@jumpingrivers.com
Jumping Rivers Ltd
@jumpingrivers@fosstodon.org

Jumping Rivers

Validating Shiny Apps using the litmusverse

PSI 2026

About Me

- R/Shiny developer for 8 years
- Former Web Development for +10 years
- Open source creator and contributor
- Focused on R Validation with Jumping Rivers for the last 18 months

Jumping Rivers

Data Science Consultancy and Training

- All things data science
- Training
- Machine learning, DevOps, infrastructure
- Managed Posit services
- Dashboard development and deployment

Why Validation?

Why Validation Matters

- Shiny is everywhere:
  - Dashboards, review tools, exploratory apps
- Regulation means responsibility
- Regulators care about:
  - Traceability
  - Reproducibility
  - Documentation

What Happens Without Validation

- Risk of undetected bugs or misleading results
- No clear audit trail
- Compliance blockers
- Slows down internal reviews

The lack of evidence becomes a liability.

What Makes Apps Validatable

What Makes a Shiny App Validatable

- Modular, testable code
- Clear separation: UI, logic, data
- Version control for code and data
- Reproducible environments (e.g. Docker, renv)
- Minimal hidden state or side effects

Common Pitfalls

- Hard-coded file paths
- Ad hoc data manipulations in server.R
- Global variables with side effects
- No record of dependencies
- No tests, no logs

Unique challenges

- Interactivity: Real-time inputs create complex, dynamic state. Hard to test exhaustively.

  Use {shinytest2} for end-to-end tests; define expected input combinations; disable/lock irrelevant UI paths.

- Reactive Chains: Logic is hidden in reactive expressions; changes ripple unpredictably.

  Break logic into testable, modular functions. Use logs to trace reactive dependencies.

- User-Controlled Output: Users may trigger untested or invalid outputs.

  Log key actions; validate downloadable content; restrict custom inputs to valid ranges.

Unique challenges

- Environment-Specific Issues: Behavior can differ across deployment environments.

  Use {renv} or Docker to freeze environments; validate deployed versions, not just dev builds.

- No Native Audit Trail: Shiny doesn’t track user actions or input history.

  Add logging via {logger}, {loggit}, or custom solutions. Include app version and package snapshot in logs.

- Testing Complexity: Traditional unit testing doesn't cover reactive or UI-driven logic well.

  Test logic outside the Shiny context; combine {testthat} with {shinytest2} for coverage.

Practical Validation Strategies

Software Engineering Practices

- Unit tests with {testthat}
- End-to-end tests with {shinytest2}
- Linting & CI/CD pipelines
- Code review processes
- Automated report generation

Documentation Essentials

- Functional Requirements Spec (FRS)
- Test Plan & Summary (TP/TSR)
- User Guide / README
- Audit trail: who changed what and why
- Reproducibility evidence: renv.lock, Dockerfile

A Risk-Based Approach

- Match validation to app criticality
- Low risk: exploratory apps, sandboxed tools
- High risk: decision-support, regulatory outputs
- Define the Intended use, data sensitivity, user access

Example Tools for Validation

Tools for Testing & Reproducibility

- Commonly Used Tools
  - {testthat} – unit testing framework for R
  - {shinytest2} – browser-based end-to-end testing
  - {renv} – reproducible package environments
  - {lintr} – code linting and style checking

Tools for Risk & Security

- Validation & Risk Management Tools
  - {riskmetric} – package-level risk assessment
  - {oysteR} – scans R packages for known security vulnerabilities
  - diffify - comparison between different versions of R packages
  - Litmus.dashboard - Package level risk assessment explorer

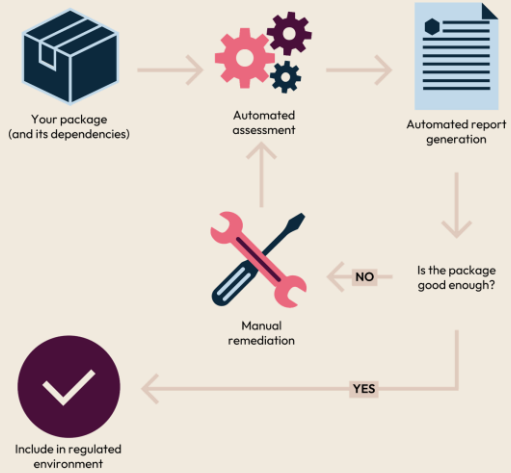
  - https://sonatype-nexus-community.github.io/oysteR/
  - https://pharmar.github.io/riskmetric/
  - https://diffify.com/
  - https://litmus-dashboard.jmpr.io/

What about dependencies?

