Pharma SUG China 2024 - Paper 10033

Best practice of industrial code collaboration

Joe Zhu
PD Data Sciences, Roche

Abstract

The pharmaceutical industry has traditionally been known for its proprietary approach to research, development, and innovation. However, in recent years, there has been a growing trend towards open-source collaboration within the industry.

By embracing open-source collaboration, the pharmaceutical industry can benefit in several ways. It allows for the pooling of resources, expertise, and data, which can lead to more efficient and cost-effective drug development processes. It also promotes transparency and reproducibility in research, enabling faster validation of findings and the identification of potential therapeutic targets.

In practice, open-source collaboration faces many challenges, such as different project timelines and resourcing, complex code design and knowledge transfer. In this article, I will share user stories of how the NEST project team overcome these obstacles.

Our approach

New data types require new tools and methodologies and there is a need in biometrics to respond quickly to understand the ever increasing growth of data generated from clinical trials. In recent years, analysis software tools have taken huge leaps forward. Therefore, Roche PD data sciences team has seized the business opportunity to build an R-based toolkit that will allow us to meet these challenges. Since 2022, the NEST project has open sourced most of its key packages. Those are well received by users across different departments at Roche. NEST presence on conferences such as R/Pharma, PHUSE, useR has contributed significantly to its recognition and adoption within the pharmaceutical industry.

There are several important software development components we pay special attention to, that led us to the success of open-sourcing NEST package:

1. Code dependencies.
2. Backwards compatibility and reproducibility

3. Balance of software development and release cycles.

We practice the following items into our daily development activities. Recently, we also extended these practices into our external collaborations, with fruitful outcomes.

Robust testing framework

For the NEST package development, we have a set of sophisticated testing frameworks, including unittesting at the function level, checking function returns as expected. For each package, we also create regression tests, that process sets of functions, and check the end results are as expected.

Our experience with NEST package development is that we find snapshot output is very useful. For development, we snapshot all our templates, and if some upstream code is breaking something it wasn't meant to, it helps to catch (at least flag there is a problem) instantly.

CICD Automation with staged.dependencies

CICD is short for continuous integration and continuous deployment. It allows developers to automate the process of testing and deploying their code changes. By using GitHub Actions, our NEST developers set up workflows to automatically build, test, and deploy code whenever changes are made to the repository. This helps to streamline the development process and ensure that changes are thoroughly tested before being deployed to our production environments.

We use an in-housed tool, namely staged.dependencies to help manage developing new features that will require changes in multiple repositories. The staged.dependencies package simplifies the development process for developing a set of inter-dependent R packages. In each repository of the set of R packages you are co-developing you should specify a staged_dependencies.yaml file containing upstream (i.e. those packages your current repo depends on) and downstream (i.e. those packages which depend on your current repo's package) dependency packages within your development set of packages.

Apply Agile methodology

The NEST project team is made up from 4 core product teams. Each of the teams has their own engineering lead and products. We use tools like daily stand-up, kanban board, and retro to help us with planning, and managing daily workflows. The entire NEST team shares the same development cycle and leverages the shared project increment plan to identify inter-team

dependencies and risks. Thirty-three team members are scattered across 6 different time zones. Managing a global agile team is challenging – while making sure effective and efficient communication within and between teams, we also need to be mindful of team members' working hours and work-life balance.

Use cases

Code collaboration with the admiral team

Background

To be able to test and demonstrate examples on realistic data is very important, for both development work and the community. During the package development, we have been using some artificial testing data to test our packages, and demo examples. However, the “fake data” was well defined, and not sufficient to help us to catch special corner cases and identify defects of our packages. The NEST team started seeking “more realistic” public data to help us with the package testing.

As part of the end-to-end process of clinical reporting, sdtm data was created and converted into ADaM dataset, then feed into NEST packages to create tables, listings and figures. Admiral is the software package that is used to create ADaM data. Therefore, we reached out to the Admiral team, and planed to use their source data, apply to the admiral package, and feed into NEST packages for the TLG generation. At the same time, these data can also be applied by the admiral team to enrich their current vignettes in admiral packages and support robust regression tests as well.

Outcome

As an outcome of this collaboration, the admiral development team created R packages pharmaversesdtm (Mancini et al, 2024a) and pharmaverseadam (Mancini et al, 2024) packages include cached data for developers to use. The pharmaversadam, data is applied in tlg-catalog to provide examples for NEST package users, and scda.test test package to support integration test consistency. The two teams continue close collaboration to enhance the dataset with additional dataset and variables.

Code collaboration with the flextable team

Background

As part of the Roche code validation process, a key step is to check if package test coverage is over 80%. When an R package need to be validated, all dependencies of an R package need to be validated.

As a result, when we validate NEST package rtables, we also need to validated its depencency package flextable. The flextable uses a grammar for creating and customizing pretty tables. The flextable package is often used with the package “officer” to produce results in word document or powerpoint slides. Both flextable and officer are created by David Gohel and his team. The package is available on CRAN since 2017, and has had millions of downloads and wide usage by R developers. It is a relatively stable and well used package, however the test coverage of this package is below 80%.

