Spis treści

Słowo wstępne

Język R przeżył niezwykły wzrost popularności w ciągu ostatnich pięciu lat. Można by się spodziewać, że jest to nowy, dopiero co stworzony język. Jednak o dziwo, język R istnieje już od roku 1993. Skąd więc ten nagły wzrost popularności? Dość oczywistą odpowiedzią wydaje się być rozwinięcie nauki o danych (*data science*) jako obszaru badań i kariery osobistej. Jednak podstawy *data science* były wokół nas od dziesięcioleci. Statystyka, algebra liniowa, badania operacyjne, sztuczna inteligencja i uczenie maszynowe – wszystkie te dziedziny wnoszą swój wkład w narzędzia używane przez dzisiejszych badaczy danych. Język R, w odróżnieniu od innych języków, został zbudowany po to, aby większość tych narzędzi była oddalona tylko o jedno wywołanie funkcji.

To dlatego byłem tak bardzo podniecony tym, że Jared zdecydował się wrócić do swojego bestsellera i przedstawić uaktualnione, drugie wydanie, które prezentuje wiele najnowszych innowacji. Język R jest niezastąpiony w wielu zadaniach analizy danych. Wiele algorytmów użytecznych w prognozach i analizach jest dostępnych poprzez zaledwie parę wierszy kodu, co sprawia, że jest doskonale dopasowany do wyzwań stawianych przez dzisiejsze problemy. Nauka o danych jako dziedzina nie ogranicza się tylko do matematyki i statystyki, jak również nie sprowadza się do programowania i budowania infrastruktury.

Książka ta zapewnia dobre wprowadzenie w siłę i skuteczność języka R. Nie umiem wymyślić lepszego autora dla wchodzenia w świat R, niż Jared Lander. Po raz pierwszy poznałem go poprzez spotkanie środowiska uczenia maszynowego w Nowym Jorku pod koniec roku 2009. W tym czasie środowisko nowojorskie było dostatecznie małe, aby zmieścić się w jednej sali konferencyjnej, zaś wiele innych, dziś popularnych spotkań specjalistów od danych dopiero miało powstać. W ciągu tych siedmiu lat Jared stale znajdował się w pierwszym szeregu dopiero tworzącej się profesji badacza danych.

Poprzez prowadzenie Open Statistical Programming Meetup, wystąpienia na różnych spotkaniach i prowadzenie wykładów o R na Uniwersytecie Columbia Jared miał swój udział we wzroście naszej społeczności, ucząc programistów, badaczy danych, dziennikarzy i statystyków. Jednak jego umiejętności nie ograniczają się do nauczania. Jako codzienny praktyk, używa tych wszystkich narzędzi, pracując dla rozmaitych klientów – wielkich i małych. W czasie od pierwszego wydania jego książki Jared nadal robił wielką robotę w społeczności R, od organizowania New York R Conference, aż po opracowanie 2016 NFL Draft w języku R.

Książka ta stanowi wprowadzenie do programowania w R, ale również do rozmaitych metod statystycznych i narzędzi używanych codziennie przez programistów R. Przykłady wykorzystują publicznie dostępne zbiory danych, które zostały (na szczęście) wysprzątane przez Jareda i udostępnione na jego stronie Web.

– Paul Dix
Redaktor serii

Wprowadzenie

Wraz z rosnącym rozpowszechnieniem danych w naszym codziennym życiu potrzebne są nowsze i lepsze narzędzia do analizowania tego potopu. Tradycyjnie dostępne były rozwiązania wyznaczające przeciwstawne końce potencjalnego spektrum: lekkie, indywidualne analizy przy użyciu takich narzędzi jak Excel lub SPSS, albo ciężkie, wysokowydajne narzędzia analityczne zbudowane specjalnie w takich językach, jak FORTRAN lub C++. Jednak wraz z rosnącą siłą komputerów osobistych pojawiło się miejsce na pośrednie rozwiązania, które są zarazem interaktywne i wydajne. Analiza wykonana przez pojedynczą osobę na jej własnym komputerze w trybie eksploracyjnym może zostać szybko przekształcona w coś przeznaczonego dla serwera wspierającego zaawansowane procesy biznesowe. Ten właśnie pośredni obszar stał się domeną R, Pythona i innych języków skryptowych.

Język R, wymyślony przez Roberta Gentlemana i Rossa Ihaka z Uniwersytetu w Auckland w roku 1993, rozwinął się z języka S, którego twórcą był John Chambers z Bell Labs. Jest to język wysokiego poziomu, od początku zaprojektowany do działania interaktywnego, gdy użytkownik wywołuje polecenie, otrzymuje wyniki i wpisuje kolejne polecenie. Od tego czasu wyewoluował do poziomu języka, który może zostać wbudowany w różne systemy i kontrolować złożone zagadnienia.

Oprócz transformowania i analizowania danych język R pozwala łatwo tworzyć zadziwiające grafiki i raporty. Obecnie jest używany w całym przebiegu analiz danych, poczynając od wydobywania i transformowania danych, poprzez dopasowywanie modeli, wyciąganie wniosków i wykonywanie prognoz aż po tworzenie wykresów i generowanie raportów z wynikami.

Popularność języka R wystrzeliła pod koniec pierwszej dekady XXI wieku, gdy opuściwszy kręgi akademickie wkroczył na obszary bankowości, marketingu, farmacji, polityki, genetyki – by wymienić tylko kilka. Jego nowi użytkownicy są niezwykle zróżnicowani – są wśród nich i tacy, którzy wcześniej posługiwali się własnymi programami pisanymi w niskopoziomowych, kompilowanych językach, takich jak C++, użytkownicy innych pakietów statystycznych, na przykład SAS lub SPSS, no i oczywiście ci, którzy oswoili 800-funtowego goryla, czyli Excel. W tym czasie można było również zauważyć gwałtowny skok w liczbie dodanych pakietów, bibliotek wstępnie przygotowanego kodu rozszerzających funkcjonalność R.

Choć język R może niekiedy onieśmielać nowicjuszy, szczególnie tych bez doświadczeń programistycznych, moim zdaniem wykonywanie analizy poprzez programowanie zamiast wskazywania i klikania szybko okaże się łatwiejsze, wygodniejsze i bardziej niezawodne. I taki jest właśnie mój cel – aby ten proces nauki był łatwiejszy i szybszy.

Książka ta przedstawia informacje w taki sposób, w jaki chciałbym być uczony języka R w szkole średniej, gdyby w ogóle istniała szkoła ucząca tego języka. Wracając do podstaw, zawartość tej książki powstała na podstawie wykładów z *data science*, które prowadzę na Uniwersytecie Columbia. Nie jest jej celem przedstawienie każdego detalu języka R – byłoby to zresztą niemożliwe, gdyż nowe funkcjonalności i koncepcje powstają w takim tempie, że zmiany nastąpiłyby w czasie potrzebnym na lekturę, że nie wspomnę o znacznie dłuższym procesie pisania, redagowania i wydawania książki. Zamiast tego chcę pokazać może 20% funkcjonalności, ale te potrzebne do realizacji 80% pracy. Nie umieszczam w niej również wykładu ze statystyki – zakładam, że Czytelnik zna przynajmniej podstawowe pojęcia i ma doświadczenie w pracy z danymi. Oczekuję natomiast, że pokazane przeze mnie proste przykłady zostaną już przez samych czytelników rozwinięte dalej. Nie chodzi o to, aby poprzestać na tym, co pokazuję, ale aby samemu próbować szukać interesujących spostrzeżeń w otaczających nas danych.

Drugie wydanie tej książki zostało uaktualnione o narzędzia, które zostały wynalezione lub znacząco usprawnione od czasu pierwszego wydania. Najważniejsze wśród tych dodatków są pakiety **dplyr**, **tidyr** oraz **purrr** do przekształcania danych, stworzone przez Tidyverse. Dopasowywanie modeli zyskało więcej uwagi dzięki omówieniu wzmacnianych drzew i pakietu **caret**. Rozdział poświęcony **knitr** został podzielony na dwa, przy czym jeden zajmuje się połączeniem **knitr** i LaTeX, zaś drugi poświęciłem formatowi RMarkdown, który został znacząco ulepszony w ciągu ostatnich kilku lat, głównie ze względu na powstanie zestawu **htmlwidgets**, pozwalającego na włączanie do dokumentów kodu JavaScript. Cały rozdział poświęciłem Shiny, nowemu narzędziu do tworzenia interakcyjnych tablic kontrolnych opartych na Web. Rozdział na temat pisania pakietów został uzupełniony o zagadnienia testowania kodu, zaś rozdział o wczytywaniu danych obejmuje nowe techniki czytania wykorzystujące pakiety **readr**, **readxl** i **jsonlite**.

Treść została podzielona na samodzielne rozdziały, jak poniżej.

Rozdział 1, „Poznajemy R", pokazuje, skąd można pobrać język R i jak go zainstalować w różnych systemach operacyjnych, zarówno w 32-, jak 64-bitowych wersjach.

Rozdział 2, „Środowisko R", zawiera wstęp do posługiwania się językiem R, a w szczególności konfigurowanie RStudio. Omówione są tam również projekty RStudio i integracja z systemem kontroli wersji Git.

Rozdział 3, „Pakiety R", poświęcony jest wyszukiwaniu, instalowaniu i ładowaniu pakietów (bibliotek) R.

Rozdział 4, „Podstawy R", pokazuje podstawowe funkcjonalności arytmetyczne R. Przedstawia elementarne typy danych, takie jak typ liczbowy (`numeric`), tekstowy (`character`) i do przechowywania dat (`date`), a także pojęcie wektora (`vector`) i jego szczególniej odmiany specyficznej dla R, jaką jest `factor`. Zawarta jest tam również krótka instrukcja, jak wywoływać funkcje i szukać ich dokumentacji.

W rozdziale 5, „Zaawansowane struktury danych", omówiona jest najbardziej uniwersalna i najczęściej używana struktura danych, jaką jest ramka danych (`data.frame`), a także macierze, tablice i listy.

Rozdział 6, „Wczytywanie danych do R", zgodnie z tytułem zajmuje się pobieraniem danych. Zanim dowolne dane będzie można analizować, muszą one zostać wczytane do środowiska. Istnieje wiele sposobów pobierania danych, w tym czytanie z plików o rozdzielanych wartościach (CSV) i baz danych.

Rozdział 7, „Grafika statystyczna", pokazuje jasno, dlaczego grafiki są kluczową częścią zarówno we wstępnej analizie danych, jak i w komunikowaniu wyników. Język R umożliwia tworzenie przepięknych wykresów dzięki potężnym narzędziom graficznym. W rozdziale tym przedstawione są zarówno podstawowe funkcje graficzne, jak i potężny pakiet **ggplot2**.

Rozdział 8, „Pisanie funkcji R", pokazuje tworzenie funkcji definiowanych przez użytkownika, które ułatwiają wykonywanie często powtarzanych operacji.

Rozdział 9, „Wyrażenia sterujące", przedstawia techniki kontrolowania przebiegu programów przy użyciu dyrektyw `if`, `ifelse` i złożonych testów.

Rozdział10, „Pętle – nie-R metoda iteracji" pokazuje wykonywanie iteracji przy użyciu pętli `for` i `while`. Choć takie podejścia są w ogólności niezalecane, niekiedy są niezbędne i trzeba je znać.

