

Leveraging on R Packages for HTA Analysis & Reporting: Our Developer Journey

Paper OS05

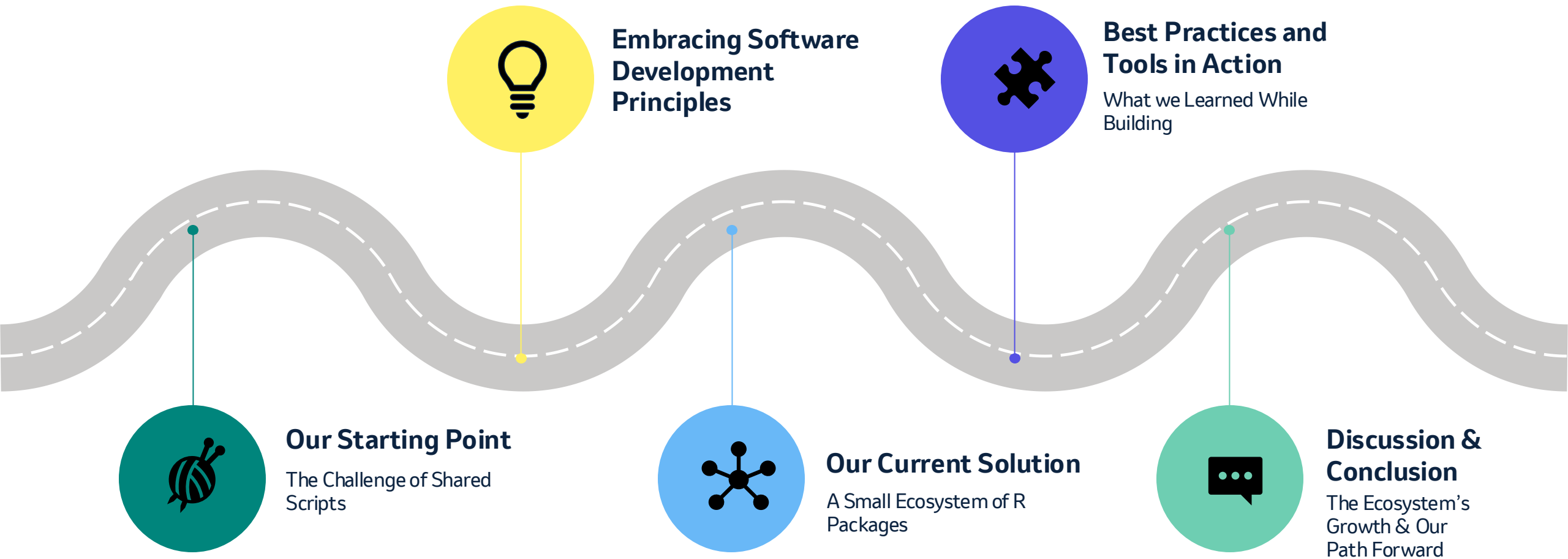
Jean Muller, MSD, Paris, France
Wenyu Li, MSD, Zurich, Switzerland
18 Nov. 2025



Disclaimer

- The content of this presentation reflects the **developer's experience and perspective** on building and using R from within the statistical programming department in collaboration with HTA statisticians @ MSD.
- This work was made possible by the robust infrastructure and guidance provided by our organization, as well as the contributions of the external open-source community.
- The **views and opinions expressed are our own** and do not necessarily represent those of our employer.

Agenda





Our Starting Point

Setting the scene

What is Health Technology Assessment (HTA)?



Health Technology Assessment (HTA)

What it is: The systematic evaluation of the properties, effects, and/or impacts of a health technology (e.g., a new drug)

Purpose: To inform decision-making in healthcare. Is a new drug effective, safe, and does it provide value for money?

Key Bodies: NICE (UK), G-BA (Germany), CDA-AMC (Canada), EUHTAR



Analysis and Reporting (A&R)

What it is: From data to insights. It involves generating the evidence required for the assessment.

Analysis Examples:

- Descriptive Statistics
- Survival Extrapolation
- Network Meta-Analyses

Output: A detailed report (the "submission dossier") containing all this evidence.



The Statistical Programming Analyst (SP)

What they do: Translate complex statistical analysis plans into robust, reproducible deliverables using codes.

Core Responsibilities:

- Build project knowledge and understanding of analysis needs
- Develop the code to perform the analyses.
- Produce the tables, listings, and figures for the report.
- Ensure the evidence is of high quality, accurate, validated, and generated efficiently.



