Spis treści

Przedmowa 13

Część I. Rozpoczęcie pracy 25

1. Cała gra 27

Wczytywanie pakietu devtools i związanych z nim narzędzi 27
Przykładowy pakiet: regexcite 28
Podgląd gotowego produktu 28
create_package() 29
use_git() 31
Tworzenie pierwszej funkcji 32
use_r() 33
load_all() 33
Przekazanie funkcji strsplit1() do repozytorium 35
check() 35
Edycja pliku DESCRIPTION 36
use_mit_license() 37
document() 37
Zmiany w pliku NAMESPACE 39
Ponowne wywołanie check() 39
install() 39
use_testthat() 40
use_package() 42
use_github() 45
use_readme_rmd() 45
Koniec pracy: check() i install() 48
Podsumowanie 49

2. Konfiguracja systemu 51
devtools, usethis i Ty 51
Osobista konfiguracja podstawowa 52
Zestaw narzędzi do tworzenia pakietów R 53
Windows 53
macOS 54
Linux 54
Weryfikacja poprawności systemu 54

3. Stan i struktura pakietu 57
Stany pakietu 57
Kod źródłowy pakietu 57
Pakiet umieszczony w paczce 58
.Rbuildignore 59
Pakiet binarny 62
Pakiet zainstalowany 63
Pakiet w pamięci 65
Biblioteki pakietu 66

4. Podstawy sposobu pracy 69
Tworzenie pakietu 69
Analiza istniejącego środowiska 69
Nadanie nazwy pakietowi 70
Tworzenie pakietu 72
Dlaczego należy korzystać z wywołania create_package()? 73
Projekty RStudio 74
Zalety używania projektów RStudio 74
Tworzenie Projektu RStudio 75
Co tworzy Projekt RStudio? 76
Uruchamianie Projektu RStudio 76
Projekt RStudio kontra aktywny projekt usethis 78
Bieżący katalog roboczy i dyscyplina związana ze ścieżkami dostępu 78
Funkcja load_all() 79
Zalety funkcji load_all() 79
Inne sposoby wywołania load_all() 80
check() i R CMD check 81
Sposób pracy 82
Pod maską polecenia R CMD check 83

5. Zawartość pakietu 85
Alfa — działający skrypt 85
Bravo — lepsza wersja działającego skryptu 87
Charlie — oddzielny plik z funkcjami pomocniczymi 88

Delta — nieudana próba utworzenia pakietu 89
Echo — działający pakiet 92
Foxtrot — kompilacja kontra uruchomienie 95
Golf — efekty uboczne 96
Rozważania końcowe 99
Skrypt kontra pakiet 99
Wyszukiwanie w pakiecie 99
Pakiet kodu jest inny 99

Część II. Komponenty pakietu 101

6. Kod w języku R 103
Umieszczanie funkcji w plikach 103
Szybsze informacje zwrotne dzięki użyciu wywołania load_all() 105
Styl tworzenia kodu 105
Kiedy kod jest wykonywany? 106
Przykład: ścieżka dostępu zwrócona przez system.file() 107
Przykład: dostępne kolory 108
Przykład: alias dla funkcji 109
Szanowanie środowiska R 110
Zarządzanie stanem za pomocą pakietu withr 111
Przywracanie stanu za pomocą wywołania base::on.exit() 113
Odizolowanie efektów ubocznych 114
Gdy potrzebujesz efektów ubocznych 114
Nieustanne sprawdzanie poprawności 115

7. Dane 117
Dane wyeksportowane 118
Zachowanie pierwotnej historii danych pakietu 120
Dokumentowanie zbiorów danych 121
Znaki inne niż ASCII w danych 122
Dane wewnętrzne 123
Plik danych niezmodyfikowanych 124
Ścieżki dostępu plików 125
Funkcje pomocnicze pkg_example() 126
Stan wewnętrzny 126
Trwałe dane użytkownika 129