Rozdział 11, „Manipulacje grupowe", przedstawia wektoryzację – lepszą alternatywę dla pętli. Wektoryzacja powoduje, że operacje wykonywane są na wszystkich elementach wektora jednocześnie, a nie poprzez iterowanie przez jego elementy, przez co jest to znacznie wydajniejsze. Rozdział zawiera też omówienie rodziny funkcji **apply** i pakietu **plyr**.

Rozdział 12, „Szybsze manipulacje grupowe przy użyciu **dplyr**", przedstawia nową generację funkcji manipulacji grupowych zawartą w pakiecie **dplyr**. Pakiet ten został zoptymalizowany do pracy z ramkami danych i wykorzystuje potoki w celu wydajniejszego i łatwiejszego w czytaniu kodowania.

Rozdział 13, „Iteracje przy użyciu **purrr**", pokazuje inną alternatywę dla pętli, czyli pakiet **purrr**, umożliwiający szybkie iteracje poprzez listy i wektory. Reprezentuje on powrót do funkcjonalnych korzeni języka R.

Rozdział 14, „Kształtowanie danych", poświęcony jest zagadnieniom scalania wielu zbiorów danych poprzez ich sklejanie lub złączanie, co jest często równie niezbędne, jak zmiana ich ukształtowania. Pakiety **plyr** i **reshape2** udostępniają wygodne funkcje

realizacji takich zadań, uzupełniające podstawowe narzędzia, takie jak **rbind**, **cbind** i **merge**.

W rozdziale 15, „Kształtowanie danych w Tidyverse", prezentuję kolejny przykład ewolucji pakietów i wprowadzenie pakietów **dplyr** i **tidyr** jako zamienników **plyr** i **reshape2** w zadaniach scalania i przekształcania danych.

Rozdział16, „Manipulowanie ciągami znaków", poświęcony jest pracy z tekstem. Większość ludzi nie kojarzy danych znakowych ze statystyką, ale są one ważną częścią danych. Język R udostępnia liczne narzędzia do pracy z ciągami znaków, pozwalajacych na ich łączenie i wydzielanie informacji, a także wyszukiwanie poprzez wyrażenia regularne.

Rozdział 17, „Rozkłady prawdopodobieństwa", stanowi szybki przegląd najpopularniejszych rozkładów prawdopodobieństwa, w tym rozkładu normalnego, dwumianowego i Poissona, wraz z odpowiadającymi im funkcjami dostępnymi w języku R.

Rozdział 18, „Podstawowe statystyki", przedstawia statystyki danych, które są zazwyczaj poznawane jako pierwsze, takie jak średnia, odchylenie standardowe lub test t Studenta.

W rozdziale 19, „Modele liniowe", przedstawione są funkcje do konstruowania najbardziej skutecznych i powszechnie używanych narzędzi statystycznych – modeli liniowych.

Rozdział 20, „Uogólnione modele liniowe", pokazuje rozszerzenie klasycznych modeli liniowych, aby uwzględniały regresję logistyczną i Poissona. Dodatkowo rozdział ten omawia również analizę przeżycia.

Rozdział 21, „Diagnostyka modelu", prezentuje metody oceniania jakości modeli i wybierania zmiennych przy użyciu technik resztowych, AIC, sprawdzianów krzyżowych, bootstrapu i krokowego wybierania.

Rozdział 22, „Regularyzacja i ściąganie", omawia techniki unikania nadmiernego dopasowania (przeuczenia modelu) wykorzystujące metody sieci elastyczne i statystykę bayesowską.

Rozdział 23, „Modele nieliniowe", omawia te przypadki, w których modele liniowe są niewłaściwe i konieczne jest wybranie modelu nieliniowego. Przedstawione w nim są techniki najmniejszych kwadratów, krzywych sklejanych, uogólnione modele addytywne, drzewa decyzyjne i lasy losowe.

Rozdział 24, „Serie czasowe i autokorelacja" pokazuje metody analizy jedno- i wielowymiarowych serii czasowych.

Rozdział 25, „Grupowanie", pokazuje techniki wykonywania analizy skupień różnymi metodami, począwszy od techniki k-średnich po grupowanie hierarchiczne.

Rozdział 26, „Dopasowywanie modelu przy użyciu **caret**", wprowadza pakiet **caret** umożliwiający zautomatyzowane dostrajanie modelu. Pakiet ten oferuje również ujednolicony interfejs dla setek różnych modeli, znacznie ułatwiając proces analizy.

Rozdział 27, „Reprodukowalność i raporty przy użyciu knitr", poświęcony jest integrowaniu kodu R i uzyskiwanych wyników z raportami tworzonymi w języku LaTeX.

Rozdział 28, „Tworzenie bogatych dokumentów przy użyciu RMarkdown", prezentuje techniki szybkiego i prostego pisania efektownych raportów, pokazów slajdów i stron Web z poziomu języka R. Omówione jest również zapewnienie interaktywności tych stron przy użyciu narzędzi z rodziny htmlwidgets, taki jak leaflet i dygraphs.

Rozdział 29, „Tworzenie interaktywnych tablic kontrolnych przy użyciu Shiny" przedstawia platformę Shiny pozwalającą na generowanie opartych na Web tablic kontrolnych wykorzystujących całą siłę języka R.

Rozdział 30, „Tworzenie pakietów R", pokazuje zasady konstruowania pakietów R jako sposobu zapewnienia przenośności i ponownego użycia kodu. Budowanie takich pakietów zostało ostatnio znacznie ułatwione dzięki pojawieniu się narzędzi devtools i Rcpp.

Dodatek A, „Praktyczne zasoby", wylicza polecane zasoby internetowe i nie tylko, które mogą być pomocne w poznawaniu języka R i kontaktach ze społecznością użytkowników R.

Dodatek B, „Słowniczek", zawiera wyjaśnienia większości terminów używanych w książce.

Znaczącą część tekstu tej książki stanowi albo kod języka R, albo rezultaty uruchomienia kodu. Dla poprawy czytelności kod i wyniki zostały złożone inną odmianą czcionki, jak poniżej. Aby uzyskać efekt możliwie podobny do widocznego w konsoli R, wiersze kodu R rozpoczynają się znakiem > lub znakiem + w razie kontynuowania polecenia. Trzeba tu podkreślić, że znaków tych *nie należy* wpisywać w konsoli R – są automatycznie dodawane w trakcie pisania kodu. Treść poleceń jest wytłuszczona, zaś komentarze złożone są kursywą. Wyniki są składane zwykłą odmianą kroju stałopozycyjnego, zaś komunikaty zwracane przez system (ostrzeżenia, błędy itp.) – również kursywą (w konsoli R są one wyróżniane kolorem, którym jednak nie dysponujemy w tej książce).

```
> # to jest komentarz
>
> # podstawowa arytmetyka
> 10 * 10
[1] 100

> # Wywołanie funkcji
> sqrt(4)
[1] 2
```

Fragmenty kodu (nazwy i wartości zmiennych, argumentów i tak dalej) wewnątrz zwykłego tekstu są przedstawiane następująco: `object1`. Nazwy **pakietów** złożone zostały czcionką bezszeryfową w kolorze szarym, zaś nazwy **funkcji** tą samą czcionką, ale czarną.

W tych miejscach, w których potrzebne są równania matematyczne, są one umieszczane w oddzielnych wersach i numerowane.

$$e^{i\pi} + 1 = 0 \tag{1}$$

W równaniach zwykłe zmienne są złożone kursywą (x), wektory są wytłuszczonymi małymi literami ($\mathbf{x}$), zaś macierze wielkimi wytłuszczonymi literami ($\mathbf{X}$). Greckie litery, takie jak α i β, stosują tę samą konwencję.

Nauka R to wspaniałe doświadczenie, które dodatkowo może sprawić, że wykonanie tak wielu zadań stanie się łatwiejsze. Mam nadzieję, że czytelnikom spodoba się nauka w moim towarzystwie.

O autorze

Jared P. Lander jest naczelnym badaczem danych w Lander Analytics, zlokalizowanej w Nowym Jorku firmie specjalizującej się w konsultacjach statystycznych i szkoleniach. Jest również organizatorem New York Open Statistical Programming Meetup – największego w świecie spotkania użytkowników R – oraz New York R Conference. Jest też adiunktem w dziedzinie statystyki na Uniwersytecie Columbia. W wolnych chwilach pełni też rolę przewodnika w Scott's Pizza Tours. Uzyskał licencjat z matematyki na Muhlenberg College i magisterium ze statystyki na Uniwersytecie Columbia. Ma znaczące doświadczenie zarówno w badaniach akademickich, jak i praktycznych zastosowaniach statystyki. Jared często zabiera głos na różnorodnych konferencjach, takich jak Strata czy MIT Sloan Sports Analytics Conference oraz spotkaniach na całym świecie. Teksty jego autorstwa można znaleźć w witrynie jaredlander.com. Jego prace były zamieszczane w rozmaitych mediach, między innymi w dzienniku CBS i w *Wall Street Journal*.

1
Poznajemy R

Język R jest wspaniałym narzędziem dla analizy statystycznej, wizualizacji i raportowania. Jego użyteczność najlepiej można poznać, przyglądając się różnorodności obszarów, w których jest używany. Osobiście używałem już języka R w projektach tworzonych dla banków, kampanii politycznych, firm technologicznych i żywieniowych, międzynarodowych organizacji rozwoju i pomocy, szpitali i deweloperów zabudowy mieszkaniowej. Inne obszary, w których można zauważyć jego stosowanie, obejmują reklamę online, ubezpieczenia, ekologię, genetykę czy branżę farmaceutyczną. R jest używany przez statystyków dysponujących zaawansowanymi umiejętnościami nauczania maszynowego i przez programistów znających inne języki, ale również przez osoby, które niekoniecznie mają przeszkolenie w zaawansowanej analizie danych, ale które są zmęczone używaniem Excela.

Zanim jednak będzie można zacząć go używać, trzeba go pobrać i zainstalować. Szczęśliwie ten proces nie jest bardziej skomplikowany, niż instalowanie dowolnego innego programu.

1.1 Pobieranie R

Pierwszym krokiem na drodze do używania języka R jest umieszczenie go na naszym komputerze. W odróżnieniu od takich języków jak C++, R musi zostać zainstalowany przed uruchomieniem*. Program jest łatwo dostępny w repozytorium Comprehensive R Archive Network (CRAN), organizacji utrzymującej język R, dostępnej pod adresem *http://cran.r-project.org/*. W górnej części strony powitalnej znajdziemy łącza pozwalające pobrać R dla Windows, Mac OS X oraz Linuxa.

Dostępne są wstępnie skompilowane instalacje dla systemów Windows i Mac OS X, podczas gdy te przeznaczone dla Linuxa zazwyczaj są kompilowane z plików źródłowych. Instalowanie R na dowolnej z tych platform nie różni się od instalowania dowolnego innego programu.

---

* Technicznie biorąc, języka C++ nie można używać samodzielnie (bez kompilatora), zatem coś nadal musi zostać zainstalowane.

Użytkownicy Windows powinni kliknąć łącze „Download R for Windows", następnie wybrać „base", a na koniec „Download R 3.x.x for Windows"; litery *x* wskazują bieżącą wersję języka R. Wersja ta zmienia się okresowo w miarę dokonywania usprawnień.