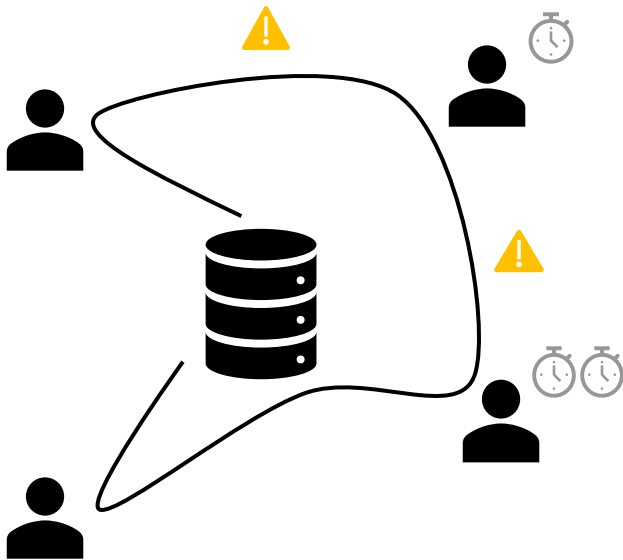
MSD

CDA-AMC: Canada's Drug Agency; EUHTAR: European Union Health Technology Assessment Regulation; G-BA: Gemeinsamer Bundesausschuss; NICE: National Institute for Health and Care Excellence

A World of Shared Scripts

The Hidden Risks of "Copy-Paste" Analytics

Situation



Risks

- **Inconsistency:** Different script versions produced different results
- **Uncertainty:** Which script has been thoroughly tested and should be used?
- **Inefficiency:** Analysts spend time finding and debugging code for issues that were potentially already solved

We Are Professional Statistical Programmers

Looking for Better Alternatives

	
Collection of large scripts, use source()	One package per major analysis
Use sessionInfo()	Use renv-like snapshots
Manual Quality Checks in a separate script	Automated testing, testthat package
Manual documentation	Embedded documentation with roxygen2 package

Good Practices

- **Modularity:** Break down big problems into small, focused packages
- **Reproducibility:** Ensure analyses are reliable and can be run by anyone, anywhere.
- **Validation:** Prove that our code is correct through automated testing.
- **Collaboration:** Make code easy to review, improve , and share.

The Limits of Our Initial Approach

The Core Problem: Scripts are Not Software

Shared Scripts	
Manual Version Control	✗
No Integrated Testing	✗
No Clear Ownership	✗
Difficult to Manage Dependencies	✗

Software (R Package)	
✓ Built-in Versioning	
✓ Includes Testing & Validation	
✓ Clear Maintainers	
✓ Reproducible Environments	



Embracing Software Development Principles

Formal Software Development Principles Through the R Package Format

Build for Scale and Quality



1. Don't Repeat Yourself (DRY)
2. Test Driven Development
3. Documentation is a feature

The Infrastructure and Tools to Support Us

Modern project management and automation infrastructure as the final piece of the puzzle



