

---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Who should read this book . . . . .	3
1.2	What you will get out of this book . . . . .	3
1.3	Meta-techniques . . . . .	4
1.4	Recommended reading . . . . .	5
1.5	Getting help . . . . .	6
1.6	Acknowledgments . . . . .	6
1.7	Conventions . . . . .	8
1.8	Colophon . . . . .	8
<b>I</b>	<b>Foundations</b>	<b>11</b>
<b>2</b>	<b>Data structures</b>	<b>13</b>
2.1	Vectors . . . . .	14
2.1.1	Atomic vectors . . . . .	15
2.1.1.1	Types and tests . . . . .	15
2.1.1.2	Coercion . . . . .	16
2.1.2	Lists . . . . .	17
2.1.3	Exercises . . . . .	18
2.2	Attributes . . . . .	19
2.2.0.1	Names . . . . .	20
2.2.1	Factors . . . . .	21
2.2.2	Exercises . . . . .	23
2.3	Matrices and arrays . . . . .	24



2.3.1	Exercises . . . . .	26
2.4	Data frames . . . . .	27
2.4.1	Creation . . . . .	27
2.4.2	Testing and coercion . . . . .	28
2.4.3	Combining data frames . . . . .	28
2.4.4	Special columns . . . . .	29
2.4.5	Exercises . . . . .	30
2.5	Answers . . . . .	31
<b>3</b>	<b>Subsetting</b>	<b>33</b>
3.1	Data types . . . . .	34
3.1.1	Atomic vectors . . . . .	34
3.1.2	Lists . . . . .	37
3.1.3	Matrices and arrays . . . . .	37
3.1.4	Data frames . . . . .	38
3.1.5	S3 objects . . . . .	39
3.1.6	S4 objects . . . . .	39
3.1.7	Exercises . . . . .	39
3.2	Subsetting operators . . . . .	40
3.2.1	Simplifying vs. preserving subsetting . . . . .	41
3.2.2	\$ . . . . .	43
3.2.3	Missing/out of bounds indices . . . . .	44
3.2.4	Exercises . . . . .	45
3.3	Subsetting and assignment . . . . .	45
3.4	Applications . . . . .	46
3.4.1	Lookup tables (character subsetting) . . . . .	46
3.4.2	Matching and merging by hand (integer subsetting) . . . . .	47
3.4.3	Random samples/bootstrap (integer subsetting) . . . . .	48
3.4.4	Ordering (integer subsetting) . . . . .	49



3.4.5	Expanding aggregated counts (integer subsetting) . . . . .	50
3.4.6	Removing columns from data frames (character subsetting) . . . . .	51
3.4.7	Selecting rows based on a condition (logical subsetting) . . . . .	52
3.4.8	Boolean algebra vs. sets (logical & integer subsetting) . . . . .	53
3.4.9	Exercises . . . . .	55
3.5	Answers . . . . .	55
<b>4</b>	<b>Vocabulary</b>	<b>57</b>
4.1	The basics . . . . .	57
4.2	Common data structures . . . . .	59
4.3	Statistics . . . . .	60
4.4	Working with R . . . . .	61
4.5	I/O . . . . .	62
<b>5</b>	<b>Style guide</b>	<b>63</b>
5.1	Notation and naming . . . . .	63
5.1.1	File names . . . . .	63
5.1.2	Object names . . . . .	64
5.2	Syntax . . . . .	65
5.2.1	Spacing . . . . .	65
5.2.2	Curly braces . . . . .	66
5.2.3	Line length . . . . .	67
5.2.4	Indentation . . . . .	67
5.2.5	Assignment . . . . .	67
5.3	Organisation . . . . .	68
5.3.1	Commenting guidelines . . . . .	68



<b>6</b>	<b>Functions</b>	<b>69</b>
6.1	Function components . . . . .	71
6.1.1	Primitive functions . . . . .	71
6.1.2	Exercises . . . . .	72
6.2	Lexical scoping . . . . .	73
6.2.1	Name masking . . . . .	74
6.2.2	Functions vs. variables . . . . .	75
6.2.3	A fresh start . . . . .	76
6.2.4	Dynamic lookup . . . . .	77
6.2.5	Exercises . . . . .	78
6.3	Every operation is a function call . . . . .	79
6.4	Function arguments . . . . .	81
6.4.1	Calling functions . . . . .	81
6.4.2	Calling a function given a list of arguments . . . . .	83
6.4.3	Default and missing arguments . . . . .	83
6.4.4	Lazy evaluation . . . . .	84
6.4.5	... . . . .	88
6.4.6	Exercises . . . . .	89
6.5	Special calls . . . . .	89
6.5.1	Infix functions . . . . .	90
6.5.2	Replacement functions . . . . .	91
6.5.3	Exercises . . . . .	93
6.6	Return values . . . . .	94
6.6.1	On exit . . . . .	97
6.6.2	Exercises . . . . .	97
6.7	Quiz answers . . . . .	98