Analogicznie użytkownicy Maców powinni kliknąć „Download R for (Mac) OS X", a następnie „R-3.x.x.pkg"; także tu *x* oznacza bieżącą dostępną wersję R. Spowoduje to zainstalowanie zarówno 32-bitowej, jak i 64-bitowej wersji.

Użytkownicy systemu Linux powinni pobrać R przy użyciu mechanizmu standardowego dla używanej dystrybucji, a więc np. apt-get (Ubuntu i Debian), yum (Red Hat), zypper (SUSE) lub innego źródła. Spowoduje to również skompilowanie i zainstalowanie języka R.

1.2 Wersja R

W chwili pisania tych słów najnowsza dostępna wersja R miała oznaczenie 3.4.0 i obejmowała bardzo liczne usprawnienia od czasu pierwszego wydania książki, gdy dostępna była wersja 3.0.1. CRAN wykorzystuje roczny cykl publikowania, przy czym każda znacząca zmiana wersji zwiększa środkową spośród trzech wartości identyfikujących wersję. Przykładowo wersja 3.2.0 została opublikowana w roku 2015. W roku 2016 numer wersji został zwiększony do 3.3.0, zaś wersję 3.4.0 udostępniono w roku 2017. Ostatnia część numeru wersji odróżnia pomniejsze aktualizacje aktualnej wersji zasadniczej. Większość funkcjonalności języka R jest zazwyczaj kompatybilna wstecz z wersjami poprzednimi.

1.3 Wersja 32- czy 64-bitowa

Wybór pomiędzy używaniem wersji 32- lub 64-bitowej sprowadza się do tego, czy komputer wspiera kod 64-bitowy – co jest prawdą dla większości nowszych maszyn – i wielkości danych, z którymi zamierzamy pracować. Wersje 64-bitowe mogą adresować zdecydowanie większe ilości pamięci (RAM), zatem więcej jej można użyć.

Jest to szczególnie istotne poczynając od wersji 3.0.0, w której dodano obsługę 64-bitowych liczb całkowitych, co oznacza, że w obiektach R można przechowywać znacznie większe ilości danych.

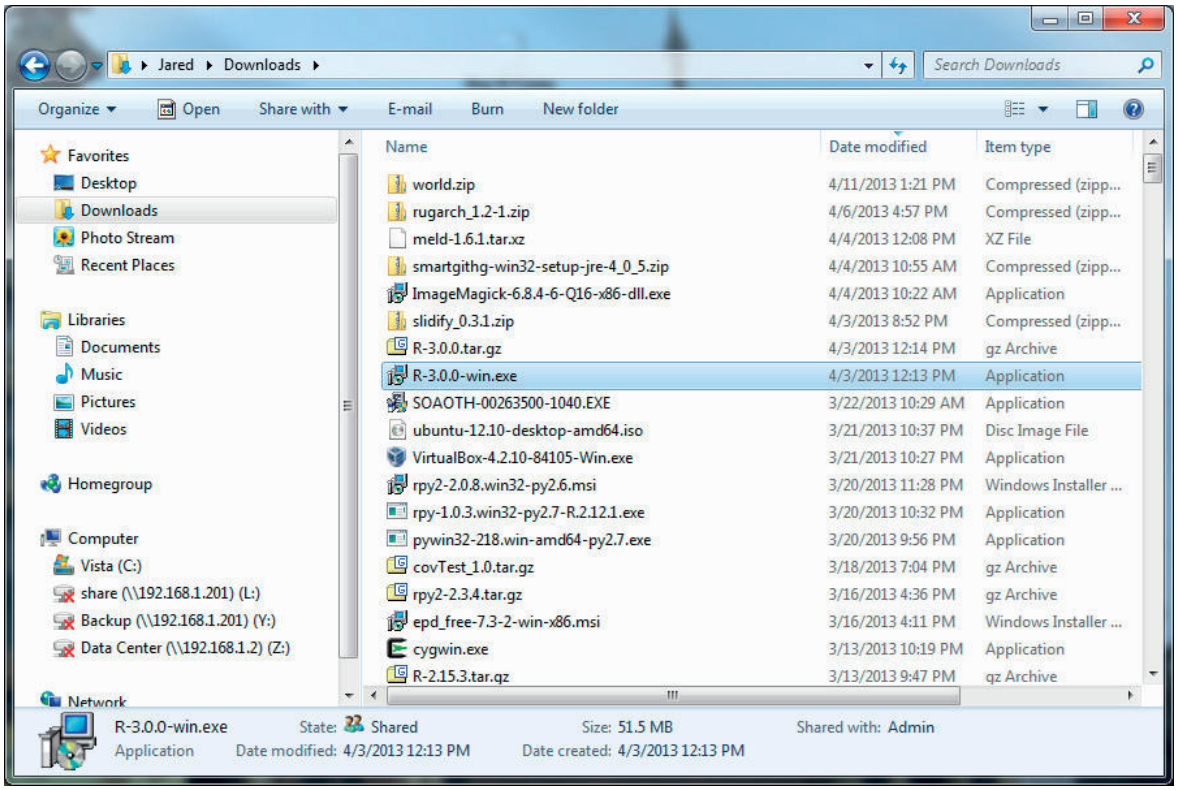
W przeszłości niektóre pakiety wymagały stosowania 32-bitowej wersji R, ale obecnie jest to niezwykle rzadkie. Jedynym powodem instalowania wersji 32-bitowej jest obecnie potrzeba obsłużenia jakiś starszych pakietów analitycznych lub potrzeba używania jej na komputerze wyposażonym w procesor 32-bitowy, taki jak energooszczędny układ Atom firmy Intel.

1.4 Instalowanie

Instalowanie R w systemie Windows lub na Macu odbywa się tak samo, jak instalacja dowolnego innego programu.

1.4.1 Instalowanie w systemie Windows

Odszukaj odpowiedni instalator w miejscu, do którego został pobrany. Dla użytkownika będzie on wyglądał podobnie jak na rysunku 1.1.



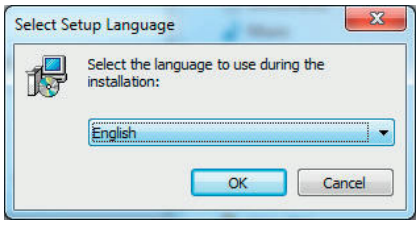
Rysunek 1.1 Lokalizacja instalatora R

Język R należy instalować przy użyciu przywilejów administratora. Oznacza to konieczność kliknięcia instalatora prawym klawiszem myszy i wybranie polecenia Run as Administrator (Uruchom jako administrator). Wywoła to monit, w którym konieczne może być wpisanie hasła administratora.

Pierwsze okno dialogowe, pokazane na rysunku 1.2, udostępnia wybór języka instalacji, którym domyślnie jest język zgodny z językiem systemu operacyjnego, ale można wybrać inny z długiej listy dostępnych*. Po wybraniu właściwego języka należy kliknąć OK.

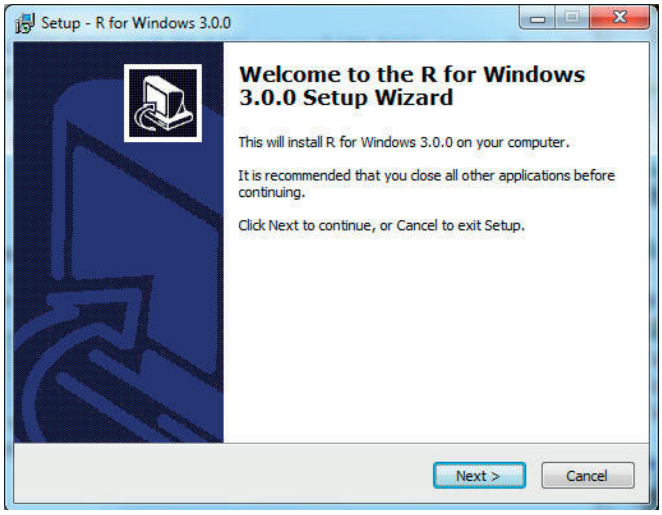
---

* Lista ta zawiera również język polski. Trzeba jednak zauważyć, że choć menu konsoli i część komunikatów została spolonizowana, nie obejmuje to wszystkich informacji zwrotnych, co powoduje dość



Rysunek 1.2 Wybieranie języka dla Windows

Następnie pojawi się ostrzeżenie pokazane na rysunku 1.3, zalecające zamknięcie wszystkich innych programów na czas instalacji. To zalecenie w istocie nie jest już ani niezbędne, ani przestrzegane, zatem można tu po prostu kliknąć Next (Dalej).



Rysunek 1.3 W nowych wersjach systemu Windows tę sugestię można zignorować.

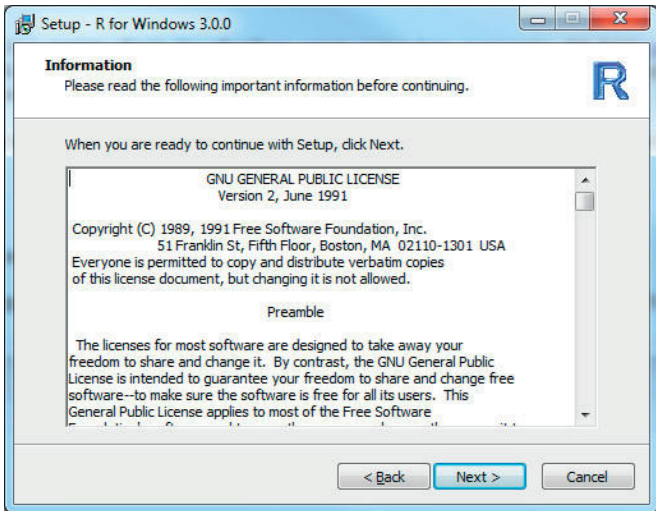
W kolejnym kroku zostaje wyświetlona licencja oprogramowania, pokazana na rysunku 1.4. Języka R nie można używać bez wyrażenia zgody na te (ważne) warunki licencyjne, zatem jedyną możliwością jest kliknięcie Next.

Instalator następnie zapyta o lokalizację docelową. Wprawdzie oficjalne zalecenie CRAN głosi, że R powinien być instalowany w katalogu nie zawierającym spacji, niemal w połowie przypadków domyślnym katalogiem instalacji jest `Program Files\R`, co może spowodować problemy, jeśli później spróbujemy budować pakiety wymagające skompilowanego kodu, napisanego w takich językach, jak C++ lub FORTRAN. To okno dialogowe pokazane jest na rysunku 1.5. Ważne jest, aby wybrać katalog nie zawierający spacji w nazwie, nawet jeśli domyślnie proponowana lokalizacja instalacji sugeruje co innego.

