

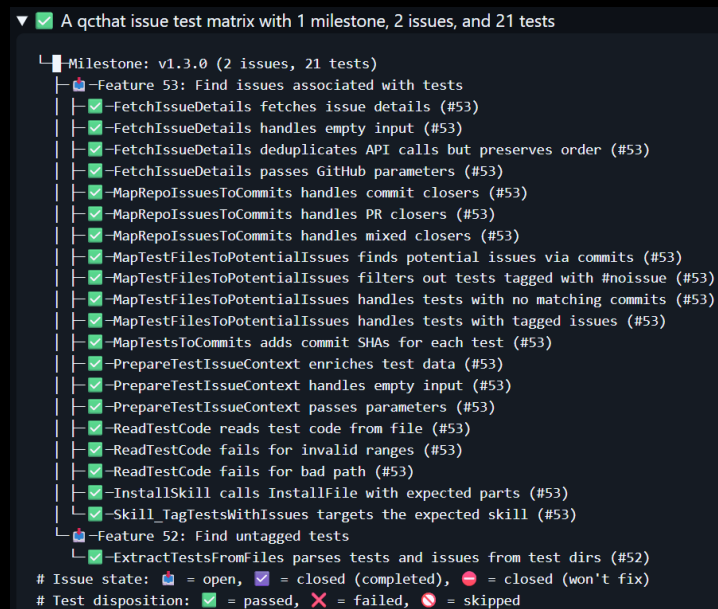
Scalable R Package Qualification Using a CI/CD Approach

{qcthat} generates qualification
reports automatically from your test
suite

A report links every requirement to its test evidence



A GitHub report automatically generated by **{qcthat}**, in this case applied to itself



The PR-Associated issues report

The report is generated from data you (might) already have

- GitHub issues = requirements
- {testthat} results = evidence
- {qcthat} connects them in automated reports

**Package qualification is
necessary but challenging**

Regulators require evidence that software performs as intended

- FDA (etc) require documented proof that software produces reliable results
- Adopting org often takes on burden of proof
- For R packages: must connect requirements to tested functionality

Qualification is often a manual, retrospective burden

- Qualification documents often created *after* development
- Dev team manually maps specs to functions and evidence
- Repeated for every new release

“Validation debt” delays releases and increases risk

- Deferred qualification: harder to reconstruct what was tested and why
- Manual docs drift out of sync with code
- Teams avoid upgrading packages to dodge re-qualification

**The standard R package lifecycle
generates evidence but lacks
traceability**

Modern R packages use `{testthat}` for robust automated testing

- Industry-standard testing framework
- Confirms functions behave as expected *and keep working after changes*
- Activate with `usethis::use_testthat()`

CI/CD pipelines ensure code integrity on every commit

- GitHub Actions run tests on every push
- Failures caught long before code reaches production
- Standard practice for well-maintained R packages

Standard {testthat} scripts provide the evidence of implementation

- Every test run produces machine-readable results: pass, fail, skip, or warning
- Proves that code *works*

But: A passing test suite doesn't prove which requirement was met

- `{testthat}` shows *what was tested*, not *why*
- No built-in link between GitHub issue and its tests
- Reviewers must manually reconstruct the mapping

{qcthat} connects issues to
tests for continuous qualification

Connecting tests to issues is straightforward

- Add `(#123)` (issue number) to `test_that()` description to tag test to issue
- Tag multiple issues: `(#84, #132)`
- Also acts as a clickable link in RStudio

The “Issue-Test Matrix” shows the connections between issues and tests

- `QCPackage()` generates a full report linking issues to tagged tests
- Organized by milestone; tests nested under linked issues
- Filter to subsets: `QCCompletedIssues()`, `QCPR()`, `QCMilestones()`

User Acceptance Testing (UAT) is integrated directly into `{testthat}` tests

- `ExpectUserAccepts()` creates GitHub issues requiring manual sign-off
- UAT issues = children of originating requirement issue
- UAT issue closed → test passes

{qcthat} transforms qualification
from a retrospective burden into a
seamless byproduct of development

Automated bots bring qualification reports directly to the Pull Request

- GitHub Action comments on every PR with reports:
 - PR-Associated Issues
 - Completed Issues
 - Milestone
 - UAT
- Reviewers see qualification status *before* merging
- Reports update automatically as the PR changes

Immutable artifacts are automatically attached to GitHub Releases

- Completed Issues and Milestone reports embedded in GitHub Release description
- Each release carries its own qualification evidence
- Version-controlled and tamper-evident by design

Packages are qualified continuously with every change

- Qualification as a byproduct of development, not a separate phase
- Devs tag tests to issues as they're written
- Reports identify untested issues and unlinked tests

You can add `{qcthat}` to your
package today

Setup takes minutes, not hours

- `pak::pak("Gilead-BioStats/qcthat")`
- `qcthat::use_qcthat()`
- Tag your tests:
`test_that("Thing works (#123)", ...)`
- Experimental: `qcthat::Skill_TagTestsWithIssues()`

Contact us with questions and comments

- `{qcthat}` package website:
gilead-biostats.github.io/qcthat
- Submit issues & discuss at
github.com/Gilead-BioStats/qcthat
- Chat with us after the talk!