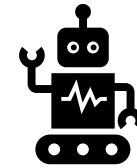
Source Code Management & Collaboration

Pull Requests (PR)
Branching
Visibility



Project & Task Management

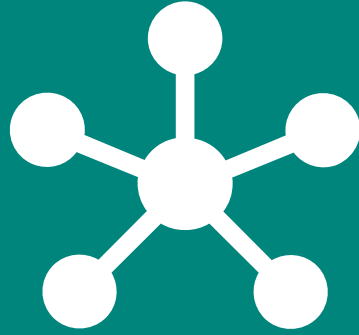
Transparency
Traceability
Roadmap



Automation / CI/CD

Continuous Integration
Quality Gate

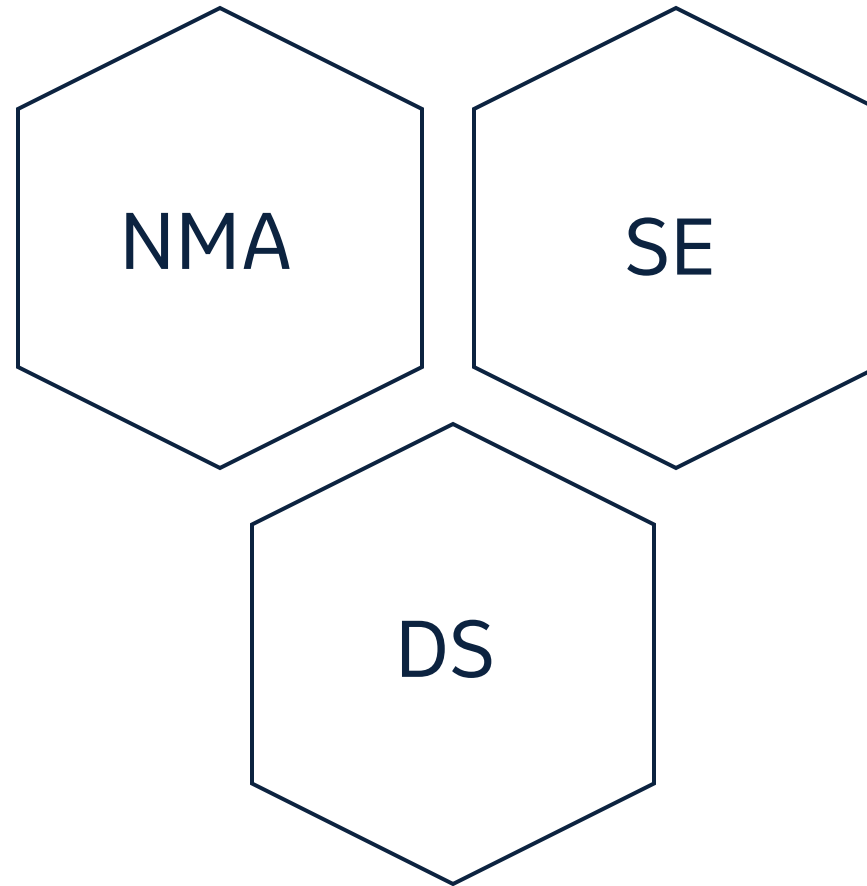
CI/CD: Continuous Integration and Continuous Delivery/Deployment



Our Current Solution

The Result: Small R Package Ecosystem

Collection of formal, versioned, and validated modules that work together



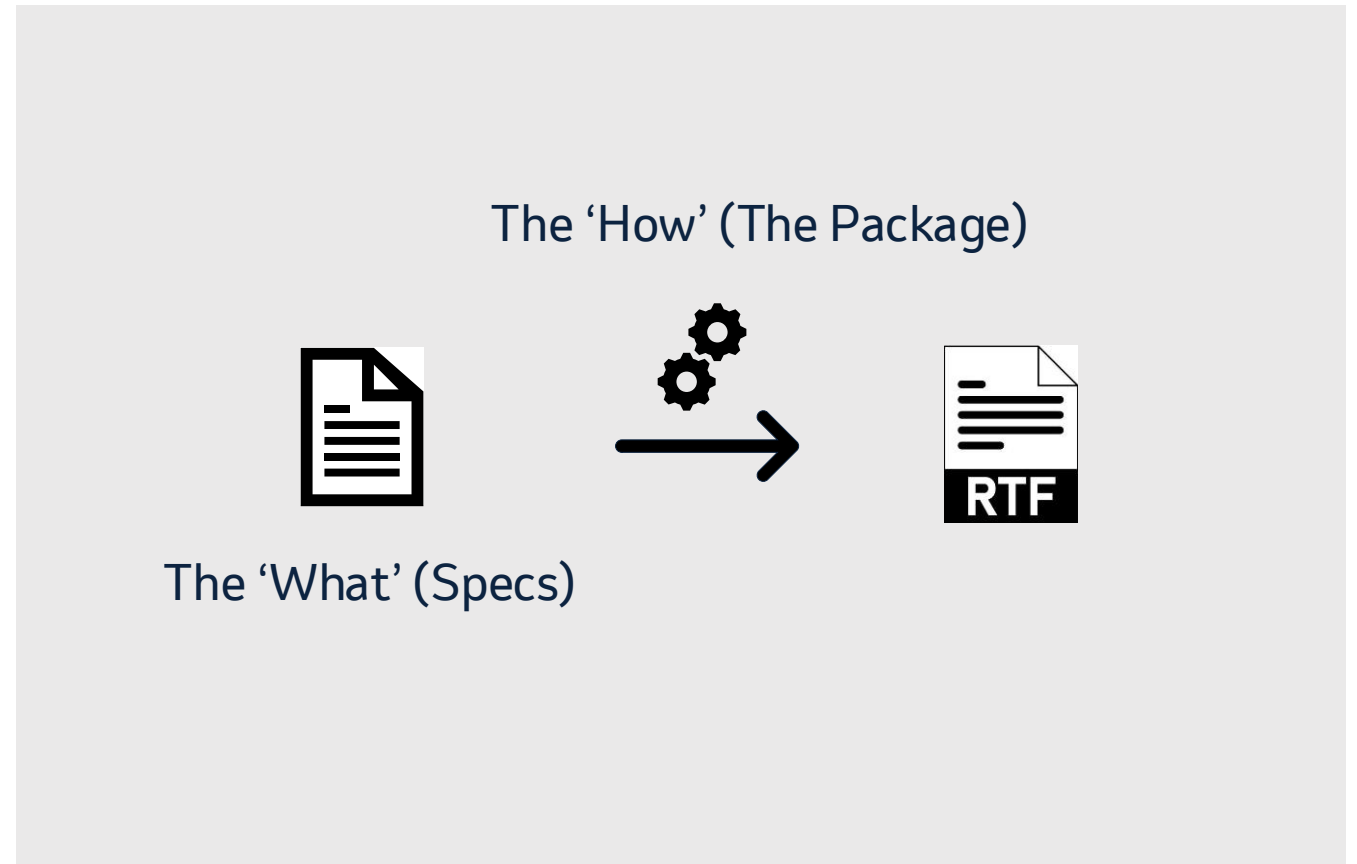
DS: Descriptive Statistics; NMA: Network Meta-Analysis; SE: Survival Extrapolation