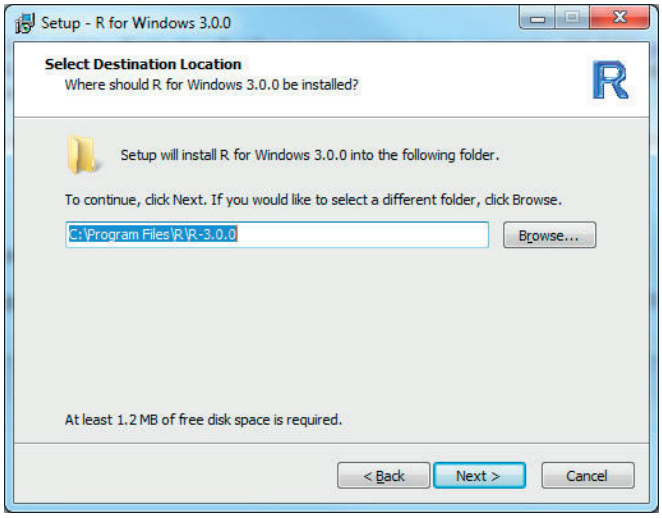
Jeśli mamy do czynienia z taką sytuacją, można kliknąć przycisk Browse (Przeglądaj), aby wyświetlić opcje folderów pokazane na rysunku 1.6.

---

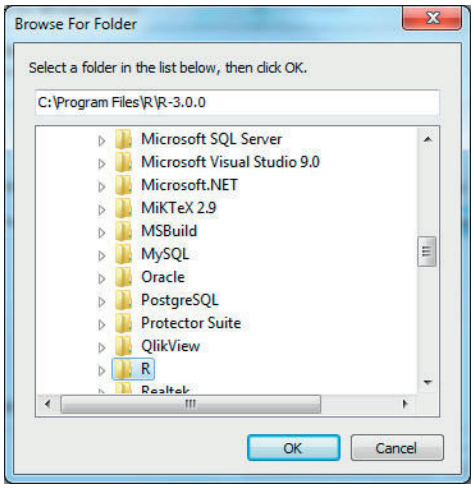
kłopotliwą „mieszaninę" języków. Co więcej, RStudio, na którego używaniu skupia się ta książka, nie jest spolonizowane zupełnie, zatem zapewne lepszym wyborem będzie użycie interfejsu w języku angielskim (przyp. tłum.).



Rysunek 1.4 Warunki licencji trzeba zaakceptować przed rozpoczęciem używania języka R.

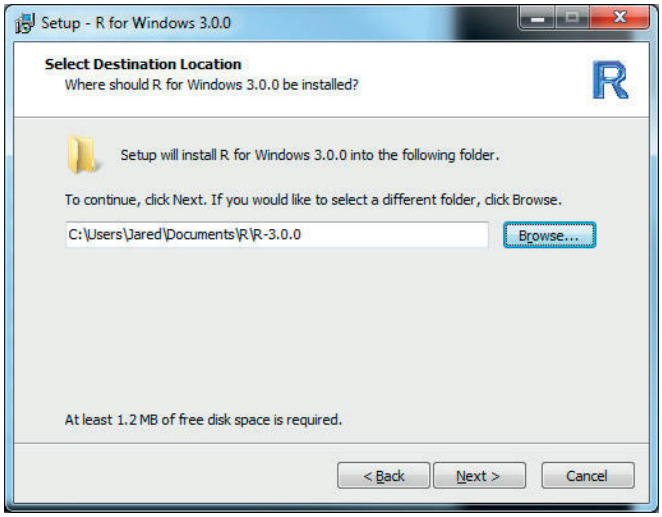


Rysunek 1.5 Lepiej wybrać folder docelowy, którego nazwa nie zawiera spacji.



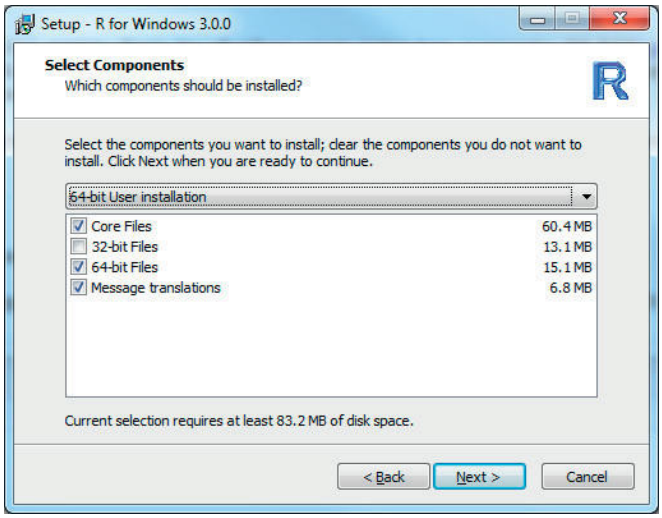
Rysunek 1.6 To okno dialogowe umożliwia wybranie docelowego folderu.

Zalecane jest wybranie folderu docelowego zlokalizowanego wewnątrz folderu `My Documents` (zazwyczaj na dysku `C:`, choć niekiedy może być to inny napęd), który – niezależnie od tego, co pokazuje przyjazna nazwa, w rzeczywistości jest katalogiem `C:\Users\<UserName>\Documents`, nie zawierającym spacji w nazwie. Rysunek 1.7 pokazuje właściwe miejsce docelowe instalacji.



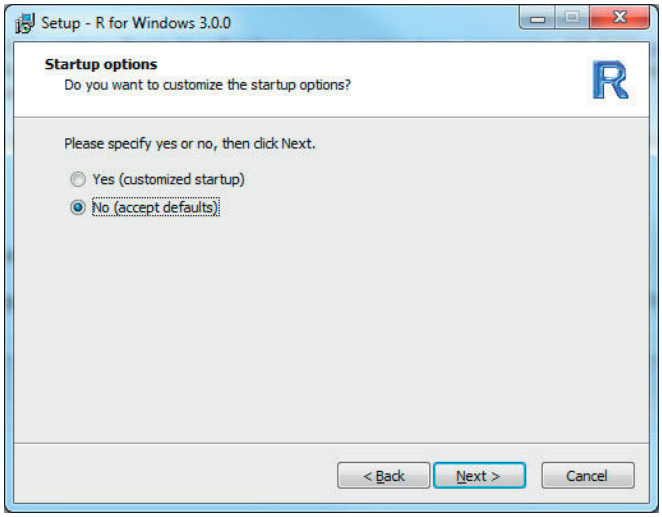
Rysunek 1.7 Właściwe miejsce docelowe, bez spacji w nazwie katalogu.

W kolejnym kroku, pokazanym na rysunku 1.8, wyświetlona jest lista komponentów do zainstalowania. O ile nie występuje potrzeba uwzględnienia 32-bitowych plików, opcję tę można bezpiecznie wyczyścić. Wszystkie pozostałe elementy należy pozostawić zaznaczone.



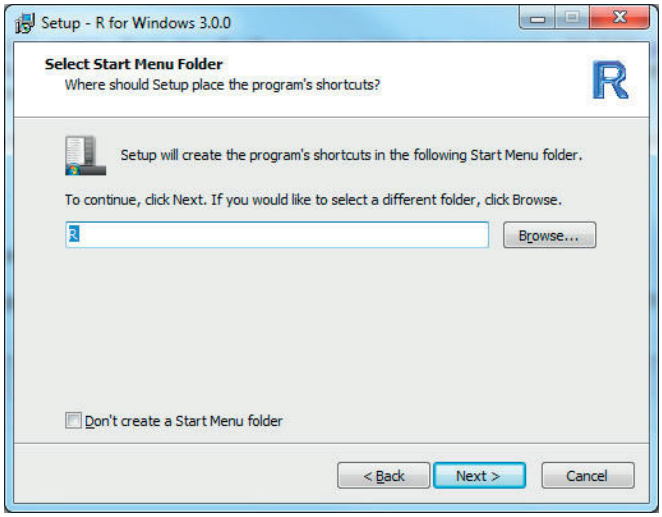
Rysunek 1.8 Najlepiej zaznaczyć wszystko poza komponentami 32-bitowymi.

Na kolejnej stronie Startup options (Opcje uruchomieniowe) najlepiej pozostawić domyślną odpowiedź No (Nie), jak na rysunku 1.9, gdyż w istocie nie ma zbyt wielu opcji do wyboru, a osobiście i tak zalecam używanie RStudio jako frontonu.



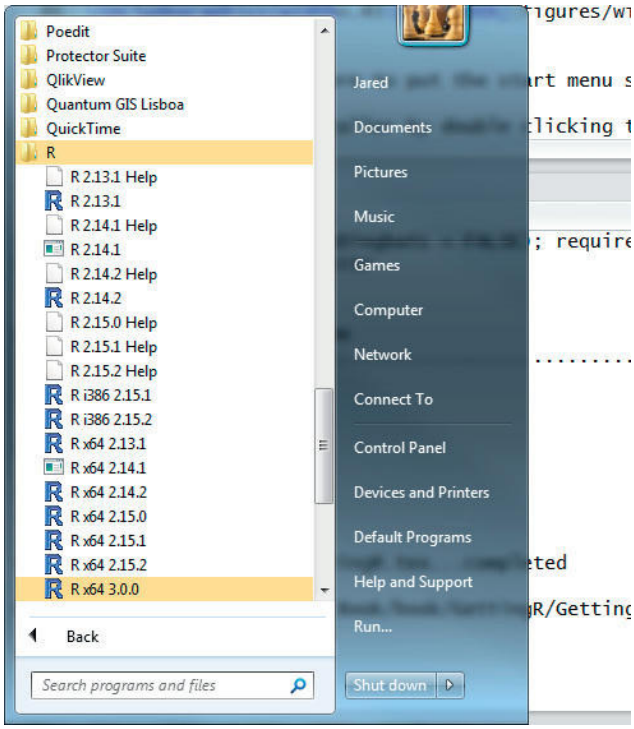
Rysunek 1.9 Można zaakceptować domyślne opcje uruchomieniowe, gdyż będziemy używać RStudio jako frontonu i opcje te nie będą miały większego znaczenia.

W kolejnym kroku należy określić, gdzie mają zostać umieszczone skróty menu start. Zalecam użycie folderu o nazwie R i umieszczanie w nim każdej wersji, jak na rysunku 1.10.



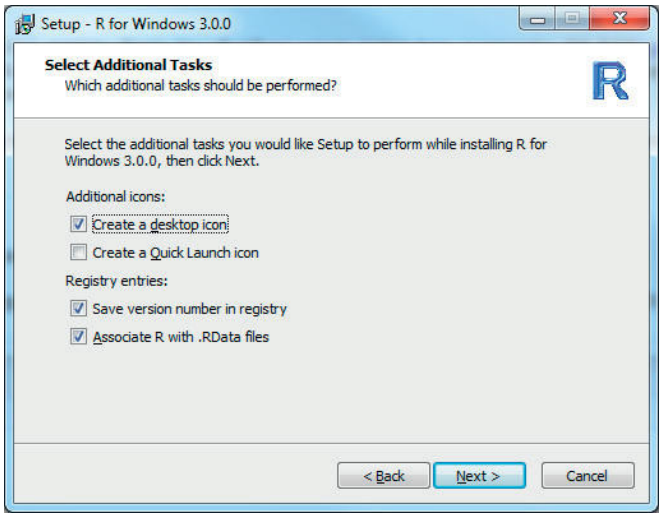
Rysunek 1.10 Wybieranie folderu menu Start, w którym zostaną umieszczone skróty.

Osobiście używam wielu wersji języka R, wszystkich umieszczonych w tym samym folderze menu Start, co pozwala na łatwe testowanie kodu w różnych wersjach. Można to zobaczyć na rysunku 1.11.



