From Methods to Maintainability:

The Statistical Software Revolution in the Pharmaceutical Industry and Minimum Viable Good Practices for High Quality Statistical Software Packages

Alessandro Gasparini, PhD
E-mail: <**alessandro.gasparini@reddooranalytics.se**>

Red Door Analytics

Red Door Analytics

Disclaimer:

Views are solely my own and do not express the opinions or views of Red Door Analytics AB or the openstatsware working group.

Introduction

This is a presentation in two acts:

1. The statistical software revolution in the pharmaceutical industry
2. Minimum viable good practices for high quality statistical software packages

ACT 1:
The Statistical Software Revolution in the Pharmaceutical Industry

Red Door Analytics

Background

- Challenges with productivity, efficiency, and innovation in the pharmaceutical industry
- Flat number of new drug approvals per year, but increasing R&D spend per approval
- Several potential reasons:

1. New, high-risk therapeutic areas
2. Advances in technology leading to a wider range of potential targets
3. Increased competition and regulation

Delivering R&D value

Methodological innovation is required due to:

- New data modalities and endpoints
- Higher volume of data
- Novel designs
- […]

In this context, open-source software development has emerged as an agile solution for keeping pace with advances in statistical methodology.

Why open-source statistical software?

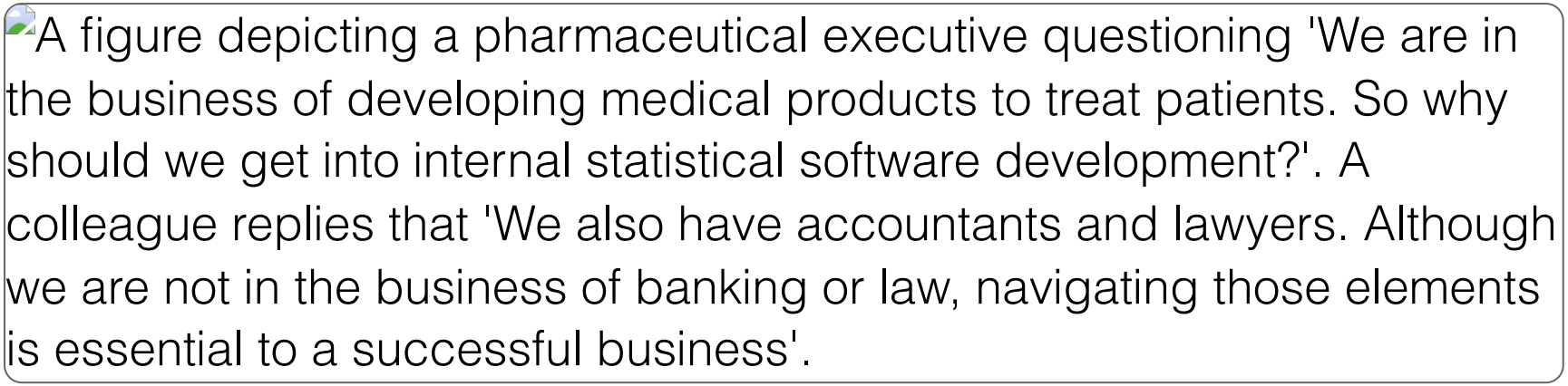
- Speed:
- Implementation of innovative designs and statistical methods
- Collaboration with academia and between companies
- Shared experience with potential bugs and edge cases
- Rapidly evolving needs

Why open-source statistical software?

- Scale:
- Sharing the burden of developing, maintaining statistical packages
- Joint, faster adoption across the industry
- Developing talent and capability

Why open-source statistical software?

- Transparency:
- Valuable for audits
- Valuable for regulators
- Valuable for further developments


A figure depicting a pharmaceutical executive questioning 'We are in the business of developing medical products to treat patients. So why should we get into internal statistical software development?'. A colleague replies that 'We also have accountants and lawyers. Although we are not in the business of banking or law, navigating those elements is essential to a successful business'.