Note: Package names are for illustration purposes only.

DS: Descriptive Statistics (Overall)

How do we translate a statistician's complex table request into a pixel-perfect, validated report without them writing a single line of R code?

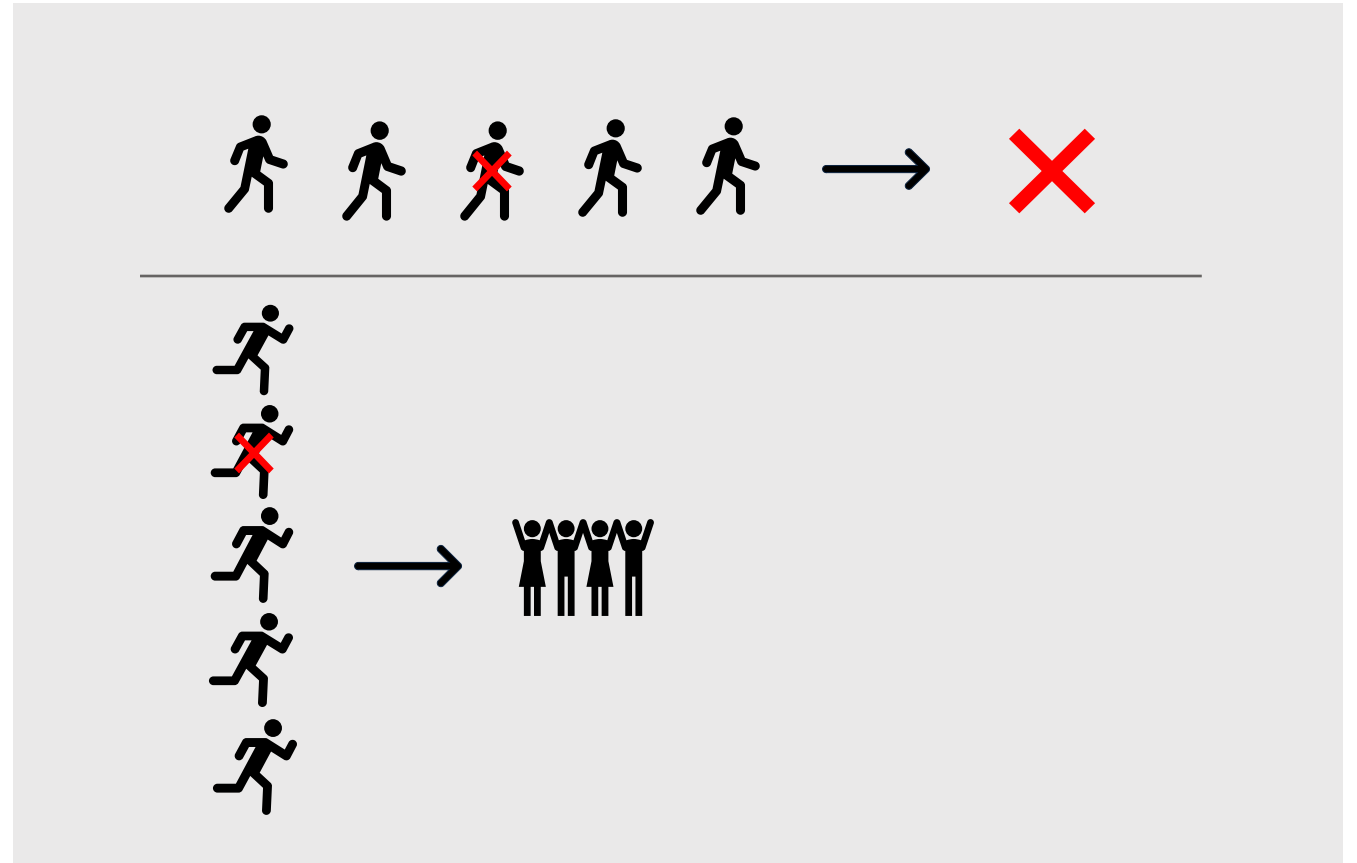
1. **Don't reinvent the wheel** for presentation. We leveraged the powerful r2rtf package as our rendering engine. Our focus shifted from how to draw a table to what data to feed the engine.
2. **Separate the “What” from the “How”**
We created a system where table specifications are defined in a simple, machine-readable format.