<b>7 OO field guide</b>	<b>99</b>
7.1 Base types . . . . .	101
7.2 S3 . . . . .	102
7.2.1 Recognising objects, generic functions, and meth- ods . . . . .	102
7.2.2 Defining classes and creating objects . . . . .	105
7.2.3 Creating new methods and generics . . . . .	106
7.2.4 Method dispatch . . . . .	107
7.2.5 Exercises . . . . .	109
7.3 S4 . . . . .	111
7.3.1 Recognising objects, generic functions, and meth- ods . . . . .	111
7.3.2 Defining classes and creating objects . . . . .	113
7.3.3 Creating new methods and generics . . . . .	115
7.3.4 Method dispatch . . . . .	115
7.3.5 Exercises . . . . .	116
7.4 RC . . . . .	116
7.4.1 Defining classes and creating objects . . . . .	117
7.4.2 Recognising objects and methods . . . . .	119
7.4.3 Method dispatch . . . . .	119
7.4.4 Exercises . . . . .	120
7.5 Picking a system . . . . .	120
7.6 Quiz answers . . . . .	121
<b>8 Environments</b>	<b>123</b>
8.1 Environment basics . . . . .	124
8.1.1 Exercises . . . . .	130
8.2 Recursing over environments . . . . .	130
8.2.1 Exercises . . . . .	132
8.3 Function environments . . . . .	133



8.3.1	The enclosing environment . . . . .	133
8.3.2	Binding environments . . . . .	134
8.3.3	Execution environments . . . . .	136
8.3.4	Calling environments . . . . .	138
8.3.5	Exercises . . . . .	140
8.4	Binding names to values . . . . .	141
8.4.1	Exercises . . . . .	143
8.5	Explicit environments . . . . .	144
8.5.1	Avoiding copies . . . . .	145
8.5.2	Package state . . . . .	146
8.5.3	As a hashmap . . . . .	146
8.6	Quiz answers . . . . .	147
<b>9</b>	<b>Debugging, condition handling, and defensive programming</b>	<b>149</b>
9.1	Debugging techniques . . . . .	151
9.2	Debugging tools . . . . .	153
9.2.1	Determining the sequence of calls . . . . .	154
9.2.2	Browsing on error . . . . .	155
9.2.3	Browsing arbitrary code . . . . .	157
9.2.4	The call stack: <code>traceback()</code> , <code>where</code> , and <code>recover()</code>	158
9.2.5	Other types of failure . . . . .	158
9.3	Condition handling . . . . .	160
9.3.1	Ignore errors with <code>try</code> . . . . .	160
9.3.2	Handle conditions with <code>tryCatch()</code> . . . . .	162
9.3.3	<code>withCallingHandlers()</code> . . . . .	165
9.3.4	Custom signal classes . . . . .	166
9.3.5	Exercises . . . . .	168
9.4	Defensive programming . . . . .	168
9.4.1	Exercises . . . . .	169
9.5	Quiz answers . . . . .	170



<b>II Functional programming</b>	<b>173</b>
<b>10 Functional programming</b>	<b>175</b>
10.1 Motivation . . . . .	176
10.2 Anonymous functions . . . . .	181
10.2.1 Exercises . . . . .	183
10.3 Closures . . . . .	183
10.3.1 Function factories . . . . .	186
10.3.2 Mutable state . . . . .	186
10.3.3 Exercises . . . . .	188
10.4 Lists of functions . . . . .	189
10.4.1 Moving lists of functions to the global environ- ment . . . . .	191
10.4.2 Exercises . . . . .	193
10.5 Case study: numerical integration . . . . .	193
10.5.1 Exercises . . . . .	196
<b>11 Functionals</b>	<b>199</b>
11.1 My first functional: <code>lapply()</code> . . . . .	201
11.1.1 Looping patterns . . . . .	203
11.1.2 Exercises . . . . .	204
11.2 For loop functionals: friends of <code>lapply()</code> . . . . .	205
11.2.1 Vector output: <code>sapply</code> and <code>vapply</code> . . . . .	205
11.2.2 Multiple inputs: <code>Map</code> (and <code>mapply</code> ) . . . . .	207
11.2.3 Rolling computations . . . . .	209
11.2.4 Parallelisation . . . . .	212
11.2.5 Exercises . . . . .	213
11.3 Manipulating matrices and data frames . . . . .	214
11.3.1 Matrix and array operations . . . . .	215
11.3.2 Group apply . . . . .	216