In early 2024, our NEST team collaborated with the flextable team to improve the test coverage.

Outcome

We have contributed to the flextable project, to help creating several unittests for the package to help increasing the test coverage. These changes have been included in the new CRAN release version of flextable.

Code collaboration with Johnson & Johnson

Background

Johnson & Johnson's biometrics team planned to pilot the rtables package to support their clinical reporting events. However, there was a page split feature not yet implemented to support customizedble column split. J&J’s software development team would love to add this feature to the rtables package, with code reviews by the Roche NEST team.

Since both development teams were international. There were very limited time windows to allow all of us to meet and discuss implementation details. There were different priorities and limited time and resources for each company's software team. In order to make the collaboration more efficient and productive, our NEST team proposed the following criteria and way of working:

1. New PRs will need to pass all CICD tests and checks, including
    a. CICD pipeline of the package itself.
    b. CICD pipelines of the downstream NEST packages.
    c. Integration test, snapshot tests in the {scda.test} package.

Code reviews by the Roche NEST team, will not kick off If there are any breaking changes highlighted by the above tests. We encourage open discussion, even though the window is small.

2. If it is a complex feature, meeting to discuss implementation details

During the development process, we discovered another issue: the downstream packages CICD checks failed on forked repos. The main issue was due to the upstream dependencies specified in the staged.dependencies.yml file, upstream packages were pointing to repos within insightsengineering, yet upstream dependency changes were from another forked repo, for example, developer John Doe forked both rtables and formatters packages, and raised PR in both repos, the pull requests (PRs) in rtables (downstream of formatters) would require updates in the staged.dependencies.yml, yet we do not want to include such change into the source. Therefore, John Doe needed to raise to PRs in his folked rtables repo, where PR1 is the true PR, PR2 include additional staged.dependencies.yml updates, PR1's merge is conditional on the success of PR2's CICD check results.

Alternatively, we added developer John Doe into rtables, formatters, rlistings, tern and scda.test, and CICD pipelines were able to initiate without forking.

Outcome

As the result of this close collaboration between Roche and Johnson&Johnson, in 2024 June, we re-released four packages, including formatters (Becker and Waddell, 2024a), rtables (Becker and Waddell, 2024b), rlistings (Becker et al., 2024) and tern (Zhu et al., 2024) on CRAN. As a result of this cohort release, we updated package version in all downstream package dependencies, including teal.modules.clinical and chevron. No breaking changes were raised.

Summary

Our NEST team has successfully collaborated externally with several teams. During the process, our code development philosophy are well-acknowledged and shared with the other teams. In practice, we openly share our team capacity with external collaborators during the PI planning stage, and make feasible plans for the development cycle (8 weeks). We exercise agile methodology with external collaborators, and leverage robust testing framework to ensure package development are as planned. We use github CICD to automate tests and save develop teams valuble time.

Reference

Becker G, Waddell A (2024a). _formatters: ASCII Formatting for Values and Tables_. R package version 0.5.8, https://CRAN.R-project.org/package=formatters

Becker G, Waddell A (2024b). _rtables: Reporting Tables_. R package version 0.6.8, https://CRAN.R-project.org/package=rtables

Becker G, Waddell A, Zhu J (2024). _rlistings: Clinical Trial Style Data Readout Listings_. R package version 0.2.9, https://CRAN.R-project.org/package=rlistings

Mancini E, Gayatri G, Zhang K,Kumari P, Bundfuss S, Zhu Z, Mascary S (2024a), _pharmaversesdtm: SDTM Test Data for the 'Pharmaverse' Family of Packages, R package version 1.0.0, https://pharmaverse.github.io/pharmaversesdtm/

Mancini E, Zhang K, Bundfuss S, Gayatri G, Grassely D, Zhu Z, Mascary S (2024b), _pharmaverseadam: ADaM Test Data for the 'Pharmaverse' Family of Packages, R package version 1.0.0, https://pharmaverse.github.io/pharmaverseadam/

Zhu J, Khatri L, Rucki P, Shen V, Bove DS, Black J (2023). “Most recent update on the NEST project “ In PharmaSUG. https://www.lexjansen.com/pharmasug-cn/2023/AA/Pharmasug-China-2023-AA102.pdf

Zhu J and Bove DS (2023). “Agile team organization in pharmaceutical data science, Why we should be moving lots of small rocks rather than the one big mountain”. In PharmaSUG. https://www.lexjansen.com/pharmasug-cn/2023/LS/Pharmasug-China-2023-LS109.pdf

Zhu J, Sabanés Bové D, Stoilova J, Wang H, Collin F, Waddell A, Rucki P, Liao C, Li J (2024). _tern: Create Common TLGs Used in Clinical Trials_. R package version 0.9.5, https://CRAN.R-project.org/package=tern

Contact information

Joe Zhu
Roche
joe.zhu@roche.com