SE: Survival Extrapolation

How do we automate fitting dozens of models, handle inevitable failures gracefully, and visualize the results in a standard way?

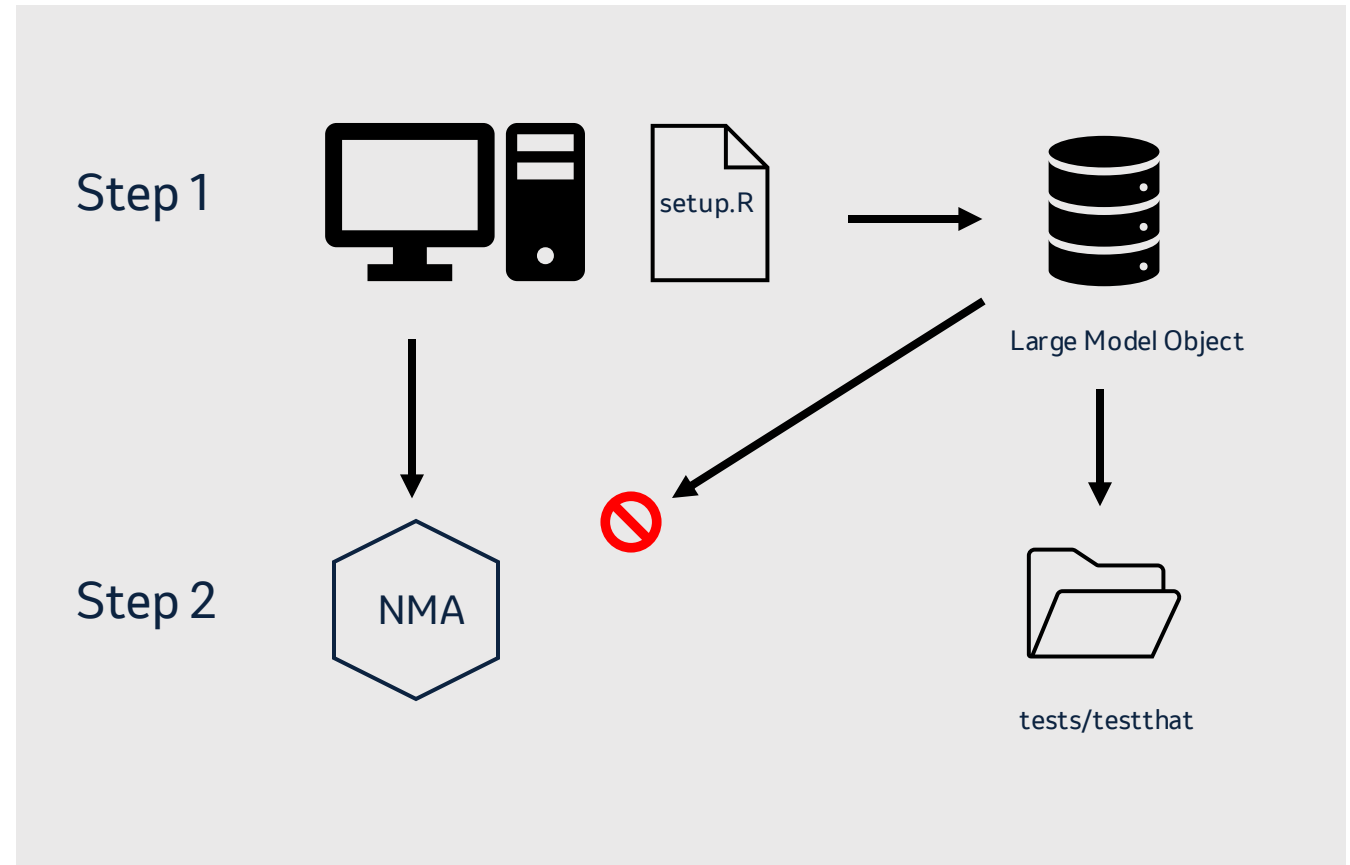
1. **Embrace parallelization.** We refactored our code to use the map and future.apply families. This allowed us to run models in parallel, cutting down analysis running time from hours to minutes.
2. **Build for failure.** A robust workflow must anticipate and handle errors. We wrapped our model-fitting logic in tryCatch(). Now, if one model fails, the package logs the error and simply moves on to the next one, ensuring the entire sequence completes.



