Paper: OS04

How CI/CD Enhances the Development of R Packages in the Pharmaverse

Ben Straub, GSK, USA
Dinakar Kulkarni, Roche, USA

ABSTRACT

Continuous integration (CI) and continuous delivery (CD) are playing a pivotal role in ensuring that R packages in Pharma meet the highest standards. Focus is placed on ensuring that packages are fit for purpose for both internal systems as well as the various requirements for CRAN/BioConductor. In this paper, we discuss a few best practices that were adopted into making developer-friendly and efficient CI/CD pipelines and the impact that these pipelines have had in the open-source Pharma community and at Roche/Genetech. Two case studies of package and pipelines will be discussed - one on a beginner level and one on an advanced level. The first will be CI/CD workflows for the admiral R package, used for building Analysis Data Models (ADaMs) and the second case will be regarding the NEST framework, a collection of R packages for creating TLGs.

INTRODUCTION

Both small and large software projects can greatly benefit from employing CI/CD into their workflows. A lone software developer maintaining a project over many years can benefit from continually checking that their software runs under current and new dependent packages and operating systems. A large open-source projects can benefit from continually checking that contributors code in Pull Requests/Merges meet standards set by the project. Both can leverage the growing CI/CD open-source pipelines being developed across the myriad and vibrant open-source projects.

Below we will discuss a few scenarios where CI/CD has greatly benefited the open-source R package {admiral} and {nest} framework.

{admiral}

{admiral}, ADaM in R Asset Library, is an open-source R package that seeks to build a modularized toolbox to develop ADaMs according to CDSIC standards. The package was initially started by a collaboration between GSK and Roche and has since expanded to more companies and created additional extension packages for specific disease areas. The expansion has come with growing pains, but these pains are greatly reduced by CI/CD workflows.

Below we will look at two examples of CI/CD workflows employed in {admiral}:

1. Continuously checking template code that build standard ADaMs
2. Managing multiple CI/CD workflows across multiple packages repositories

ADaM Template Codes

The community of developers building {admiral} have a strong desire to help users build ADaMs in R. This is accomplished by robust documentation for {admiral}'s functions as well as User Guides often referred to as Vignettes. However, documentation can only go so far in demonstrating the fitness of the package. Robust code examples that can build entire ADaM datasets like ADAE or ADLB are incredibly helpful to showcase the modularized approach to {admiral}. These ADaM code examples are given the name **templates** within {admiral}. A user after installing {admiral} can simply call:

```
admiral::use_ad_template("ADVS")
```

to have a fully formed R script for creating an ADVS dataset with most common variables derived using {admiral} functions.

Unfortunately, these templates are not checked when building the R package as they would violate CRAN policies around examples run-times(citation). CRAN has many checks on a package, one being that only allows examples to execute under a certain time limit. To address this limitation from CRAN, a custom CI workflow was developed to test that these templates are indeed working as intended. We discuss this workflow in the following scenario.

Scenario: Update to an {admiral} function

The following scenario is common on {admiral}'s GitHub Repository. A user has identified an issue with a function in {admiral} that does a derivation for a BDS-Finding dataset. A developer on {admiral} updates the function in a feature branch as well as updates the BDS-Findings template. The developer then initiates a Pull Request of their feature branch into the main code branch of {admiral}. Before the Pull Request of the feature branch is allowed to be merged into main, a series of CI workflows are ran and must all pass. One of these CI process is the **Check Templates Workflow**. The workflow in its simplest form, has a new instance of R installed on a GitHub server with appropriate operating system and dependent packages installed. All the template code, for each ADaM, is then unpacked from the package and run on that GitHub server. If each template runs free of errors on the server, then the Check Templates Workflow is shown to have passed.

We have provided the first 10 lines of the `yaml` file that GitHub uses to run this workflow on the {admiral} GitHub repository. The yaml file is the syntax of choice to define each workflow on the CI/CD platform on GitHub, which is called GitHub Actions.

```
---
name: Check Templates

on:
  workflow_dispatch:
  pull_request_review:
    types: [submitted]

jobs:
  templates:
    name: Check Templates
    uses: pharmaverse/admiralci/.github/workflows/check-templates.yml@main
    if: github.event.review.state == 'approved'
    with:
      r-version: "4.0"
```

A few parts to note in the file:

- `uses:` - The core workflow instructions are housed on the `admiralci` GitHub repository. Essentially, the workflow in {admiral} pulls from the repository {admiralci} to run. This is a common practice across GitHub. More will be discussed on this `admiralci` repository later in the **Common CI/CD framework** framework section and how to leverage it in our family of repositories.

- `if: github.event.review.state == 'approved'` - This workflow is only run after a reviewer of the code **Approves** the Pull Request. As these runs of the templates can be resources intensive it was decided to only run after a Pull Request is approved.
- `r-version:` The Version of R can be customized as needed in the package repository.

In summary, every time there is a need to add or update code to {admiral} a Pull Request must be initiated. For that code in the Pull Request to be successfully merged in all of the CI process must pass. One of those CI workflows is around the ADaM template code that can *NOT** be executed with the common package checks for CRAN. The Check Templates makes sure that this template code is robust and current by continuously checking the templates as {admiral} code is developed and ready for user use.