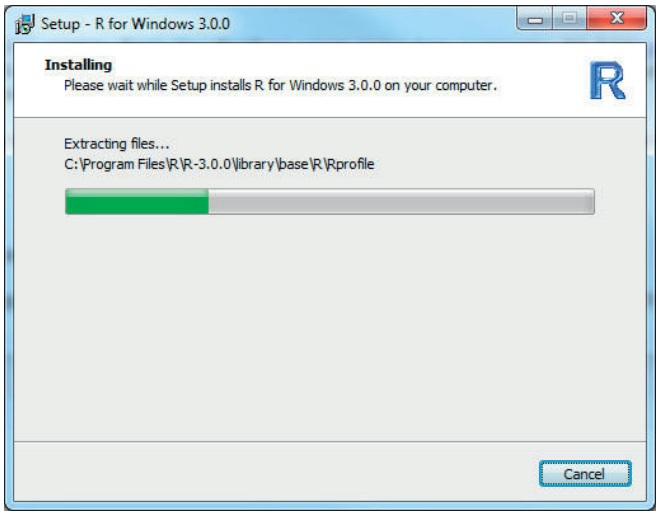
Rysunek 1.11 Wiele zainstalowanych wersji języka R umożliwia projektowanie i testowanie kodu w różnych wersjach.

Ostatnią opcją jest wybór kilku dodatkowych zadań, takich jak tworzenie ikony na pulpicie (niezbyt użyteczne, jeśli będziemy używać RStudio). Zdecydowanie polecam zapisanie numeru wersji w rejestrze i skojarzenie R z plikami `RData`. Opcje te są pokazane na rysunku 1.12.



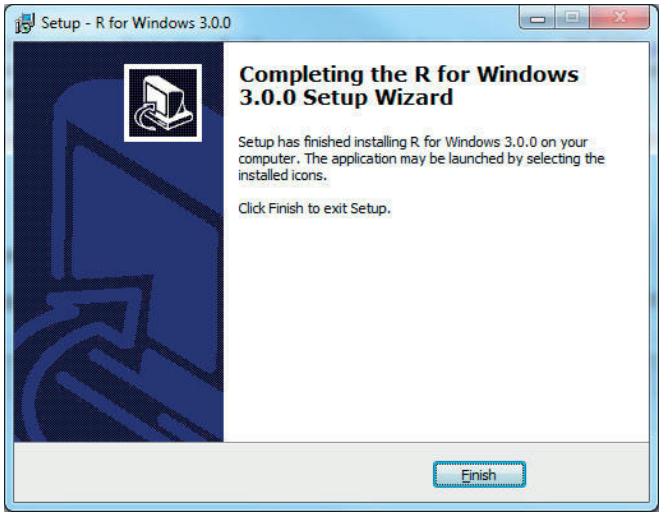
Rysunek 1.12 Zalecam zapisanie numeru wersji w rejestrze i skojarzenie R z plikami `RData`.

Kliknięcie Next rozpoczyna instalację i wyświetla pasek postępu, jak na rysunku 1.13.



Rysunek 1.13 Pasek postępu wyświetlany w trakcie instalacji.

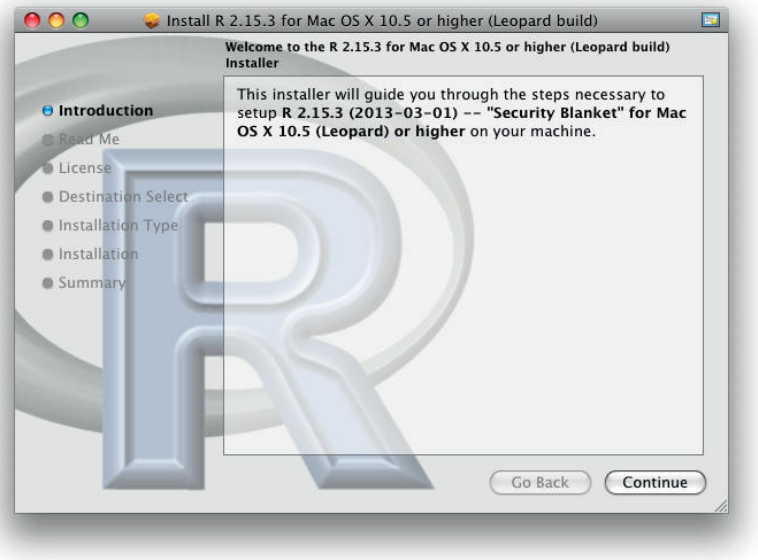
Ostatnim krokiem, pokazanym na rysunku 1.14, jest kliknięcie Finish, potwierdzające zakończenie instalacji.



Rysunek 1.14 Potwierdzenie, że instalacja jest zakończona.

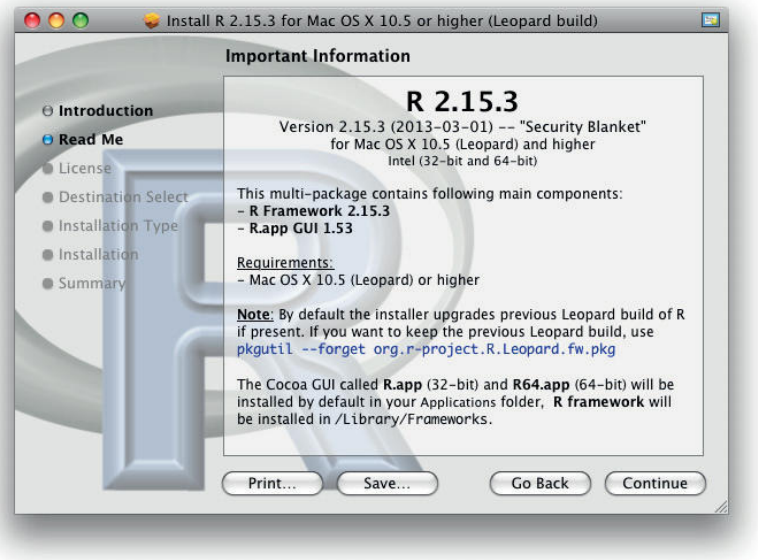
1.4.2 Instalowanie w Mac OS X

Po pobraniu należy odszukać plik instalatora o rozszerzeniu `.pkg`, po czym uruchomić go podwójnym kliknięciem. Spowoduje to wyświetlenie ekranu powitalnego, pokazanego na rysunku 1.15. W celu rozpoczęcia instalacji klikamy Continue.



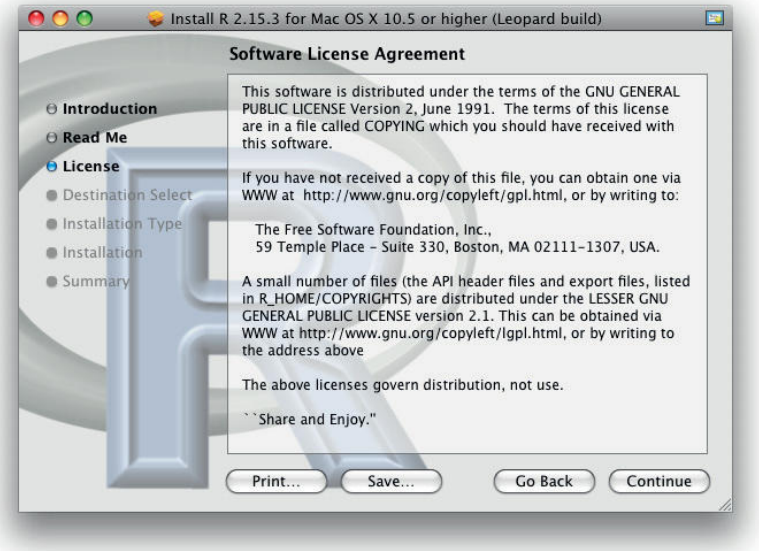
Rysunek 1.15 Powitalny ekran instalacji na Macu

Zostanie wyświetlonych kilka informacji o instalowanej wersji R. Nie ma tu nic do zrobienia poza kliknięciem Continue, jak na rysunku 1.16.



Rysunek 1.16 Informacje o wersji

W kolejnym kroku wyświetlane są informacje o licencji dystrybucyjnej, pokazane na rysunku 1.17. Jedyną możliwością, aby móc używać R, jest ich zaakceptowanie kliknięciem przycisku Continue.

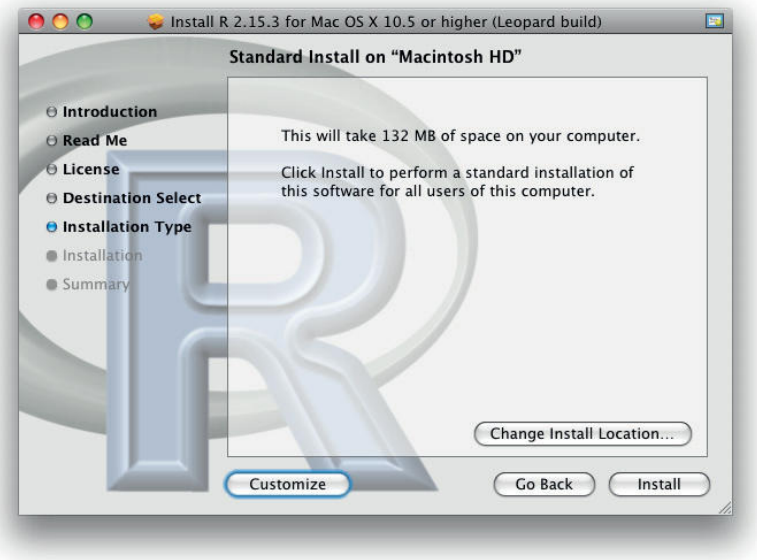


Rysunek 1.17 Porozumienie licencyjne

W kolejnym kroku trzeba wyrazić zgodę na warunki używania oprogramowania, klikając Agree, jak na rysunku 1.18.

Rysunek 1.18 Wymagane jest jawne wyrażenie zgody na warunki licencji.

Aby zainstalować R dla wszystkich użytkowników, wystarczy kliknąć Install; w przeciwnym razie należy użyć przycisku Change Install Location (Zmień lokalizację instalacji), aby zmienić miejsce, jak na rysunku 1.19.

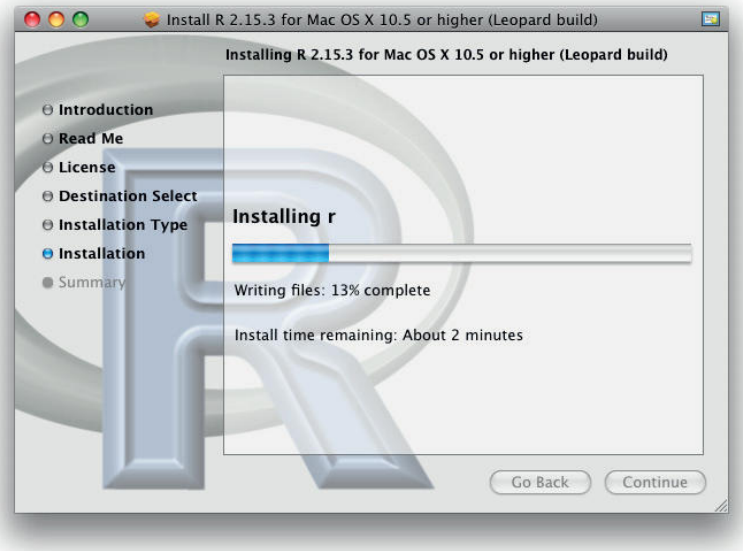


Rysunek 1.19 Domyślnie R jest instalowane dla wszystkich użytkowników, choć dostępna jest opcja wybrania określonej lokalizacji.