Challenges

- **Regulatory and validation hurdles:**
- Documentation, versioning, testing, and traceability requirements for submissions
- The R Validation Hub is working on developing validated systems fit for submission to FDA/EMA

Challenges

- Organizational and cultural barriers:
- Proprietary mindset, reluctance to share code due to perceived IP risk
- Departmental silos, creating communication gaps between departments (e.g., IT and statistics)
- Purpose silos, different software packages for different tasks
- Project silos, lack of reusable/general code shared across projects

Challenges

- Technical quality and usability risks:
- Usability gaps
- Poor documentation and inconsistent interfaces may limit uptake
- Cross-language and interoperability issues for multi-language teams

Challenges

- Software quality:
- Lack of testing
- Lack of reproducible workflows
- Increased need for professional research software engineers

Challenges

- Sustainability:
- Maintenance risk, single-author projects can stagnate without funding or contributors
- License complexity (in a legal sense)

The way forward

- **Risk-based validation:** align package risk level with validation effort and documentation
- **Usability first:** API design, examples, and user testing as core deliverables
- **Break silos:** cross-functional governance and shared infrastructure teams

The way forward

- **Create RSE roles:** embed software engineering expertise in biostatistics groups
- **Develop engineering standards:** version control, CI, automated tests, release management
- **Leverage the community:** openstatsware, pharmaverse, and other cross-company initiatives

Key takeaways

1. The open-source revolution in the pharmaceutical industry is in full swing
2. This is exemplified by successful submissions by Novo Nordisk and Roche (among others)
3. Open-source software offers faster innovation, transparency, and shared efficiency if paired with governance, validation, and good software engineering practices

Illustration from DOI: **10.1016/j.drudis.2026.104613**

ACT 2:
Minimum Viable Good Practices for High Quality Statistical Software Packages

Red Door Analytics

Minimum set of good practices

- Documentation
- Vignettes
- Tests
- Functions
- Style
- Life cycle

> Developers Value Tests For Software Longevity

Documentation

Documentation is important for users and developers to understand all objects in your package, without reading and interpreting the underlying source code.

- Use inline comments next to functions, classes and other objects to generate their corresponding documentation
- Do document internal functions and classes for maintenance by future developers
- Add code comments for ambiguous or complex pieces of internal code

Vignettes

Vignettes are documents that complement the documentation by providing a comprehensive and long-form overview of the package from a user perspective.

- Provide an introduction vignette that introduces the package to new users
- Include code examples and automatically compile the vignette to ensure reproducibility
- Include deep dive vignettes that go into depth on specific use cases, functionalities or underlying theory
- Host your vignettes on a dedicated website

Tests

Tests are a fundamental safety net and development tool to ensure that your package works as expected, both during development as well as on user systems.

- Write unit tests for all functions and classes in your package, to ensure that all building blocks work correctly on their own
- Write functional tests for all user-facing functionality, to ensure that the package is stable when refactoring internal code
- Ensure adequate coverage of your code

Functions

Function definitions should be short, simple and enforce argument types with assertions.

- Write short functions for a single and well-defined purpose, with few arguments, and low complexity
- Use type hints to explain to users which argument of the function expects which type of input
- Enforce types and other expected properties of function arguments with assertions
- Catch errors and fail early with human-readable error messages

Style

Code style is important because it makes software easier to read, safer to change, and more consistent for everyone who works on it.

- Use idiomatic code and follow *clean code* rules
- Use a formatting tool to automatically implement a consistent and readable code format
- Use style checking tools to enforce a consistent and readable code style

Life cycle

Life cycle management is simplified by reducing dependencies, and should include a central code repository.

- Reduce dependencies to simplify maintenance
- Only depend on other packages that you trust
- Give clear information on user-facing changes in the package, and first deprecate functionality before removing it
- Use a central repository for version control, collecting and resolving issues, and managing releases

Resources

- `openstatsware` website:
  https://www.openstatsware.org
- Current version of our `openstatsguide`:
  https://www.openstatsware.org/guide
- *Good Software Engineering Practice for R Packages* course material:
  https://openstatsware.github.io/shortcourse-iscb2025/

Check our website: **reddooranalytics.se/career**

DOI: **10.5281/zenodo.13882307**

Speaker notes