Jumping Rivers Toolkit – The Litmusverse

https://www.jumpingrivers.com/litmus/

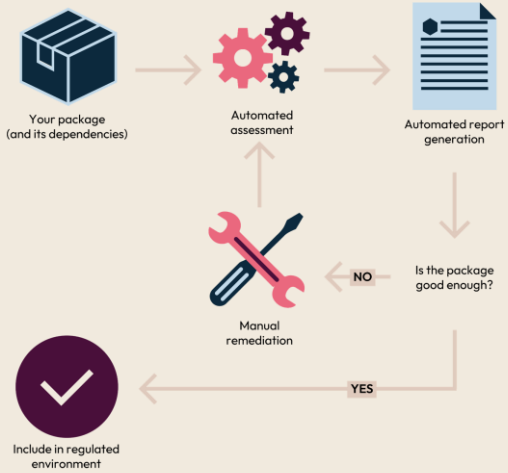
- Assess R packages with confidence
- Transparent scoring and reproducible regulated environments
- Explore results with interactive dashboards adn reports
- Built for compliance, trusted by data scientists



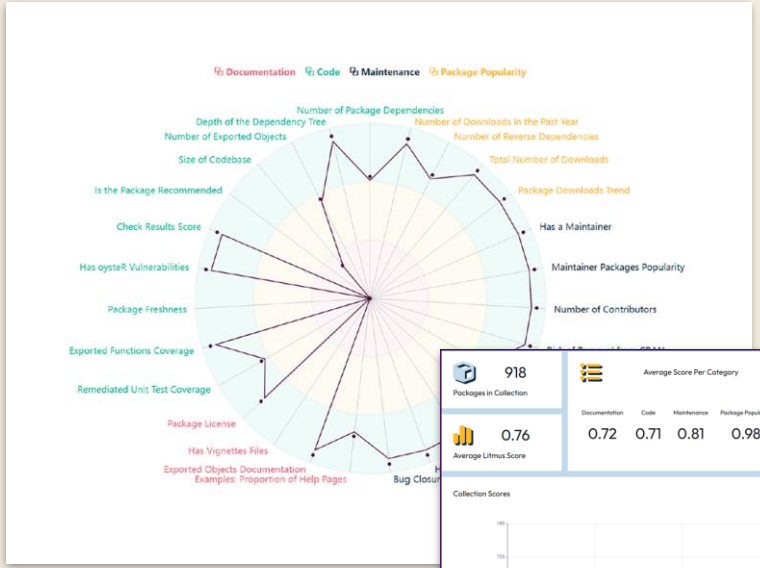
Jumping Rivers Toolkit – The Litmusverse

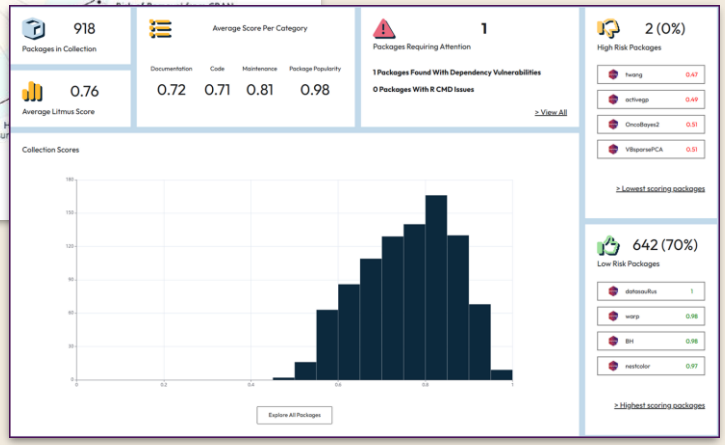
https://www.jumpingrivers.com/litmus/

- Assess the packages (and functions) you actually use
- Generate reports that record decisions validators can review
- If your dashboard is a package, run litmus on it as well in addition to the application own test suite



Jumping Rivers Toolkit – The Litmusverse





Closing Thoughts

Start Validatable, Stay Validated

- Build validation in from day one
- Document as you go
- Automate wherever possible
- Choose tools that support transparency

Key Takeaways

- Shiny apps = powerful, but risky if unmanaged
- Validation is about trust, not just compliance
- Good engineering = easier validation

Thank you!