Jeśli pojawi się monit, trzeba wpisać hasło, jak na rysunku 1.20.

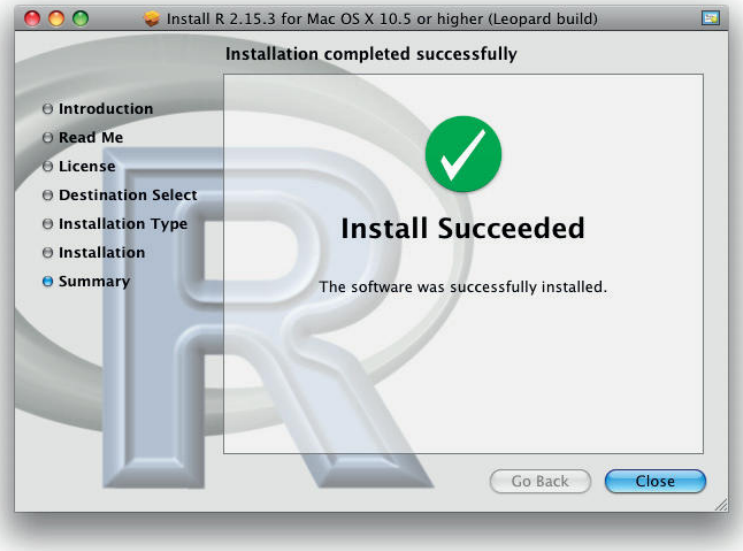
Rysunek 1.20 Wykonanie instalacji może wymagać podania hasła administratora.

Spowoduje to rozpoczęcie procesu instalacyjnego i wyświetlenie paska postępu, jak rysunku 1.21.



Rysunek 1.21 Postęp instalacji jest sygnalizowany paskiem postępu.

Po zakończeniu instalator zasygnalizuje sukces, jak na rysunku 1.22. Kliknięcie Close zakończy instalację.



Rysunek 1.22 Instalacja zakończona sukcesem.

1.4.3 Instalowanie w systemach Linux

Uzyskanie R ze standardowego mechanizmu dystrybucji powoduje pobranie, skompilowanie i zainstalowanie R w jednym kroku. W tym rozdziale skupię się na Ubuntu, używającym mechanizmu `apt-get`.

Pierwszym rokiem jest edycja pliku `/etc/apt/sources.list`, zawierającego listę źródeł pakietów. Trzeba do niego dołączyć dwa elementy informacji: używany mirror CRAN oraz wersję Ubuntu lub Debiana.

Można użyć dowolnego mirroru CRAN, zatem zdecydowałem się na wybranie mirroru RStudio dostępnego pod adresem *http://cran.rstudio.com/bin/linux/ubuntu.*

Obsługiwane wersje Ubuntu według stanu na początek roku 2017 to Yakkety Yak (16.10), Xenial Xerus (16.04), Wily Werewolf (15.10), Vivid Vervet (15.04), Trusty Tahr (14.04; LTS) oraz Precise Pangolin (12.04; LTS)*.

Aby zainstalować R przy użyciu mirroru RStudio w systemie Ubuntu 16.04, trzeba dodać wiersz

```
deb http://cran.rstudio.com/bin/linux/ubuntu xenial/
```

do pliku `/etc/apt/sources.list`. Można to zrobić ręcznie lub wywołując poniższe polecenie w terminalu.

```
sudo sh -c \
'echo "deb http://cran.rstudio.com/bin/linux/ubuntu xenial/" \
>> /etc/apt/sources.list'
```

Następnie trzeba dodać klucz publiczny uwierzytelniający pakiety.

```
sudo apt-key adv --keyserver keyserver.ubuntu.com
--recv-keys E084DAB9
```

Teraz można zaktualizować `apt-get` i zainstalować R. Zalecane jest zainstalowanie zarówno pakietu podstawowego (base), jak i deweloperskiego (base-dev), aby możliwe było kompilowanie pakietów ze źródeł i budowanie swoich własnych.

```
sudo apt-get update
sudo apt-get install r-base
sudo apt-get install r-base-dev
```

Oprócz Ubuntu, język R jest również natywnie wspierany w dystrybucjach Debian, Red Hat i SuSE.

1.5 Microsoft R Open

Firma Microsoft, która kupiła Revolution Analytics, oferuje społecznościową wersję swojej własnej kompilacji R, nazywaną Microsoft R Open, która udostępnia środowisko IDE (Integrated Development Environment) oparte na Visual Studio. Wersja

* Zgodnie z zawartością pliku *https://cran.r-project.org/bin/linux/ubuntu/README.*

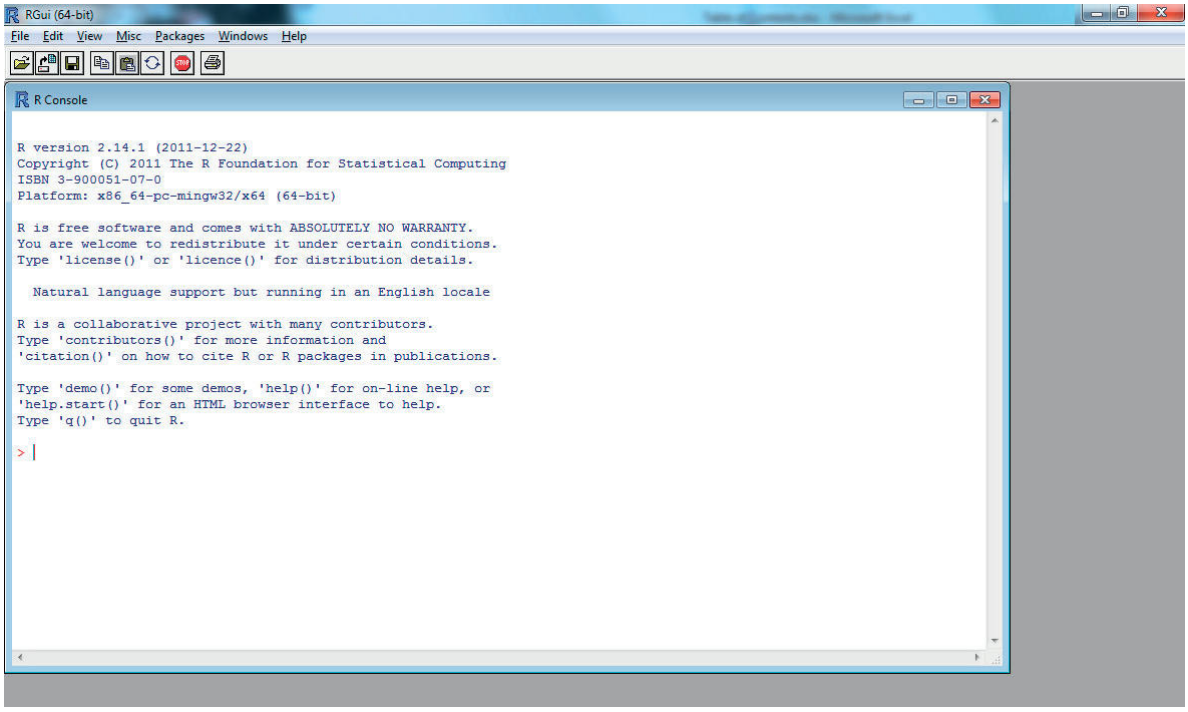
ta jest skompilowana przy użyciu Intel Matrix Kernel Library (MKL), pozwalającej na szybsze przetwarzanie macierzy. Wersja ta jest dostępna bez opłat pod adresem *https://mran.microsoft.com/download/*. Istnieje również wersja płatna – Microsoft R Server – udostępniająca specjalizowane algorytmy do pracy na wielkich zbiorach danych, a także lepsze współdziałanie z Microsoft SQL Server i Hadoop. Więcej informacji na ten temat można uzyskać pod adresem *https://www.microsoft.com/en-us/server-cloud/products/r-server/*.

1.6 Podsumowanie

W tym momencie R jest już w pełni użytkowalny, choć wyposażony jedynie w prymitywny interfejs użytkownika. Znacznie lepiej będzie jednak zainstalować RStudio i posługiwać się tym interfejsem, co omówię szczegółowo w punkcie 2.2. Proces ten wymaga jedynie pobrania i uruchomienia instalatora, podobnie jak w przypadku dowolnego innego programu.

2
Środowisko R

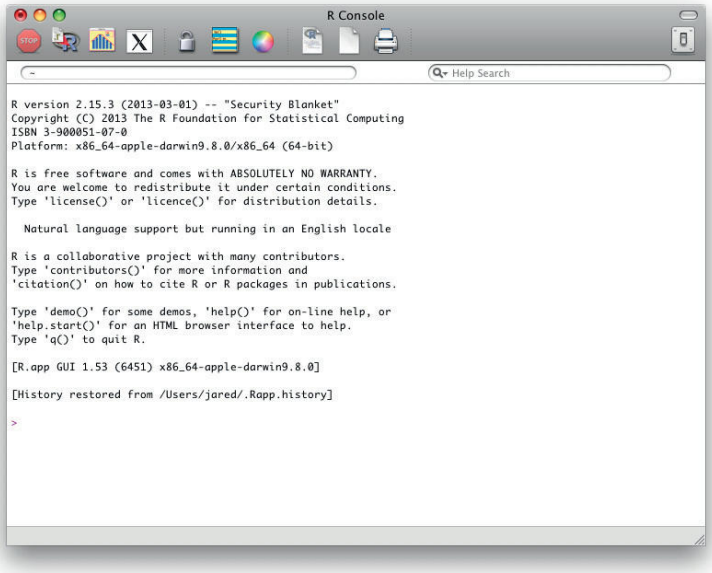
Teraz, gdy R został pobrany i zainstalowany, czas na zapoznanie się ze sposobami korzystania z niego. Podstawowy interfejs R w Windows jest zdecydowanie spartański, co można zobaczyć na rysunku 2.1. Interfejs dla Maców (rysunek 2.2) ma kilka dodatkowych funkcjonalności, zaś linuksowy znacznie mniej, będąc po prostu oknem terminala.



Rysunek 2.1 Standardowy interfejs R w systemie Windows

W odróżnieniu do wielu innych języków, R jest bardzo interaktywny. Innymi słowy, wyniki można uzyskiwać bezpośrednio dla każdego indywidualnego polecenia. Takie języki, jak C++, Pascal czy Delphi wymagają wpisania całej sekcji kodu, skompilowania go i uruchomienia, aby móc zobaczyć rezultaty. W R można w dowolnym momencie zobaczyć stan obiektów i wyniki. Ta interaktywność to jeden z najbardziej fascynujących aspektów pracy z językiem R.

Istnieje wiele zintegrowanych środowisk programistycznych (Integrated Development Environment – IDE) dostosowanych do języka R. Na użytek tej książki będę zakładał, że używane jest środowisko RStudio, które zostanie omówione w punkcie 2.2.



