

Learn how to write R code with fewer bugs.

The problem with programming is that you are always one typo away from writing something silly. Likewise with data analysis, a small mistake in your model can lead to a big mistake in your results. Combining the two disciplines means that it is all too easy for a missed minus sign to generate a false prediction that you don't spot until it's too late. Testing is the only way to be sure that your code, and your results, are correct.

Testing R Code teaches you how to perform development-time testing using the `testthat` package, allowing you to ensure that your code works as intended. The book also teaches run-time testing using the `assertive` package; enabling your users to correctly run your code.

After beginning with an introduction to testing in R, the book explores more advanced cases such as integrating tests into R packages; testing code that accesses databases; testing C++ code with `Rcpp`; and testing graphics. Each topic is explained with real-world examples and has accompanying exercises for readers to practise their skills — only a small amount of experience with R is needed to get started!

"In short, I loved it. There is a dearth of good material (or any material, really) on this topic so I'm excited to say that 'Testing R Code' met my high expectations. I know the subject rather well but not only did I see topics that were explained in new and easier ways than I have previously seen, I even learned quite a bit of new information myself. It is obvious that the author knows this topic inside and out."

—**Tony Fischetti**, Rensselaer Polytechnic Institute



CRC Press

Taylor & Francis Group
an informa business

www.crcpress.com

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487

711 Third Avenue
New York, NY 10017

2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK

K28980

ISBN: 978-1-4987-6365-3
90000



9 781498 763653

www.crcpress.com

1	Introduction	1
1.1	Interacting vs. Programming	1
1.2	Two Kinds of Testing	2
1.3	How Run-Time Testing Will Help You	2
1.4	How Development-Time Testing Will Help You	4
1.5	Summary	7
2	Run-Time Testing with <i>assertive</i>	9
2.1	Using Assertions	9
2.1.1	Chaining Assertions Together with Pipes	10
2.2	Using Predicates: <i>is</i> and <i>has</i> Functions	11
2.2.1	Exercise: Using Predicates and Assertions	13
2.3	The Virtual Package System	14
2.4	A Tour of the <i>assertive</i> Package	14
2.4.1	<i>assertive.base</i>	14
2.4.2	<i>assertive.properties</i>	16
2.4.3	<i>assertive.types</i>	19
2.4.4	<i>assertive.numbers</i>	20
2.4.5	Exercise: Examining an Object	22
2.4.6	<i>assertive.files</i>	22
2.4.7	Exercise: Inspecting Files and Directories	23
2.4.8	<i>assertive.strings</i>	24
2.4.9	<i>assertive.matrices</i>	25
2.4.10	Exercise: Testing Properties of Matrices	25
2.4.11	<i>assertive.sets</i>	25
2.4.12	<i>assertive.models</i>	26
2.4.13	<i>assertive.reflection</i>	27
2.4.14	Exercise: Explore Your R Setup	28
2.4.15	<i>assertive.datetimes</i>	28
2.4.16	<i>assertive.data</i> , <i>assertive.data.us</i> , and <i>assertive.data.uk</i>	29
2.4.17	Exercise: Checking Customer Data	30
2.4.18	<i>assertive.code</i>	31
2.5	Controlling Severity	32
2.6	Fail Early, Fail Often	33

2.7	Case Study: Calculating the Geometric Mean	34
2.7.1	Exercise: Calculating the Harmonic Mean	37
2.8	Alternatives	37
2.9	Summary	38
3	Development-Time Testing with testthat	39
3.1	Using Unit Tests	39
3.2	The Structure of a Unit Test	39
3.2.1	Exercise: Using <code>expect_equal</code>	40
3.2.2	How Equal Is Equal?	41
3.3	Testing Errors	42
3.3.1	Exercise: Using <code>expect_error</code>	42
3.4	Different Types of Expectation	42
3.5	Testing Warnings and Messages	43
3.5.1	Chaining Expectations Together with Pipes	44
3.5.2	Exercise: Using <code>expect_output</code>	45
3.5.3	Testing for Lack of Output	45
3.6	Varying the Strictness of Expectations	46
3.7	Providing Additional Information on Failure	47
3.8	Case Study: Calculating Square Roots	48
3.8.1	Exercise: Find More Tests for <code>square_root</code>	53
3.9	Other Expectations	54
3.9.1	Exercise: Testing the Return Type of Replicates	56
3.10	Organising Tests Using Contexts	57
3.11	Running Your Tests	57
3.12	Customizing How Test Results Are Reported	59
3.13	Alternatives	60
3.14	Summary	61
4	Writing Easily Maintainable and Testable Code	63
4.1	Don't Repeat Yourself	63
4.1.1	Case Study: Drawing Lots of Plots	64
4.1.2	Idea 1: Use Variables Rather Than Hard-Coded Values	68
4.1.3	Idea 2: For Values That You Want To Change Every- where, Update Global Settings	69
4.1.4	Idea 3: Wrap the Contents into a Function	70
4.1.5	Exercise: Reducing Duplication	71
4.2	Keep It Simple, Stupid	72
4.2.1	Simplifying Function Interfaces	72
4.2.2	Idea 1: Pass Arguments for Advanced Functionality to Another Function	74
4.2.3	Exercise: Outsourcing Argument Checking	76
4.2.4	Idea 2: Having Wrapper Functions for Specific Use Cases	76
4.2.5	Exercise: Wrappers for Formatting Currency	77