NMA: Network Meta Analysis

How do we harmonize wildly different data sources and manage the testing of computationally intensive models without creating a bloated, slow package?

1. **Standardize at the entrance.** We designed a strict, standardized format for all incoming data. The very first step of any workflow is a function that processes and validates data, assigning it a formal class.
2. **Separate test generation from test execution.** We use the `tests/testthat/setup.R` file to build the large model objects once. These objects are available locally for developers to run tests, but they are not saved in the final package bundle, keeping it lean and fast.





Best Practices and Tools in Action

Best Practices: People & Process

Aligning on the "What" and "Why"

Good Practice	Risk Mitigated	Impact
Regular stakeholder discussions to refine requirements	Misaligned scope; rework due to misunderstood needs	Very High
Clear project management (Jira/GitHub Projects)	Unclear ownership; missed deadlines; scope drift	Very High
Formal "Definition of Done" (tests, docs, review)	Incomplete or low-quality features slipping through	High
Pilot studies of features in real analysis contexts	Building unusable or impractical features; late discovery	High

Best Practices: Code & Quality

Building It Right

Good Practice	Risk Mitigated	Impact
Single Responsibility Principle (SRP) for functions	Tightly coupled, hard-to-test, and hard-to-change code	Moderate
Minimize dependencies and use explicit namespaces	Dependency bloat; breakage surface; name masking bugs	High
Unit tests with testthat package integrated with CI/CD	Silent regressions; fragile refactoring; lack of a safety net	Very High
Object-Oriented Programming (S3/S4)	Unpredictable function behavior, code that is hard to maintain and extend	High

CI/CD: Continuous Integration and Continuous Delivery/Deployment

Best Practices: Collaboration & Delivery

Shipping It Safely

Good Practice	Risk Mitigated	Impact
Feature Branch Workflow with a protected main branch	Instability in production code; conflicting changes	High
Mandatory Pull Request (PR) with peer review	Low-quality, unreviewed code entering the main branch	Very High
Automated checks on PR (R CMD check, tests, website deployment)	Human error in manual checks; regressions; non-compliant code	Very High
Project-level dependency snapshots (e.g., with renv package)	"Works on my machine" syndrome; non-reproducible results	Very High



Discussion & Conclusion

Discussion: How to Design an Ecosystem?

Thinking about its architecture and a governance model to guide its evolution.



DS: Descriptive Statistics; NMA: Network Meta-Analysis; SE: Survival Extrapolation

Conclusion & The Next Evolution

A well-managed package ecosystem, built on modern software practices, is a powerful strategy for delivering reliable and reproducible results. However, the tools only succeed when supported by a collaborative team culture.



Thank you

Wenyu Li, Statistical Programming, MSD , Zurich, Switzerland
wenyu.li1@msd.com

Jean Muller, Statistical Programming, MSD, Paris, France
jean.muller@msd.com

