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 ADaM datasets and the second case will be regarding the NEST framework, a collection of R packages for creating TLGs.

INTRODUCTION

Both large and small software projects can greatly benefit from employing CI/CD into their processes. A lone software developer maintaining a project can benefit from continually checking that their software runs under current 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. 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. Below we will discuss a few scenarios where CI/CD has greatly benefited the R package {admiral} and {nest} framework.

{admiral}

{admiral} is an open-source R package that seeks to build a modularized toolbox to develop Analysis Data Models (ADaMs) according to CDSIC standards. The package was initially started by a collaboration between GSK and Roche and has since expanded to multiple companies and extension packages. The expansion has come with growing pains, but these pains are greatly reduced by CI processes.

Below we will look at two examples of CI processes employed in {admiral}:

1. Continuously checking template code that build standard ADaMs
2. Managing multiple CI processes 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}. Unfortunately, these templates are not checked when building the R package as they would violate CRAN policies around examples run-times(citation). CRAN as one of their many checks on a package only allows examples to execute under a certain time limit. To address this limitation from CRAN, a 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. 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, dependent packages are installed and then template code is unpacked and run on that GitHub server. If the code 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 use to run this workflow on the {admiral} GitHub repository.

A few parts to note in the file:

- `uses:` - The workflow is housed on the `admiralci` GitHub repository. More will be discussed on this `admiralci` later.
- `if: github.event.review.state == 'approved'` - This workflow is only run after a reviewer of the code **Approves** the Pull Request
- `r-version:` The Version of R can be customized as needed in the package repository.

```yaml
---
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"
```

Common CI/CD framework

admiral has grown into multiple extension packages for specific disease areas. The team has also created a CDISC specific data package as well as a developer tool package.

```
---
# Source: https://github.com/pharmaverse/admiralci
# Common workflows designed for Admiral
# but can be easily used by any other R package
name: admiral CI/CD Workflows

on:
  # 'workflow_dispatch' gives you the ability
  # to run this workflow on demand, anytime
  workflow_dispatch:
  # 'push' events are triggered when commits
  # are pushed to one of these branches
  push:
    branches:
      - main
      - devel
      ...
```

Continuation of body – after source code.

{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. 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

The conclusion summarises your paper and ties together any loose ends. You can use the conclusion to make any final points such as recommendations predictions, or judgments.

REFERENCES

- admiral
- pharmaverse
- nest

ACKNOWLEDGMENTS

- GSK and Roche
- Huge shoutout 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