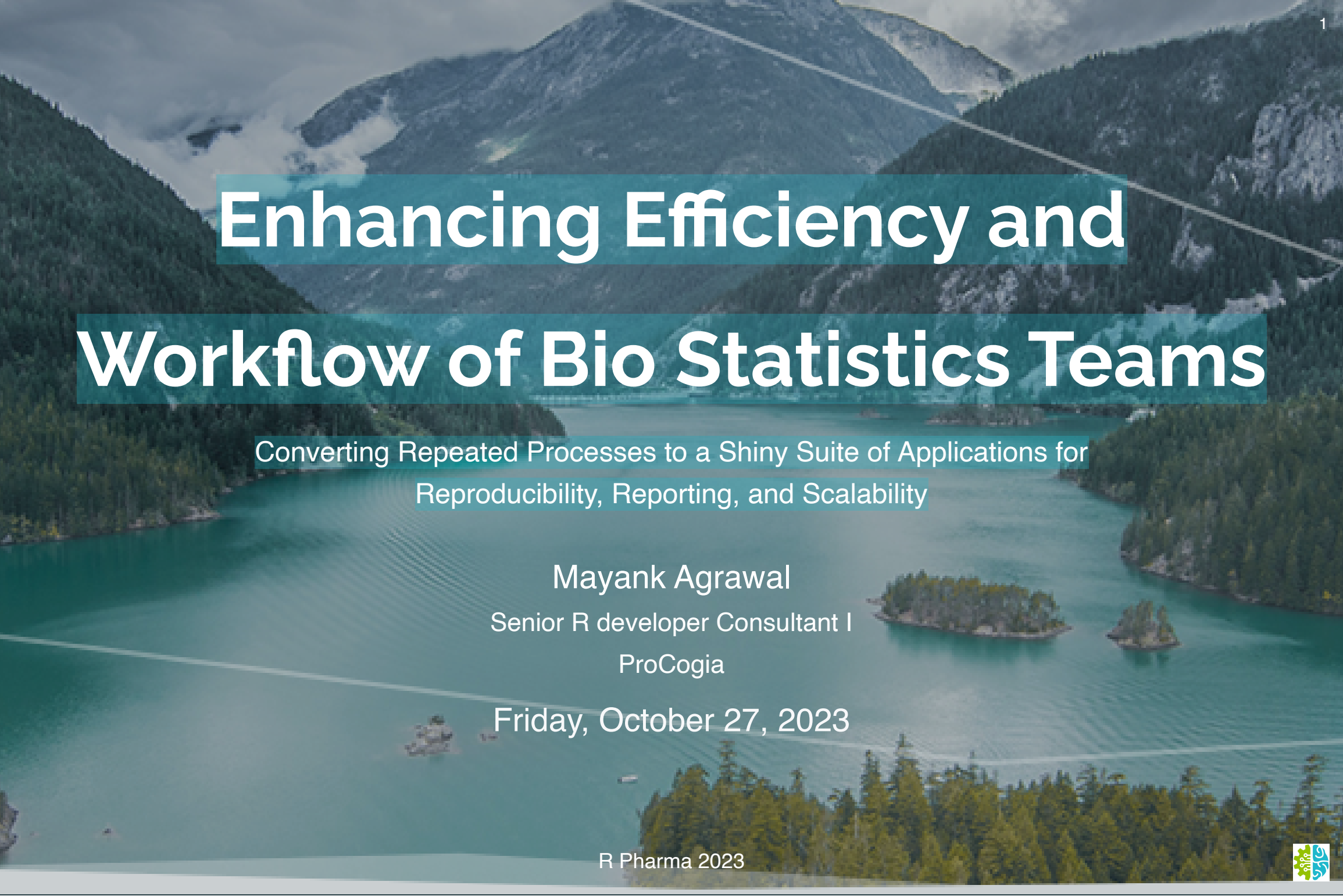

Enhancing Efficiency and Workflow of Bio Statistics Teams

Converting Repeated Processes to a Shiny Suite of Applications for Reproducibility, Reporting, and Scalability

Mayank Agrawal

Senior R developer Consultant I

ProCogia

Friday, October 27, 2023

Phase 2 Trails

> *A Phase 2 trial answers the question, "Does Drug X improve Disease Y?"*

- Phase 2 clinical trials assess the safety and efficacy of a new drug or drug combination for a specific medical condition.
- **Goal**: Determine appropriate dose and treatment plan for Phase 3 testing.
- **Phase 2a**: Involves fewer patients, generally 100-300 patients to focus on dose-response relationships and optimal dosing frequency.
- **Phase 2b**: Rigorously assesses drug's effectiveness in disease treatment, prevention, or diagnosis.
- Assess **therapeutic effectiveness** in a specific patient group for potential Phase 3 study.
- Also used to assess and review **safety parameters** for **potential adverse events** that might have been missed in a particular patient group.

