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

PhUSE 2023, US Connect
March 7th, 2023
Ben Straub (GSK) & Dinakar Kulkarni (Roche)

< **CI-CD GitHub** >

What is CI/CD?

- Continuous Integration (CI): Frequent merging of several small changes into a main branch

- Continuous Delivery (CD): Repeatable deployment process when deciding to deploy

  CI/CD bridges the gaps between development and operation activities and teams by **enforcing automation** in building, testing and deployment of applications. CI/CD services compile the incremental code changes made by developers, then link and package them into software deliverables

< **CI-CD GitHub** >
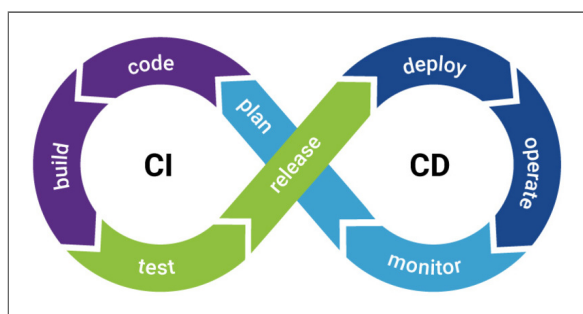
# Does it help?

...Yes! Yes, it does!!

<**CI-CD GitHub** >
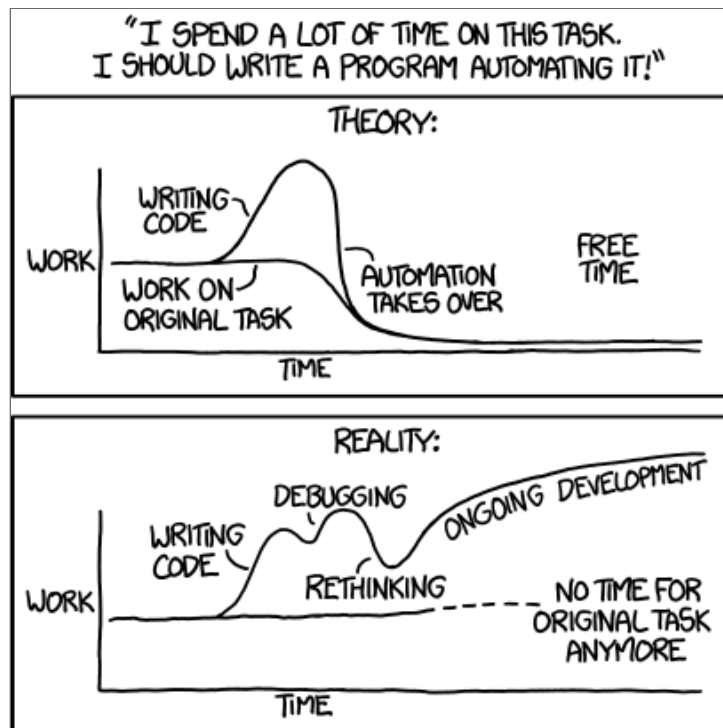
How does CI/CD help R packages?

- Catch issues (bugs) early on

- User base on multiple OSes and multiple R versions

- Faster turnaround on Code Review

- Multiple Contributors on your R Package

- Enforce style conventions and preferences

- Measure test coverage for new code

- Keep docs up-to-date

- And we can just keep going!

  We covered a lot of custom CI/CD actions for R packages in the R/Pharma Workshop in 2022: **Intro to CI/CD for R Packages** ⌕

< **CI-CD GitHub**   >

Two Case Studies
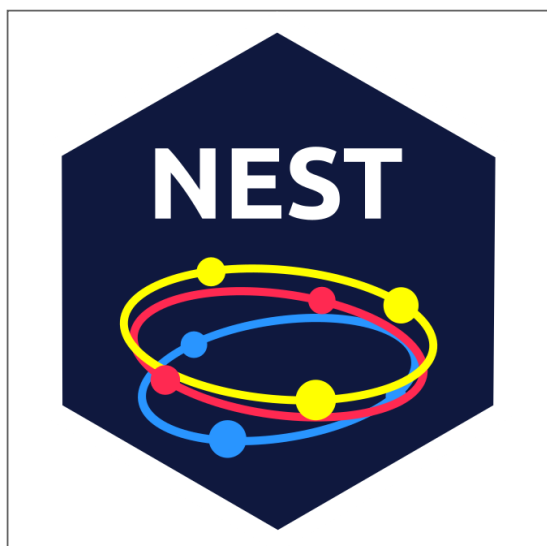
< **CI-CD GitHub** >

Case Study - admiral

< **CI-CD GitHub** >

About admiral

- Provide an open source, modularized toolbox that enables the pharmaceutical programming community to develop ADaM datasets in R.

- ADaM is one of the required standards for data submission to FDA (U.S.) and PMDA (Japan) for clinical trials

- Links

    - **CDISC** ↗

    - **https://github.com/pharmaverse/admiral** ↗

- **Issue 1:** Checking ADaM Template code

- **Issue 2:** Common CI/CD workflows for the `admiral` family of packages

< **CI-CD GitHub** >
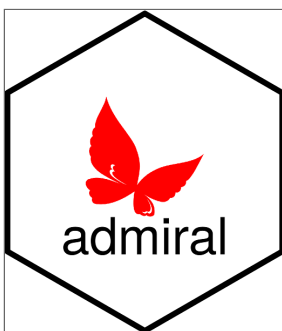
Issue 1 - How to Check our Template Code

- Create a reference files to build common ADaM datasets that shows users how to implement our functions

- Way less text than a Vignette - Code is ready to go and build a dataset

- Where we store this code is not checked by R-CMD

- How to ensure code stays up to date with deprecated functions or unforeseen bugs get in from functions working together?

