal has been to, wherever possible, open source our code base. There may be some exceptional cases where code is confidential, or, more likely, the code refers to internal data which would make it useless outside of Roche. Those cases are quite rare, and you will see in the following sections many examples of the open source efforts we are making.

It is our belief that these efforts will have outsized gains, both in the quality of our code base, and the acceptance of the usage of these tools in regulatory reporting. In particular, we have already seen increased collaboration across different companies, via the Pharmaverse. You can read more about the forming of this collection of open source pharma R tools in Section 4.3.

4.2 OAK GARDEN AND SDTM

SDTM creation is handled by a set of tools called oak garden. This is a collection of R packages, applications and databases that aim to make the majority of SDTM programming automatic, based on our internal standards.

Unlike some of the tools I will present here this is not currently open sourced; this is because the tools are heavily integrated with our internal data collection systems, so the solution would not work for external users.

Excitingly however, as of July 2023 an effort under CDISC COSA to make an open source version of this package, based on a standard which can be used across the industry has been kicked off.

4.3 ADMIRAL AND ADAM

As we began to prepare for the creation of OCEAN, we saw an exciting opportunity. At this point, no solution existed specifically for deriving ADaM in R, other than building on top of existing data manipulation tools. While we could have simply worked on an in house solution, we saw an opportunity to truly collaborate to come with a solution which works across the industry.

This led into a partnership, initially with GSK, to work on Admiral, an open source solution for generating ADaM data. This has already had a great deal of success, with collaborations to create sister packages for specific use cases:

- Admiralonco, a collaboration between Roche, GSK, AMGEN and Bristol Myers Squibb
- Admiralophtha, a collaboration between Roche and Novartis
- Admiralvaccine, a collaboration between Pfizer, GSK and Johnson & Johnson

Admiral provides a perfect example of how open source can benefit us all in producing solutions for delivering results faster. The Pharmaverse now contains not just Admiral, but many other packages for solving other problems common across the industry, such as handling XPT files, and applying data cuts consistently.

4.4 NEST AND TLG

NEST is the team with the longest history at Roche; they produced packages internally to support work before OCEAN was in development. Over time they have built a rich suite of inter connected packages, each of which fulfills specific functions. Rtables provides the ability to make flexible, high quality ascii tables, tern provides graphical and statistical layers to support specific output types and teal allows users to display these results in an interactive manner.

While initially the NEST packages were mostly in house, recently they were open sourced, and are now available for use by anyone.

4.5 STATISTICAL ENGINEERING AND SUPPORTING PACKAGES

So far we have discussed only the packages needed to produce the core products for delivering clinical reporting. However, there are many use cases not covered here. In some cases there are already R packages developed by other companies, or already existing, and in others we needed to develop them ourselves.

We have a devoted team for statistical engineering, who produce high quality R packages solving particular statistical challenges. Their creation of a new R package called rbmi, aiming to provide a novel imputation procedure which is fully deterministic, received the 2022 pharmaceutical award from the Royal Statistical Society and Statisticians in the Pharmaceutical Industry.

5. MAKING THE TRANSITION

In this section I will discuss how we are approaching this transition; I will discuss some of the challenges inherent to this process, and the methods we are using to overcome these challenges.

5.1 CHALLENGES WITH TRANSITIONING

There are a number of quite large challenges which we need to overcome as we transition to OCEAN

- There is a wide range of experience in R and Git:
  - Some people have been in the industry for a long time but have never used R or Git.
  - Some are joining and have experience in R and Git, but not in our context.
  - Some may know R and not Git, or visa versa.
- Not everyone is coming to OCEAN at the same time; we are not bringing all teams at once. Learning skills at the wrong time is usually an extremely bad use of resource.
- We need users to not just know how to use R and Git, but know how to use it well.

5.1 SELF DRIVEN LEARNING: WITH HELP!

Rather than try to run full scale in person training, which would be difficult to resource, and would not be effective, we instead initially focused on building excellent training resources which could be reviewed at any time. We created a site we called the Data Science University. This consists of a number of different learning paths on a variety of topics, including Git and R. Each path follows a particular subject, and guides the user through it. For example, one learning path introduces R, while another introduces R Shiny.

For each learning path we made multiple different learning resources available; in some cases these would be developed in house, but where excellent resources were already available online, we pointed users to them rather than recreating them from scratch.

On its own, this would not be sufficient, as users do not always know for sure what subjects they might need to learn. So we created training matrices which broke down the different subjects users might need to understand before coming onto OCEAN. We also created an app, the Data Science Profile, which through a guided questionnaire, tracked which areas a user might have a knowledge gap in.

This gave users the resources, and the path to take to fill in their knowledge gaps.

5.2 TRAINING AT THE POINT OF USE

To provide additional support where necessary, we run short, targeted training sessions at certain sites where we feel demand is sufficient. We also have ongoing drop-in sessions to support users in a more general fashion.

We have also licensed some learning products to provide more support. The aim is to make sure users feel like they have the resources they need, without wasting their time on unnecessary training.

5.3 ONBOARDING COACHES

Finally, we know when different teams will be coming on OCEAN, and can look at what support they are likely to need. This includes interviewing team members early to understand their needs, both in terms of required resources, and what kind of activities the study team is likely to be doing after they come on to OCEAN.

One key part of this process involves assigning an onboarding coach. This is a dedicated resource with experience in OCEAN, who guides them through the most difficult parts of getting on board. They are always available to answer the team's questions, and provide direct support when needed.

The onboarding coach is the first step in getting teams comfortable with working on OCEAN. Once they have gone through this initial period we wanted teams to continue feeling supported, so are working hard to build a wider support community.

This community combines our development teams and the teams who have already onboarded to OCEAN. We have brought them together using a few different mechanisms: Google ® chat rooms where questions can be quickly asked and answered, and a support forum, powered by Discourse ®, where we can build up answers to common questions over time.

CONCLUSION

The journey to get where we are at Roche has taken a long time. It has involved building an understanding of R, Git and other tools, and how they can be best used for delivering in clinical reporting.

This effort has not just involved our own department, but involved collaboration with our quality and IT departments, as we build a robust framework for using tools which are completely new in this context.

There have been countless meetings and discussions as we understand perspectives from both inside and external to the company, which has helped guide us to the solution we currently have.

One key understanding we had as we developed OCEAN was that this was merely the first step along a long journey; ultimately our goal is to revolutionize how we work. To truly see long term gains in efficiency, we expect that these are only the first of many changes to how clinical reporting is done at Roche.

Some of the most important next steps cannot be taken alone. We as an industry will need to discuss what we think the future of clinical reporting looks like, and how we can truly take advantage of these more modern techniques. I look forward to having these conversations and finding new ways we can deliver better and faster results to get medicines to patients more quickly.

ACKNOWLEDGMENTS

This paper reflects the efforts and work of a huge number of people who have collaborated to create OCEAN. Many thanks to everyone at Roche supporting the transition to OCEAN, including the OCEAN development team, Autovalidate, and the many R package developers.

CONTACT INFORMATION

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

Kieran Martin
Roche
Hexagon Place, Shire Park, Falcon Way
Welwyn Garden City AL7 1TW
Email: Kieran.martin@roche.com