Challenges

Dose Simulations for Phase 2 trails are often complex and time consuming with repeated similar workflow steps for each new variation of the dosage trail.

This often leads to

- Delayed Analysis and reporting
- Longer study time (years)
- Delayed Time to Market

resulting in

- Hindered workflows
- Lower productivity
- Repeated boring processes

Key Issues

- Manual processes
- Lengthy simulation times
- Scalability constraints
- Limited collaboration
- Reporting challenges
- Flexibility
- Reproducibility

Solution

Empowering Biostatisticians with R Shiny suite of applications

- Collaborated with Biostats to understand their pain points and challenges.
- Developed a R Shiny application for Phase 2 trails of dosage simulation.
- Fully scaled through an iterative process of Agile development and feedback.
- Created multiple R Shiny apps targetted to specific workflows.
- Developed an ecosystem of Biostatistics R Shiny applications.
- Automated and improved reporting.
- Enhanced workflow efficiency and productivity among biostats team.

What was the journey to success?

Migration Process Steps

- Define scope of each process.
- Document repetition rate, importance, and time investment for each work request.
- Identify the most time-consuming, yet simplest workflow.
- Develop a **MVP** (Minimum Viable Product)
  - Showcase a demo with the smallest workflow.
  - This aids leaders in visualizing the impact of approval.
- Integrate workflows incrementally from small to large.

Why Shiny?

- Helps to build interactive web apps straight from R.
- Shiny offers dynamic filtering, enabling instant analysis and visualization of data.
- Shiny is compatible for generating Tables, Listings, and Graphs (TLGs).
- Enables efficient visualization of complex clinical trial data.
- Allows for easy exploration of various dosage scenarios.
- Supports seamless collaboration among trial stakeholders.
- Enables real-time updates, crucial for adapting to evolving trial needs.
- Its interactive features enhance decision-making and adaptability for evolving trial needs in Phase 2 trials.
- Shiny has a quick to moderate learning curve.

Principles of Application Design

- Showcase early application design outlines using draw.io for UI layout prototyping.
- Establish a standard application template layout using either Shiny Dashboard, bslib and/or bs4Dash.
  - Create custom R function wrappers based on organization theme and color layout.
  - Re-use the template for new builds and ideas to maintain consistency and coherence.
- Define the **flow of reactivity** for the overall application/dashboard.
  - **Waterfall Reactivity Model**.
  - Avoid **Reactivity Spaghetti Mess**
  - Leverage reactive dynamic UI elements based on user input to enhance interactivity and responsiveness.

Principles of Application Design

- Create smaller, independent **Proof of Concepts (POCs)** for new feature requests.
- Prioritize user-friendliness: if it's not intuitive, it won't be used.
  - Set default selections for input widgets to make application exploration and understanding easier for users.
  - Incorporate tooltips, notifications, sectional write-ups and clear instructions to guide users in using the application.
- Facilitate Consistent Reporting:
  - Utilize parameterized Markdown/Quarto reports for dynamic MS Word report generation within the application.
  - Adhere to the organization's document template for uniformity in reporting.
  - Ensure each page or process has a predefined standard write-up with dynamic bits/sections based on simulation calculations.