11.3.3	The plyr package . . . . .	218
11.3.4	Exercises . . . . .	219
11.4	Manipulating lists . . . . .	219
11.4.1	Reduce() . . . . .	219
11.4.2	Predicate functionals . . . . .	221
11.4.3	Exercises . . . . .	221
11.5	Mathematical functionals . . . . .	222
11.5.1	Exercises . . . . .	224
11.6	Loops that should be left as is . . . . .	224
11.6.1	Modifying in place . . . . .	225
11.6.2	Recursive relationships . . . . .	225
11.6.3	While loops . . . . .	226
11.7	A family of functions . . . . .	227
11.7.1	Exercises . . . . .	232
<b>12</b>	<b>Function operators</b>	<b>233</b>
12.1	Behavioural FOs . . . . .	235
12.1.1	Memoisation . . . . .	237
12.1.2	Capturing function invocations . . . . .	239
12.1.3	Laziness . . . . .	242
12.1.4	Exercises . . . . .	243
12.2	Output FOs . . . . .	244
12.2.1	Minor modifications . . . . .	245
12.2.2	Changing what a function does . . . . .	246
12.2.3	Exercises . . . . .	248
12.3	Input FOs . . . . .	248
12.3.1	Prefilling function arguments: partial function application . . . . .	248
12.3.2	Changing input types . . . . .	249
12.3.3	Exercises . . . . .	251



12.4	Combining FOs . . . . .	252
12.4.1	Function composition . . . . .	252
12.4.2	Logical predicates and boolean algebra . . . . .	254
12.4.3	Exercises . . . . .	255
<b>III</b>	<b>Computing on the language</b>	<b>257</b>
<b>13</b>	<b>Non-standard evaluation</b>	<b>259</b>
13.1	Capturing expressions . . . . .	260
13.1.1	Exercises . . . . .	262
13.2	Non-standard evaluation in subset . . . . .	263
13.2.1	Exercises . . . . .	266
13.3	Scoping issues . . . . .	267
13.3.1	Exercises . . . . .	269
13.4	Calling from another function . . . . .	269
13.4.1	Exercises . . . . .	272
13.5	Substitute . . . . .	273
13.5.1	Adding an escape hatch to substitute . . . . .	276
13.5.2	Capturing unevaluated . . . . .	277
13.5.3	Exercises . . . . .	277
13.6	The downsides of non-standard evaluation . . . . .	278
13.6.1	Exercises . . . . .	279
<b>14</b>	<b>Expressions</b>	<b>281</b>
14.1	Structure of expressions . . . . .	282
14.1.1	Exercises . . . . .	286
14.2	Names . . . . .	286
14.2.1	Exercises . . . . .	287
14.3	Calls . . . . .	288
14.3.1	Modifying a call . . . . .	289



14.3.2	Creating a call from its components . . . . .	290
14.3.3	Exercises . . . . .	291
14.4	Capturing the current call . . . . .	292
14.4.1	Exercises . . . . .	295
14.5	Pairlists . . . . .	295
14.5.1	Exercises . . . . .	298
14.6	Parsing and deparsing . . . . .	298
14.6.1	Exercises . . . . .	300
14.7	Walking the AST with recursive functions . . . . .	300
14.7.1	Finding F and T . . . . .	301
14.7.2	Finding all variables created by assignment . . . . .	302
14.7.3	Modifying the call tree . . . . .	307
14.7.4	Exercises . . . . .	309
<b>15</b>	<b>Domain specific languages</b>	<b>311</b>
15.1	HTML . . . . .	312
15.1.1	Goal . . . . .	313
15.1.2	Escaping . . . . .	314
15.1.3	Basic tag functions . . . . .	315
15.1.4	Tag functions . . . . .	317
15.1.5	Processing all tags . . . . .	318
15.1.6	Exercises . . . . .	320
15.2	LaTeX . . . . .	320
15.2.1	LaTeX mathematics . . . . .	321
15.2.2	Goal . . . . .	321
15.2.3	<code>to_math</code> . . . . .	322
15.2.4	Known symbols . . . . .	322
15.2.5	Unknown symbols . . . . .	323
15.2.6	Known functions . . . . .	325
15.2.7	Unknown functions . . . . .	326
15.2.8	Exercises . . . . .	328