4.2.6	Idea 3: Auto-Guessing Defaults	77
4.2.7	Exercise: Providing Better Defaults for <code>write.csv</code> . .	78
4.2.8	Idea 4: Split Functionality into Many Functions	79
4.2.9	Exercise: Decomposing the <code>quantile</code> Function	80
4.2.10	Cyclomatic Complexity	80
4.2.11	How to Reduce Cyclomatic Complexity	83
4.2.12	Exercise: Calculating Leap Years	83
4.3	Summary	84
5	Integrating Testing into Packages	85
5.1	How to Make an R Package	85
5.1.1	Prerequisites	85
5.1.2	The Package Directory Structure	86
5.1.3	Including Tests in Your Package	86
5.2	Case Study: The <i>hypotenususer</i> Package	87
5.3	Checking Packages	90
5.3.1	Exercise: Make a Package with Tests	91
5.4	Using Version Control, Online Package Hosting, and Continu- ous Integration	92
5.4.1	Version Control with <i>git</i>	92
5.4.2	Online Project Hosting	92
5.4.3	Continuous Integration Services	93
5.5	Testing Packages on CRAN	95
5.5.1	Testing Packages with <i>r-hub</i>	96
5.6	Calculating Test Coverage Using <i>coveralls.io</i>	97
5.7	Summary	98
6	Advanced Development-Time Testing	99
6.1	Code with Side Effects	99
6.1.1	Exercise: Writing a Test That Handles Side Effects . .	101
6.2	Testing Complex Objects	101
6.2.1	Exercise: Testing a Complex Object	104
6.3	Testing Database Connections	104
6.3.1	Option 1: Mock the Connection	106
6.3.2	Option 2: Mock the Connection Wrapper	107
6.3.3	Option 3: Mock the Specific Query Functions	108
6.3.4	Summarizing the Pros and Cons of Each Database Method	110
6.4	Testing Rcpp Code	110
6.4.1	Getting Set Up to Use C++	111
6.4.2	Case Study: Extending the <i>hypotenususer</i> Package to In- clude C++ Code	114
6.4.3	Exercise: Testing an <i>Rcpp</i> Function	117
6.5	Testing Write Functions	118
6.5.1	Exercise: Writing INI Configuration Files	120

6.6	Testing Graphics	120
6.6.1	Generating the Report from the <i>markdown</i>	124
6.6.2	Including Graphics Tests in Packages	124
6.6.3	Exercise: Write a Graphics Test Report	124
6.7	Summary	125
7	Writing Your Own Assertions and Expectations	127
7.1	The Quick, Nearly Good Enough Option	127
7.2	Writing Scalar Predicates	128
7.2.1	Exercise: Writing a Custom Scalar Predicate	129
7.2.2	Writing Type Predicates	129
7.3	Writing Scalar Assertions	130
7.3.1	Exercise: Writing a Custom Scalar Assertion	131
7.4	Writing Vector Predicates	131
7.4.1	Exercise: Writing a Vector Predicate	132
7.5	Writing Vector Assertions	133
7.5.1	Exercise: Writing a Vector Assertion	134
7.6	Creating Custom Expectations	135
7.6.1	Exercise: Create a Custom Expectation	137
7.7	Summary	137
A	Answers to Exercises	139
A.1	Preface	139
A.1.1	Exercise: Are You Ready?	139
A.2	Chapter 2	139
A.2.1	Exercise: Using Predicates and Assertions	139
A.2.2	Exercise: Examining an Object	141
A.2.3	Exercise: Inspecting Files and Directories	142
A.2.4	Exercise: Testing Properties of Matrices	143
A.2.5	Exercise: Explore Your R Setup	143
A.2.6	Exercise: Checking Customer Data	143
A.2.7	Exercise: Calculating the Harmonic Mean	147
A.3	Chapter 3	149
A.3.1	Exercise: Using <code>expect_equal</code>	149
A.3.2	Exercise: Using <code>expect_error</code>	150
A.3.3	Exercise: Using <code>expect_output</code>	151
A.3.4	Exercise: Find More Tests for <code>square_root</code>	151
A.3.5	Exercise: Testing the Return Type of Replicates	152
A.4	Chapter 4	153
A.4.1	Exercise: Reducing duplication	153
A.4.2	Exercise: Outsourcing Argument Checking	155
A.4.3	Exercise: Wrappers for Formatting Currency	156
A.4.4	Exercise: Providing Better Defaults for <code>write.csv</code>	157
A.4.5	Exercise: Decomposing the <code>quantile</code> Function	157
A.4.6	Exercise: Calculating Leap Years	158

A.5	Chapter 5	159
A.5.1	Exercise: Make a Package with Tests	159
A.6	Chapter 6	162
A.6.1	Exercise: Writing a Test That Handles Side Effects . .	162
A.6.2	Exercise: Testing a Complex Object	162
A.6.3	Exercise: Testing an <i>Rcpp</i> Function	163
A.6.4	Exercise: Writing INI Configuration Files	165
A.6.5	Exercise: Write a Graphics Test Report	167
A.7	Chapter 7	167
A.7.1	Exercise: Writing a Custom Scalar Predicate	167
A.7.2	Exercise: Writing a Custom Scalar Assertion	168
A.7.3	Exercise: Writing a Vector Predicate	169
A.7.4	Exercise: Writing a Vector Assertion	169
A.7.5	Exercise: Create a Custom Expectation	170
	Bibliography	173
	Concept Index	177
	Package Index	179
	Dataset Index	181
	People Index	183
	Function Index	185