Automated Testing in R

Jakub Sobolewski

2025-01-01

Table of contents

Preface

> ! This book is a work in progress.
>
> To read about automated testing in R, visit my blog.

Testing can feel like a chore.

You're coding away, everything seems fine—until it isn't. A bug pops up. You fix it, but then something else breaks. It's frustrating, time-consuming, and stressful. But what if it didn't have to be? What if testing wasn't just about catching bugs, but about making your work smoother, faster, and more reliable?

That's what this book is about.

R has grown from a tool for statisticians into a powerful programming language used for everything from exploratory data analysis to building Shiny apps and production-grade data pipelines. But as your projects grow, so do the risks. Code becomes harder to change. Bugs creep in. Shipping features slows down.

Testing can change that.

With automated testing, you can catch problems early, make changes with confidence, and keep your code in shape no matter how complex it gets. It's not just about preventing errors — it's about giving yourself the freedom to experiment and innovate without fear.

This book is your guide to testing in R.

We'll start with the fundamentals: why testing matters, what makes a good test, and how to think like a tester. From there, we'll get into the practical stuff. You'll learn how to write unit tests, handle tricky scenarios with mocks and stubs, and test Shiny apps. We'll cover advanced techniques, like snapshot testing and strategies for tackling legacy code.

But this isn't just about the "how." It's about the "why."

Testing isn't just a skill — it's a mindset. It's about approaching your work with care and confidence. It's about solving problems before they happen. And it's about making sure the code you write today won't break tomorrow.

Whether you're building packages, working on Shiny apps, or writing scripts for data analysis, this book will show you how testing can make your life easier — and your code better.

Let's get started.

The book assumes you have some experience with unit-testing and using {testthat}. If you are new to unit testing in R, I recommend you to read the {testthat} documentation first.

License

1 Anatomy of a test

A good test makes it clear what it's checking.

When a test mixes setup, running the system, and checking results, it gets confusing. Separating these steps into clear phases helps make the test's purpose obvious. This way, anyone reading the test can follow along easily.

We can make this separation explicit with Arrange, Act, Assert comments.

Those comments aren't just for show. They help you structure your tests and keep them focused. They also make it easier to spot missing steps or unnecessary complexity. They keep your tests consistent, which makes them easier to read and maintain.

If you are starting to write a test, put those comments in place first, then fill in the blanks.

1.1 Arrange

The first part of each test should setup the environment for the tested code.

```
test_that("...", {
  # Arrange #<<
  machine <- deep_thought$connect() #<<
  question <- "Everything" #<<

  # Act

  # Assert

})
```

1.2 Act

The second part of each test is to call the code that's being tested.

```
test_that("...", {
  # Arrange
  machine <- deep_thought$connect()
  question <- "Everything"

  # Act #<<
  result <- the_ultimate_question_of(machine, question) #<<

  # Assert

})
```

1.3 Assert

The third part of each test is to assert that the code behaves as expected.

```
test_that("...", {
  # Arrange
  machine <- deep_thought$connect()
  question <- "Everything"

  # Act
  result <- the_ultimate_question_of(machine, question)

  # Assert #<<
  expect_equal(result, 42) #<<
})
```

1.4 Teardown if needed

If we need to free resources, we should do that at the end of the test.

```
test_that(" ", {
  # Arrange
  machine <- deep_thought$connect()
  question <- "Everything"

  # Act
  result <- the_ultimate_question_of(machine, question)
```

```
  # Assert
  expect_equal(result, 42)

  machine$disconnect() #<<
})
```