**IV Performance 329****16 Performance 331**

16.1 Why is R slow? . . . . . 332

16.2 Microbenchmarking . . . . . 333

16.2.1 Exercises . . . . . 334

16.3 Language performance . . . . . 335

16.3.1 Extreme dynamism . . . . . 335

16.3.2 Name lookup with mutable environments . . . . . 337

16.3.3 Lazy evaluation overhead . . . . . 339

16.3.4 Exercises . . . . . 340

16.4 Implementation performance . . . . . 341

16.4.1 Extracting a single value from a data frame . . . 341

16.4.2 `ifelse()`, `pmin()`, and `pmax()` . . . . . 342

16.4.3 Exercises . . . . . 344

16.5 Alternative R implementations . . . . . 344

**17 Optimising code 349**

17.1 Measuring performance . . . . . 350

17.1.1 Limitations . . . . . 354

17.2 Improving performance . . . . . 355

17.3 Code organisation . . . . . 356

17.4 Has someone already solved the problem? . . . . . 357

17.4.1 Exercises . . . . . 358

17.5 Do as little as possible . . . . . 359

17.5.1 Exercises . . . . . 365

17.6 Vectorise . . . . . 366

17.6.1 Exercises . . . . . 368

17.7 Avoid copies . . . . . 368

17.8 Byte code compilation . . . . . 370



17.9	Case study: t-test . . . . .	371
17.10	Parallelise . . . . .	373
17.11	Other techniques . . . . .	375
<b>18</b>	<b>Memory</b>	<b>377</b>
18.1	Object size . . . . .	378
18.1.1	Exercises . . . . .	382
18.2	Memory usage and garbage collection . . . . .	383
18.3	Memory profiling with lineprof . . . . .	385
18.3.1	Exercises . . . . .	388
18.4	Modification in place . . . . .	389
18.4.1	Loops . . . . .	392
18.4.2	Exercises . . . . .	393
<b>19</b>	<b>High performance functions with Rcpp</b>	<b>395</b>
19.1	Getting started with C++ . . . . .	397
19.1.1	No inputs, scalar output . . . . .	398
19.1.2	Scalar input, scalar output . . . . .	399
19.1.3	Vector input, scalar output . . . . .	399
19.1.4	Vector input, vector output . . . . .	401
19.1.5	Matrix input, vector output . . . . .	402
19.1.6	Using sourceCpp . . . . .	403
19.1.7	Exercises . . . . .	405
19.2	Attributes and other classes . . . . .	406
19.2.1	Lists and data frames . . . . .	407
19.2.2	Functions . . . . .	408
19.2.3	Other types . . . . .	409
19.3	Missing values . . . . .	409
19.3.1	Scalars . . . . .	410
19.3.1.1	Integers . . . . .	410



19.3.1.2	Doubles . . . . .	411
19.3.2	Strings . . . . .	411
19.3.3	Boolean . . . . .	412
19.3.4	Vectors . . . . .	412
19.3.5	Exercises . . . . .	413
19.4	Rcpp sugar . . . . .	413
19.4.1	Arithmetic and logical operators . . . . .	414
19.4.2	Logical summary functions . . . . .	414
19.4.3	Vector views . . . . .	415
19.4.4	Other useful functions . . . . .	416
19.5	The STL . . . . .	416
19.5.1	Using iterators . . . . .	417
19.5.2	Algorithms . . . . .	418
19.5.3	Data structures . . . . .	419
19.5.4	Vectors . . . . .	420
19.5.5	Sets . . . . .	421
19.5.6	Map . . . . .	422
19.5.7	Exercises . . . . .	423
19.6	Case studies . . . . .	423
19.6.1	Gibbs sampler . . . . .	424
19.6.2	R vectorisation vs. C++ vectorisation . . . . .	425
19.7	Using Rcpp in a package . . . . .	428
19.8	Learning more . . . . .	429
19.9	Acknowledgments . . . . .	430
<b>20</b>	<b>R's C interface</b> . . . . .	<b>431</b>
20.1	Calling C functions from R . . . . .	432
20.2	C data structures . . . . .	434
20.3	Creating and modifying vectors . . . . .	435
20.3.1	Creating vectors and garbage collection . . . . .	435



20.3.2	Missing and non-finite values . . . . .	437
20.3.3	Accessing vector data . . . . .	439
20.3.4	Character vectors and lists . . . . .	440
20.3.5	Modifying inputs . . . . .	441
20.3.6	Coercing scalars . . . . .	442
20.3.7	Long vectors . . . . .	442
20.4	Pairlists . . . . .	443
20.5	Input validation . . . . .	445
20.6	Finding the C source code for a function . . . . .	447
<b>Index</b>		<b>451</b>