Rysunek 2.2 Standardowy interfejs R w Mac OS X

2.1 Interfejs wiersza polecenia

Interfejs wiersza polecenia (*command line interface* – CLI) sprawia, że R jest tak efektywny, ale również jest frustrujący w nauce. Były pewne próby zbudowania „klikalnych" interfejsów dla języka R, takich jak Rcmdr, ale żaden z nich się naprawdę nie przyjął. Okazuje się bowiem, że wpisywanie poleceń jest znacznie lepsze od posługiwania się myszą. Niektórym użytkownikom będzie trudno w to uwierzyć, szczególnie tym przyzwyczajonym do pracy w Excelu, ale z upływem czasu stanie się to łatwiejsze i mniej podatne na błędy.

Dla przykładu, dopasowanie regresji w Excelu wymaga co najmniej siedmiu kliknięć myszą, a często więcej: Data >> Data Analysis >> Regression >> OK >> Input Y Range >> Input X Range >> OK. Wszystko to musi zostać wykonane ponownie, gdy chcemy dokonać jakiejś drobnej poprawki lub gdy pojawią się nowe dane. Jeszcze trudniejsze jest wytłumaczenie koledze tych kroków przez telefon lub w mailu. Dla kontrastu, to samo działanie jest po prostu jednym wierszem w języku R, który można łatwo powtórzyć albo skopiować i wkleić. Początkowo może to się wydawać trudne, ale z czasem okaże się, że używanie wiersza poleceń sprawia, że życie jest znacznie lżejsze.

Aby wywołać polecenie w języku R, wystarczy wpisać je w konsoli obok symbolu zachęty > i nacisnąć `Enter`. Wpisy mogą być tak proste, jak pojedyncza cyfra, ale mogą być też złożonymi funkcjami, jak te, które pokażę w dalszej części tej książki.

Aby powtórzyć wiersz kodu, wystarczy wcisnąć klawisz Strzałka w górę i ponownie nacisnąć `Enter`. Wszystkie wcześniej użyte polecenia są zapisywane i można do niech powrócić, naciskając klawisze ze strzałkami w górę lub w dół, aby przemieszczać się po liście.

Przerwanie wykonywania polecenia umożliwia klawisz `Esc` w systemach Windows i Mac oraz `Ctrl-C` w systemach Linux.

Często przy pracy nad większymi analizami przydatne jest zapisywanie całego używanego kodu do pliku. Jeszcze kilka lat temu najbardziej powszechną techniką osiągnięcia tego celu było użycie edytora tekstowego*, takiego jak Sublime Text lub Notepad++, do pisania kodu, po czym kopiowanie go i wklejenie do konsoli R. Jakkolwiek to działało, było zdecydowanie mało wygodne i wymagało ciągłego przełączania się pomiędzy programami. Szczęśliwie mamy teraz środowisko RStudio, które zasadniczo zmieniło zasady gry.

2.2 RStudio

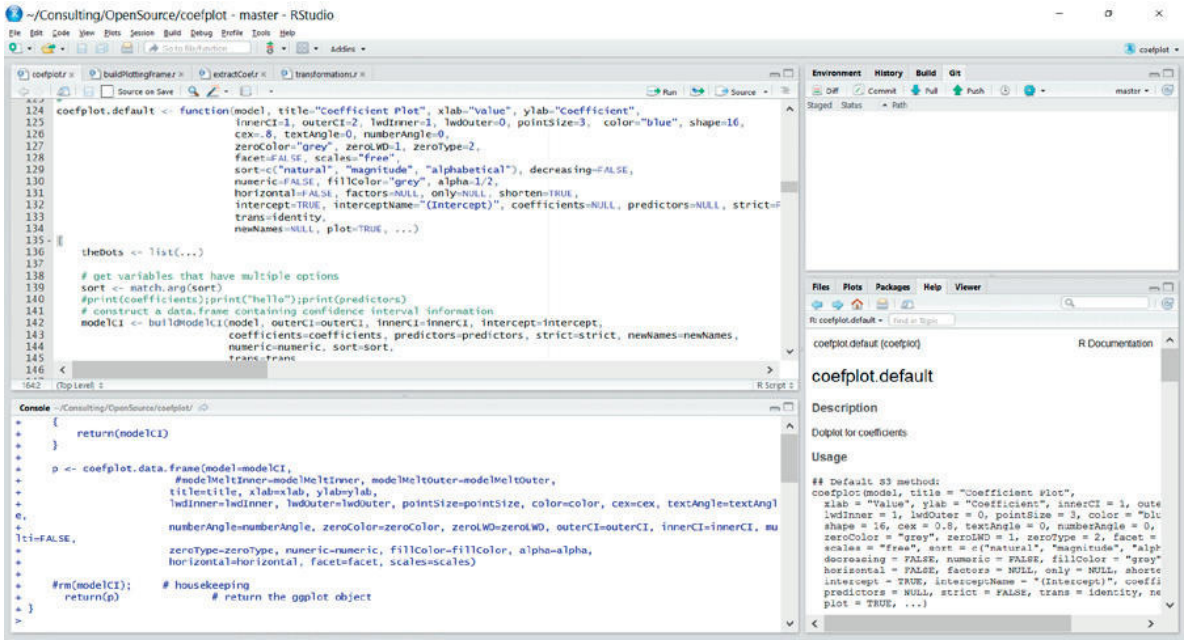
Choć obecnie dostępnych jest kilka różnych IDE dostosowanych do języka R, zdecydowanie najlepszym z nich jest RStudio, utworzone przez zespół pod kierunkiem J.J. Allaire. Wcześniejsze produkty tego autora obejmują ColdFusion oraz Windows Live Writer. RStudio jest dostępne dla systemów Windows, Mac i Linux i w każdym z nich wygląda identycznie. Jeszcze większe wrażenie robi RStudio Server, który uruchamia instancję R na linuksowym serwerze i pozwala użytkownikom wykonywać polecenia poprzez standardowy interfejs RStudio w przeglądarce sieci Web. Działa z dowolną wersją R (większą niż 2.11.1), włącznie z Microsoft R Open oraz Microsoft R Server. RStudio ma tak wiele opcji, że początkowo może wydawać się nieco przytłaczające. W tym miejscu omówię kilka najbardziej użytecznych i często używanych funkcjonalności.

RStudio ma rozległe możliwości dostosowywania, ale podstawowy interfejs wygląda mniej więcej tak, jak na rysunku 2.3. W tym przypadku lewy dolny panel to konsola R, której można używać tak samo, jak standardowej konsoli R. Lewy górny panel zawiera edytor tekstowy, ale znacznie rozbudowany. Prawy górny panel prezentuje informacje

---

* Mam tu na myśli programistyczny edytor tekstowy (*plain-text*), w odróżnieniu od procesorów tekstu, takich jak Microsoft Word. Edytor tekstowy zachowuje strukturę tekstu bez zmian, podczas gdy procesory tekstu mogą dodawać formatowanie, które utrudnia lub wręcz uniemożliwia wstawienie zawartości do konsoli.

o przestrzeni roboczej, historii poleceń, plikach w bieżącym folderze oraz kontrolę wersji Git. Prawy dolny panel pokazuje wykresy, informacje o pakietach i pliki pomocy.



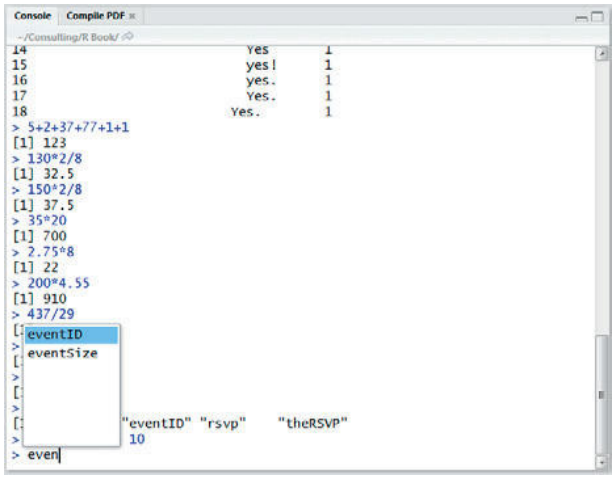
Rysunek 2.3 Ogólny układ RStudio

Istnieje kilka sposobów przesłania poleceń z edytora do konsoli i ich wykonania. Aby przesłać jeden wiersz, wystarczy umieścić w nim kursor i nacisnąć klawisze Ctrl+Enter (Command+Enter na Macu). Aby wstawić do konsoli większy (lub mniejszy) fragment, należy zaznaczyć go w panelu edytora i nacisnąć Ctrl+Enter. Aby uruchomić cały plik kodu, należy nacisnąć Ctrl+Shift+S.

Podczas wpisywania fragmentu kodu, takiego jak nazwa obiektu lub funkcji, wciśnięcie klawisza Tab spowoduje auto-dopełnienie kodu. Jeśli do wpisanych dotąd liter pasuje więcej niż jedna nazwa obiektu lub funkcji, pojawi się wyskakujące okienko z możliwymi opcjami, jak na rysunku 2.4.

Wciśnięcie Ctrl+1 przenosi kursor do obszaru edytora tekstu, zaś Ctrl+2 powoduje przejście do konsoli. Przejście do poprzedniej pozycji tabulacji w edytorze tekstowym umożliwia sekwencja klawiszy Ctrl+Alt+Left w Windows, Ctrl+PageUp w Linuksie oraz Ctrl+Option+Left na Macu. Analogicznie przejście do następnej pozycji tabulacji umożliwiają sekwencje Ctrl+Alt+Right w Windows, Ctrl+PageDown w Linuksie i Ctrl+Option+Right na Macu. W niektórych komputerach Windows te skróty mogą spowodować obrót ekranu, zatem w takiej sytuacji można użyć odpowiednio skrótów Ctrl+F11 oraz Ctrl+F12, choć działają one tylko w klienckim programie, a nie w interfejsie RStudio Server. Niemal kompletną listę skrótów można uzyskać klikając Help >> Keyboard Shortcuts albo przy użyciu skrótu klawiszowego Alt+Shift+K w Windows i Linuksie oraz Option+Shift+K na Macu. Jeszcze bardziej

wyczerpująca lista dostępna jest pod adresem *https://support.rstudio.com/hc/enus/articles/200711853-Keyboard-Shortcuts.*

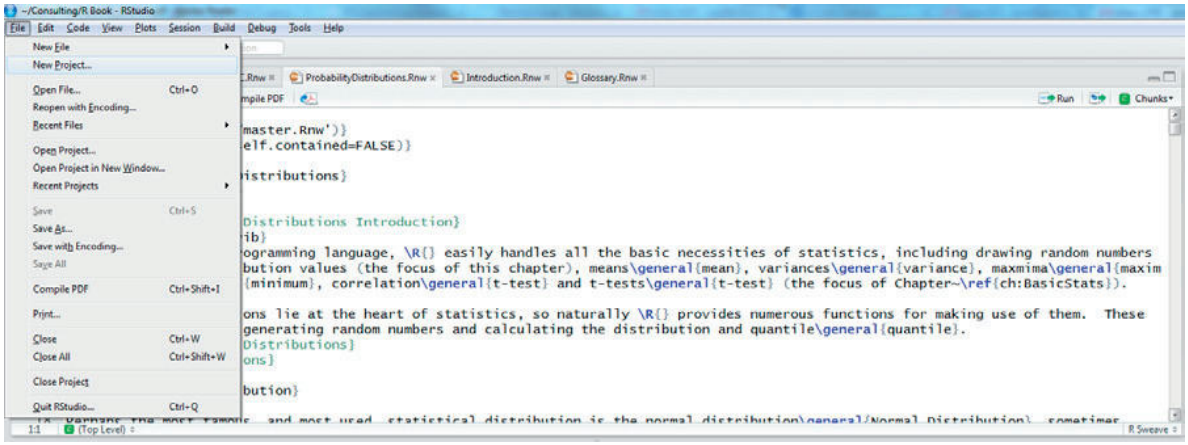


Rysunek 2.4 Dopełnianie nazw obiektów w RStudio

2.2.1 Projekty RStudio

Podstawową funkcjonalnością RStudio jest projekt. Jest to zbiór plików – a potencjalnie również danych, wyników i wykresów – powiązanych ze sobą*. Każdy pakiet ma nawet swój własny roboczy katalog. Znacznie ułatwi to zachowanie porządku.