- CI/CD for the win!

<**CI-CD GitHub** >

< CI-CD GitHub >
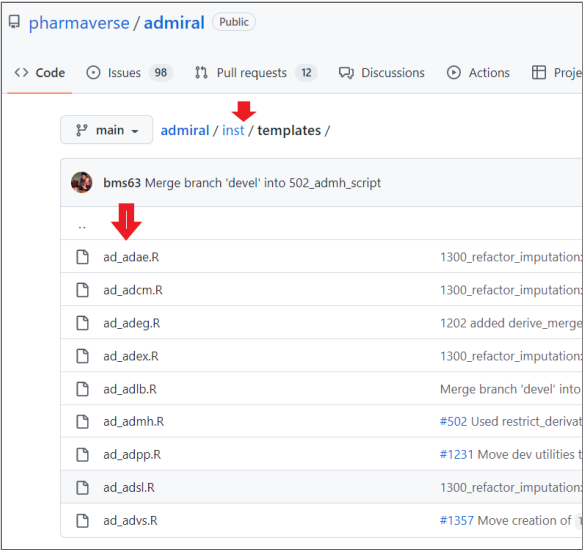
Solution 1 - CI/CD for Templates

- Dedicated CI/CD workflow that executes the Template code

- Once a Code Review is completed the `Check Template` Workflow is executed

- If any errors or warnings are detected the CI/CD check fails and the contributor must fix the error or warning.

   .github/workflows/check-templates.yml

< **CI-CD GitHub** >
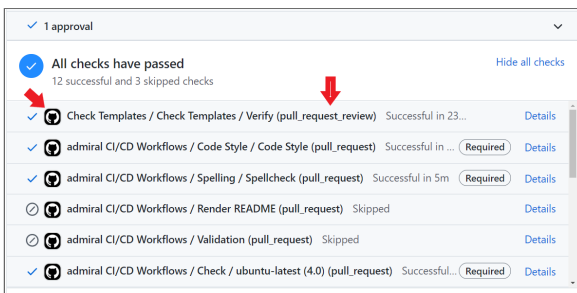
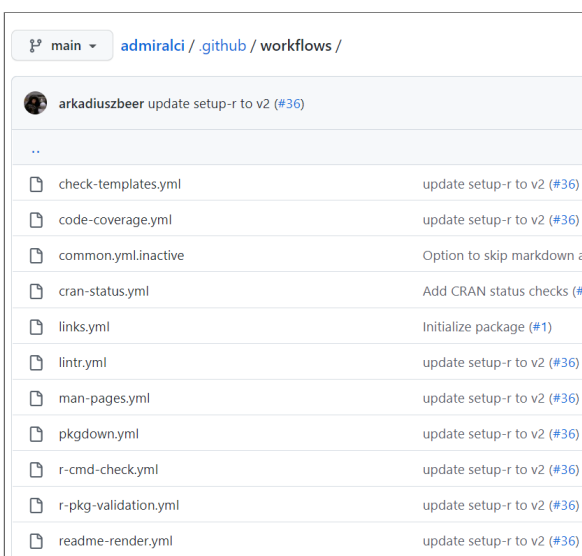# Issue 2 - admiral upstream and downstream dependencies

- As you can imagine there can be a lot of different types of ADaMs!

- Extension packages focus on specific disease areas like oncology

- The `admiral` family has a package for developers, template R package repo and dummy data

- Eek!! How to keep this all in line!

<**CI-CD GitHub** >

<CI-CD GitHub >

Solution 2 - Common CI/CD workflows for admiral upstream and downstream dependencies

- Using `{admiralci}`, we have a common set of CI/CD workflows

- Developers moving between packages are familiar with these workflows

- Common documentation between packages for CI/CD workflows - easy to maintain and provide to new contributors

< **CI-CD GitHub** >

Case Study - NEST

< **CI-CD GitHub** >

About NEST

- A collection of R packages for creating TLGs/TFLs and exploratory clinical trials data visualization

- `tern` and `rtables` for creating TLGs

- `teal` for creating exploratory web applications for analyzing clinical trial data

- Links

  - **rtables** ↗

  - **tern** ↗

  - **teal** ↗

< **CI-CD GitHub** >

Use Case 1 - Integration Testing

- An in-development package must be tested against the latest versions of upstream dependencies

- Monorepo emulation via a git branch naming strategy is achieved by using

    - the **staged.dependencies R package** ⬈

    - and the **staged.dependencies GitHub Action** ⬈

- Testing as a cohort can be done at any stage (eg. development, pre-release, release)

<**CI-CD GitHub** >

Use Case 2 - Web Application Testing & Deployment

- Analysts create Shiny web apps via the `teal` framework for analyzing data

- Apps are tested via a CI pipeline that uses the **shinytest2** ↗ R package

- Apps deployed to an Posit Connect Server instance via a CD pipeline

  - With the help of the **rsconnect** ↗ and **connectapi** ↗ R packages

< **CI-CD GitHub** >

< **CI-CD GitHub** >

Use Case 3 - Validating R Packages

- R packages are validated by an internal validation team that uses CI/CD pipelines to automatically

  - accept new package submissions via a form

  - running tests against the new package to ensure package integrity

  - enforcing criteria to ensure that the package meets regulatory requirements

- Also validated externally via an open source project called **thevalidatoR** ⬈

< CI-CD GitHub >

Additional Materials

- Further Reading

  - **GitHub Actions** ⬀

  - **GitLab CI** ⬀

- Advanced Examples

  - **r-lib/actions** ⬀

  - **`{admiralci}`** ⬀

  - **Docker** ⬀

- Presentation built with **Quarto** ⬀

- **R/Pharma 2022 CI/CD Workshop** ⬀

- **This Presentation** ⬀

< **CI-CD GitHub** >