8. Inne komponenty 131
Inne katalogi 131
Zainstalowane pliki 132
Plik CITATION w pakiecie 133
Narzędzia konfiguracji 134

Część III. Metadane pakietu 137

9. Plik DESCRIPTION 139
Plik DESCRIPTION 139
Pola Title i Description — na czym polega działanie pakietu? 140
Pole Authors — kim jesteś? 142
Adres URL i zgłaszanie błędów 144
Pole License 144
Pola Imports, Suggests i Friends 144
Wersja minimalna 145
Pola Depends i LinkingTo 146
Problem z wersją R 147
Inne pola 148
Pola niestandardowe 149

10. Zależności — nastawienie i kontekst 151
Kiedy należy skorzystać z zależności? 152
Zależności nie są takie same 152
Postaw na podejście holistyczne, zrównoważone i ilościowe 154
Przemyślenia dotyczące zależności ściśle związanych z tidyverse 156
Imports czy Suggests? 157
Przestrzeń nazw 158
Uzasadnienie 158
Plik NAMESPACE 159
Ścieżka wyszukiwania 161
Wyszukiwanie funkcji dla kodu użytkownika 161
Wyszukiwanie funkcji w pakiecie 163
Dołączanie kontra wczytywanie 165
Imports czy Depends? 166

11. Zależności — praktyka 169
Niejasności związane z importowaniem 169
Konwencje zastosowane w tym rozdziale 170
Sposób pracy z plikiem NAMESPACE 170

Pakiet został wymieniony w polu Imports 171
Kod w katalogu R 171
W kodzie testu 174
W przykładach i w ulotkach 174
Pakiet jest wymieniony w polu Suggests 175
W kodzie zdefiniowanym w plikach katalogu R 175
W kodzie testowym 176
W przykładach i w ulotkach 177
Pakiet został wymieniony w polu Depends 177
W kodzie zdefiniowanym w plikach katalogu R oraz w kodzie testowym 178
W przykładach i w ulotkach 178
Pakiet jest zależnością niestandardową 178
Zależność od programistycznej wersji pakietu 179
Pole Config/Needs/* 179
Eksportowanie 180
Co należy wyeksportować? 180
Ponowne eksportowanie 181
Operacje importowania i eksportowania powiązane z systemem S3 182

12. Licencje 187
Szersza perspektywa 187
Kod, który tworzysz 188
Pliki kluczy 188
Więcej licencji dla kodu 189
Licencje dla danych 190
Ponowne licencjonowanie 190
Kod przekazany Tobie 191
Kod dołączony do pakietu 192
Zgodność licencji 192
Jak dołączyć kod? 193
Kod używany przez Ciebie 194

Część IV. Testowanie 195

13. Podstawy testowania 197
Dlaczego warto podjąć wysiłek związany z testowaniem? 197
Wprowadzenie do pakietu testthat 199
Struktura testów i praca z nimi 199
Konfiguracja początkowa 200
Tworzenie testu 200
Uruchamianie testów 202
Organizacja testu 204

Oczekiwania 205
Sprawdzanie równości 206
Sprawdzanie pod kątem błędów 206
Testy migawek 208
Skróty dla innych często spotykanych wzorców 211

14. Projektowanie zbioru testów 213
Co należy testować? 213
Pokrycie testami 214
Ogólne reguły dotyczące testowania 215
Testy samowystarczalne 215
Testy odizolowane 217
Planowanie niepowodzenia testu 220
Powtarzanie jest w porządku 221
Eliminowanie tarć między testowaniem interaktywnym i zautomatyzowanym 222
Pliki związane z testowaniem 224
Zniknąć z pola widzenia — pliki w katalogu R 224
tests/testthat.R 225
Pliki pomocnicze testthat 225
Pliki konfiguracyjne testthat 226
Pliki ignorowane przez testthat 227
Przechowywanie danych testowych 227
Gdzie będą zapisywane pliki podczas testów? 228

15. Zaawansowane techniki testowania 229
Przygotowywanie warunków początkowych testów 229
Tworzenie useful_thing za pomocą funkcji pomocniczej 230
Tworzenie (i usuwanie) lokalnego elementu useful_thing 230
Trwałe przechowywanie konkretnego elementu useful_thing 231
Budowanie własnych narzędzi testowania 232
Funkcja pomocnicza zdefiniowana w teście 232
Oczekiwania niestandardowe 233
Kiedy testowanie staje się trudne? 234
Pomijanie testu 234
Imitacje 236
Klucze tajne użytkownika 237
Uwagi specjalne dotyczące pakietów repozytorium CRAN 237
Pomijanie testu 238
Szybkość działania 238
Odtwarzalność 238
Testy niepewne 239
Higiena związana z procesem i systemem plików 239