Streamline Workflow

- Use RStudio Projects
- Version Control with Git
  - Use Issues, Pull Requests and Connected Commits for efficient and effortless parallel development.
  - A picture speaks a thousand words. If it can be explained with a **screenshot**, don't shy away.
- Organize your project structure
  - Group related files together for easy navigation.
  - Create, adopt and enforce a standard development template for easier developer onboarding.
  - Have a project template repository for initial cloning.

```
> fs::dir_tree(recurse = TRUE)
.
├── R
|   ├── function_calculate_roi.R
|   ├── module_file_upload.R
|   ├── server_main_page.R
|   ├── ui_main_page.R
|   └── utility_functions.R
├── code
|   ├── Intro.Rmd
|   ├── app_manual.Rmd
|   ├── download_handler_steps.md
|   ├── fetch_api_data.py
|   └── tab1_description.Rmd
├── data
|   ├── data.csv
|   └── generated_data.csv
├── global.R
├── references
|   ├── authentication_options.txt
|   ├── dynamic_ui.txt
|   └── poc_dynamic_ui.R
├── server.R
├── shiny-dir-tempalte.Rproj
├── ui.R
└── www
    ├── logo.png
    ├── org_logo.svg
    ├── shiny.css
    ├── shiny.js
    ├── shiny_bottom.js
    └── shiny_custom_template.R
```

Ensuring Reproducibility

- Leverage package management tools
  - Utilize `renv` or `packrat` to manage package dependencies.
  - Provides a controlled environment for your project.
  - Ensure reproducibility and minimize version conflicts.
- Rely on packages published and actively maintained on **CRAN** for a validated R environment.
- Establish unit tests for all functions used to ensure accurate and consistent results.
- Invest time to create an in-depth GitHub ReadMe with the sections **Project Overview**, **Getting Started**, **User Application Flow and Usage**, **Key Programming Concept Implemented** (if any) and **Developer Guide** providing comprehensive project reproducibility instructions.

Enhancing Scalability

- Implement **modularization** and **functional programming** for a **plug-and-play** development format across applications.
- Enable multiple studies to be added concurrently using standard **git branching strategies**, involving multiple concurrent developers.
- **Async Programming**: Evaluate longer simulations in a **separate R process** preventing app performance issues.
  - Few R packages to aid this are: callr, mirai, crew, coro, future and promises
- Take a step further and deploy simulation functions as internal APIs with **Plumber**.
- Write your **custom JavaScript and R bindings** for implementing unique feature requests.
- Approach feature requests as a blend of **web development**, **software engineering**, and **R development**.

Utilize Automated Testing

- Writing Test Cases (Inputs, Expected Outputs)
- Full Stack Testing
  - testthat for back-end testing,
  - shinytest2 for front end testing, and
  - shinyloadtest for load testing.
- Types of Tests: **Unit**, **Functional**, **Integration**, and **End-to-End**
- Continuous Integration for Testing with Git branching strategies.

- Benefits
  - Early bug detection
  - Efficiency and speed
  - Consistent and repeatable testing
  - Increased test coverage
  - Regression testing capabilities
  - Greater confidence in release stability

Common Application Features

- Add and execute multiple simulations simultaneously.
- Compare similar (static and interactive) graphs side by side for comprehensive analysis.
- **Downloadable** visualization, simulation calculations, and dynamic FDA submission format reports across the application.
- Display mathematical equations for each study using **LaTeX**.
- Receive **email notifications** for progress updates along with attached reports.
- Receive **notifications** upon process completion.
- Introduce **Helper Tabs** for - Application Information, Usage Manual, Release Notes, System Information, User Feedback and Contact Business Lead - all through the application.

Enhancing User Adoption

- Create multiple **GIFs** showcasing the application layout and user flow.
  - Include them in the Git readme and announcement emails.
- Create detailed application interaction **user manuals** with screenshots and highlights for each step.
- Conduct regular (quarterly) **training sessions** to provide guidance, answer questions, and assist users with new features.
  - Record and share them for easier re-visit.
- Continuously engage user base for better ROI and on boarding.
- Prioritize most requested user features for each sprint.

Thank you

- Slides available on GitHub Pages at https://bit.ly/r-pharma-2023
- Quarto presentation code available on GitHub at https://bit.ly/github-r-pharma-2023
- Connect and/or send me a DM for a follow up question or catch up
  - Linkedin: mayank-agrawal-7jan
  - X (previously Twitter): mayank7jan
  - Mastodon: mayank7j

References - R Packages

- shinyDashboard, bslib, bs4dash for standard dashboard template.
- rmarkdown and Quarto for parameterized reporting.
- renv for package management in a R project.
- glue for interpreted string literals for dynamic reporting.
- callr for separate r sessions.
- plumber for API creation.
- httr2 for API calls.
- pins for shareable secured publishing of data, models, and R objects
- testthat, shinytest2 and shinyloadtest for testing.
- dplyr for data manipulation.
- ggplot2, plotly and echarts4r for visualization.
- profvis for code profiling and time estimation