Common CI/CD framework

{admiral} has grown into multiple extension packages for specific disease areas as well as two dependency packages for testing and developer tools. Multiple companies with varying expterise of developers are actively involved on a daily basis with a shared goal of helping users to build ADaMs in R. To maintain integrity within the family of admiral a common framework of CI/CD workflows have been developed and housed in the GitHub repository admiralci. This CI/CD framework is not optional and is required to be used at a minimum. Additional CI/CD checks can be added if needed to the repository.

In each of the repositories, there is a `.github/workflows` folder and in it you will find a file called `common.yml`. This file deploys the same CI/CD workflows across all repositories regardless of the package name or purpose. The workflows are around code style, linting, spell check, link validation, creating a validation report, R-CMD checks, checking documentation is current and publishing a {pkgdown} website.

A partial glimpse is provided below that showcases four CI workflows: code style, spelling, validation and R-CMD checks. Please note, the R-CMD check are the common checks employed by CRAN to test your package. Failure of any of the CRAN tests mean rejection of your package. You can run these locally on your machine or in our case run them as CI checks!

```yaml
---
jobs:
  style:
    name: Code Style
    uses: pharmaverse/admiralci/.github/workflows/style.yml@main
    if: github.event_name == 'pull_request'
    with:
      r-version: "4.0"
  spellcheck:
    name: Spelling
    uses: pharmaverse/admiralci/.github/workflows/spellcheck.yml@main
    if: github.event_name == 'pull_request'
    with:
      r-version: "4.0"
  check:
    name: Check
```

```
    uses: pharmaverse/admiralci/.github/workflows/r-cmd-check.yml@main
    if: github.event_name == 'pull_request'
  validation:
    name: Validation
    uses: pharmaverse/admiralci/.github/workflows/r-pkg-validation.yml@main
    if: github.event_name == 'release'
    with:
      r-version: "4.0"
```

A few parts to note in this partial file glimpse of `common.yml`:

- `uses:` - Just like in the above **Check Templates** section this `common.yml` file references the workflows housed on `admiralci`. Readers are encouraged to go the admiralci repository and explore the yaml syntax for these checks.
- `if: github.event_name == 'pull_request'` - A slight deviation from **Check Templates**. This CI workflow is triggered when the Pull Request is started, which differs from when the Pull Request is Approved. This is helpful for developers as they will get feedback almost immediately if their code is meeting the admiral standards.
- `if: github.event_name == 'release'` - When a Release is initiated on a GitHub Repository this will trigger an event where a custom validation report is created for the release.
- `r-version: "4.0"` - The code style and spelling CI workflows both make use of R version 4.0, i.e. the GitHub Server that runs these checks only installs R version 4.0.

- Missing R version for Check Workflow. The check workflow does not have an `r-version` listed. This workflow installs the three latest version of R. At the time of this paper, that was 4.0, 4.1 and 4.2. A snapshot of the dependent packages closes to the release of those version of R is also installed. The package is then run on these three instances of R and if it installs properly and passes all CRAN checks, then the CI workflow is passed.

Maintaining the integrity of multiple dependent and extended packages across multiple repositories is a challenge. Introducing different levels of skills with your developers compounds potential issues. Developing a common framework of CI/CD housed in the admiralci repository has allowed us to continually provide feedback to both new and veteran developers while also ensuring that our code base is maintained to a high-level of integrity. At the same time, we have a common CI/CD framework housed in a central place has enabled us to quickly bring online additional pharmaverse packages.

{nest}

This is a main topic in the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. If you need to include source code:

```
data one;
set two;
if mix(var1, var2) > 0 then do;
```

Continuation of body – after source code.

SUBHEAD

This is subtopic for the above. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. If you need to include source code:

```
data one;
set two;
if mix(var1, var2) > 0 then do;
```

Continuation of body – after source code.

SUBHEAD

This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. This is the paper body. If you need to include source code:

```
data one;
set two;
if mix(var1, var2) > 0 then do;
```

Continuation of body – after source code.

CONCLUSION

We encourage readers new to CI/CD to also check out a workshop conducted by the authors called: An intro to CI/CD for R packages, which you can find linked in the Reference section below.

REFERENCES

- admiral
- pharmaverse
- nest
- CDISC

ACKNOWLEDGMENTS

- GSK and Roche
- Huge shout out to the developers of the pharmaverse building R packages.

RECOMMENDED READING

- Further Reading
  - GitHub Actions
  - GitLab CI
- Advanced Examples
  - r-lib/actions
  - `{admiralci}`

  - Docker
- Presentation built with Quarto
- This paper
- Presentation related to this paper
- Workshop at R/Pharma: An intro to CI/CD for R packages

CONTACT INFORMATION

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

Author Name: Ben Straub

Company: GSK

Address: 1000 Black Rock Rd

City / Postcode: Collegeville, PA 19426

Work Phone: (610) 917-3493

Email: ben.x.straub@gsk.com

Web: www.gsk.com