Część V. Dokumentacja 241

16. Dokumentacja funkcji 243
Podstawy pracy z roxygen2 244
Sposób pracy z dokumentacją 244
Komentarze, bloki i tagi roxygen2 247
Najważniejsze funkcjonalności składni Markdown 248
Tytuł, opis, szczegóły 249
Tytuł 250
Opis 251
Szczegóły 252
Argumenty 253
Wiele argumentów 253
Dziedziczenie argumentów 254
Wartość zwrotna 255
Przykłady 256
Treść 257
Pozostawienie środowiska w jego początkowej postaci 259
Błędy 260
Zależności i wykonywanie warunkowe 260
Łączenie przykładów i tekstu 262
Wielokrotne wykorzystywanie dokumentacji 263
Wiele funkcji w jednym temacie 263
Dokumentacja dziedziczona 264
Dokumenty potomne 264
Temat pomocy dla pakietu 265

17. Ulotki 267
Sposób pracy podczas tworzenia ulotki 268
Metadane 269
Rada dotycząca tworzenia ulotek 271
Diagramy 272
Łącza 272
Ścieżki dostępu plików 272
Ile ulotek? 273
Publikacje naukowe 274
Uwagi szczególne dotyczące kodu ulotki 274
Artykuł zamiast ulotki 276
Jak są tworzone i sprawdzane ulotki? 276
Polecenie R CMD build i ulotki 276
Polecenie R CMD check i ulotki 278

18. Inne pliki Markdown 281
Plik README 281
Pliki README.Rmd i README.md 282
Plik NEWS 285

19. Witryna internetowa 287
Tworzenie witryny internetowej 287
Wdrożenie 289
Co dalej? 290
Logo 291
Indeks 291
Wygenerowane przykłady 292
Łącza 292
Ułożenie indeksu 292
Ulotki i artykuły 294
Łącza 294
Ułożenie indeksu 295
Artykuły niebędące ulotkami 295
Tryb programistyczny 296

Część VI. Obsługa techniczna i dystrybucja 299

20. Praktyki dotyczące tworzenia oprogramowania 301
Git i GitHub 302
Praktyka standardowa 302
Ciągła integracja 304
Akcje GitHuba 304
Wykonywanie polecenia R CMD check za pomocą akcji GitHuba 304
Inne zastosowania dla akcji GitHuba 305

21. Cykl życiowy 307
Ewolucja pakietu 308
Numer wersji pakietu 310
Konwencje numerów wersji pakietów tidyverse 312
Zachowanie wstecznej zgodności i przełomowe zmiany 312
Wersja główna, wersja mniejsza i wersja poprawki 314
Mechanizm wersji pakietu 315
Wady i zalety zmiany określanej jako przełomowa 316
Etapy cyklu życiowego i narzędzia wspomagające 317
Etapy cyklu życiowego i plakietki 317
Uznanie funkcji za przestarzałą 319

Uznanie argumentu za przestarzały 321
Komponent pomocniczy podczas uznawania za przestarzałe 321
Zmiana w zależności 322
Zastępowanie funkcji 324

22. Przekazanie pakietu do repozytorium CRAN 325
Określenie typu wydania 327
Początkowe wydanie poprzez repozytorium CRAN — kwestie specjalne 327
Polityki stosowane w repozytorium CRAN 329
Monitorowanie pod kątem zmian 329
Dokładne sprawdzenie wyniku wykonania polecenia R CMD check 331
Operacje sprawdzenia w repozytorium CRAN i powiązane z nimi usługi 332
Sprawdzanie zależności odwrotnych 333
Zależności odwrotne i przełomowe zmiany 336
Uaktualnienie komentarzy dla repozytorium CRAN 337
Proces przekazania pakietu do repozytorium CRAN 339
Tryby niepowodzenia 340
Świętowanie sukcesu 341