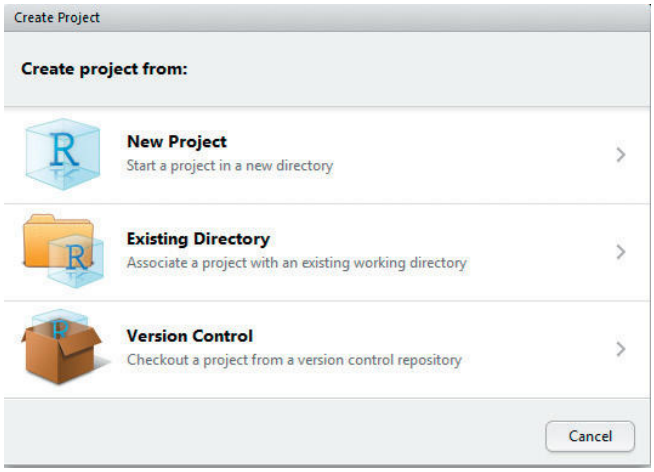
Najprostszą metodą rozpoczęcia nowego projektu jest kliknięcie File >> New Project, jak na rysunku 2.5.



Rysunek 2.5 Kliknięcie File >> New Project rozpoczyna tworzenie nowego projektu

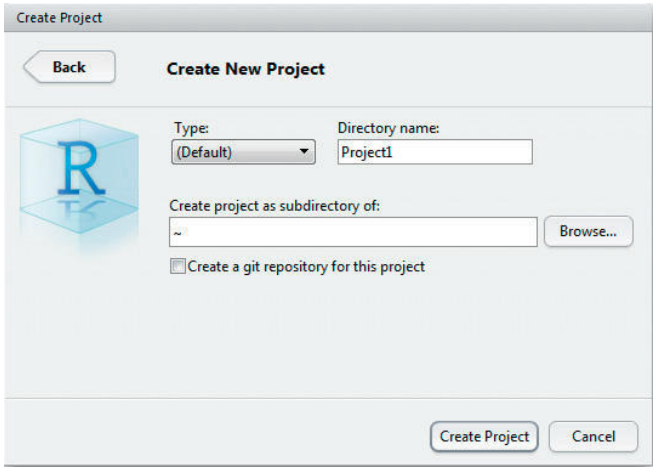
* Nie należy mylić projektu z sesją R, która obejmuje wszystkie obiekty i pracę wykonane w R i przechowywana jest w pamięci na czas działania programu, ale zazwyczaj jest resetowana po ponownym uruchomieniu R.

Dostępne są trzy opcje początkowe, widoczne na rysunku 2.6: rozpoczęcie nowego projektu w nowym katalogu, powiązanie projektu z istniejącym katalogiem lub wyewidencjonowanie projektu z repozytorium kontroli wersji, takiego jak Git lub SVN*. W każdym przypadku w katalogu umieszczany jest plik `.Rproj`, odpowiedzialny za śledzenie projektu.



Rysunek 2.6 Opcje rozpoczynania nowego projektu: nowy katalog, powiązanie projektu z istniejącym katalogiem lub pobranie projektu z repozytorium kontroli wersji

Wybranie nowego katalogu wywołuje okno dialogowe pokazane na rysunku 2.7, w którym należy określić nazwę projektu i wskazać miejsce utworzenia nowego katalogu.

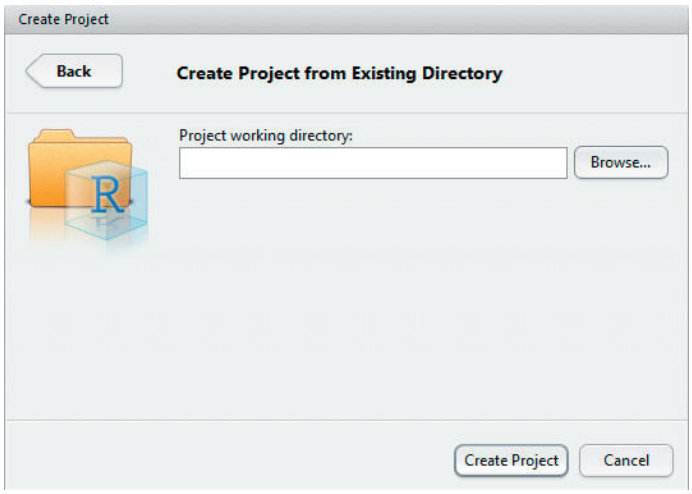


Rysunek 2.7 Okno wyboru lokalizacji dla nowego katalogu projektu

Wybranie istniejącego katalogu wymaga jego wskazania, jak na rysunku 2.8.

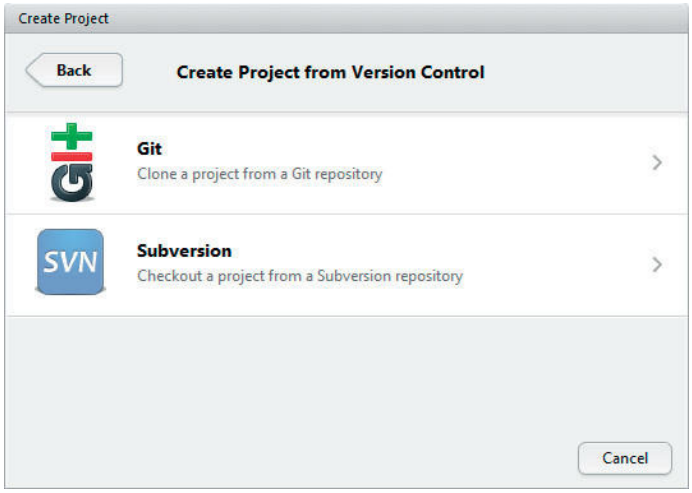
---

* Korzystanie z kontroli wersji wymaga zainstalowania odpowiedniego oprogramowania klienckiego na komputerze.



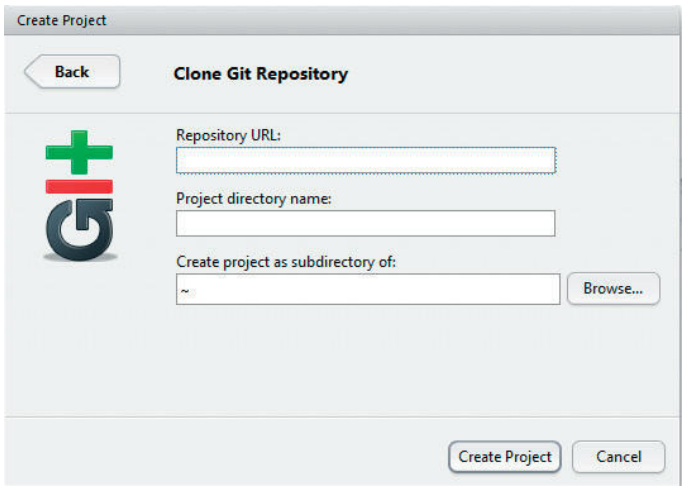
Rysunek 2.8 Okno wyboru istniejącego katalogu dla nowego projektu

Jeśli wybierzemy system kontroli wersji, pojawi się pytanie, czy użyć Git, czy SVN, jak na rysunku 2.9 (osobiście preferuję Git).



Rysunek 2.9 Opcje wyboru typu repozytorium przy rozpoczynaniu nowego projektu

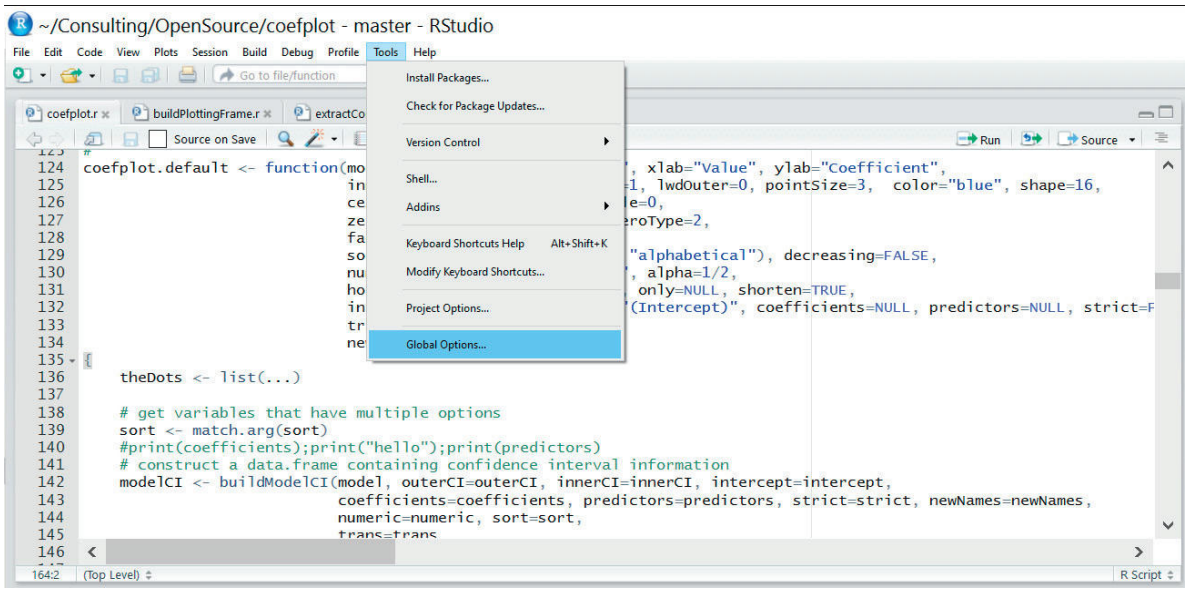
Wybranie Git powoduje konieczność wskazania adresu URL repozytorium, takiego jak `git@github.com:jaredlander/coefplot.git`, który następnie zostanie użyty do zbudowania nazwy katalogu projektu, jak na rysunku 2.10. Podobnie jak przy tworzeniu nowego katalogu, konieczne będzie wskazanie miejsca, w którym ma zostać umieszczony katalog projektu.



Rysunek 2.10 Wprowadzanie adresu URL dla repozytorium Git, a także folderu, do którego ma zostać sklonowany nowy projekt

2.2.2 Narzędzia RStudio

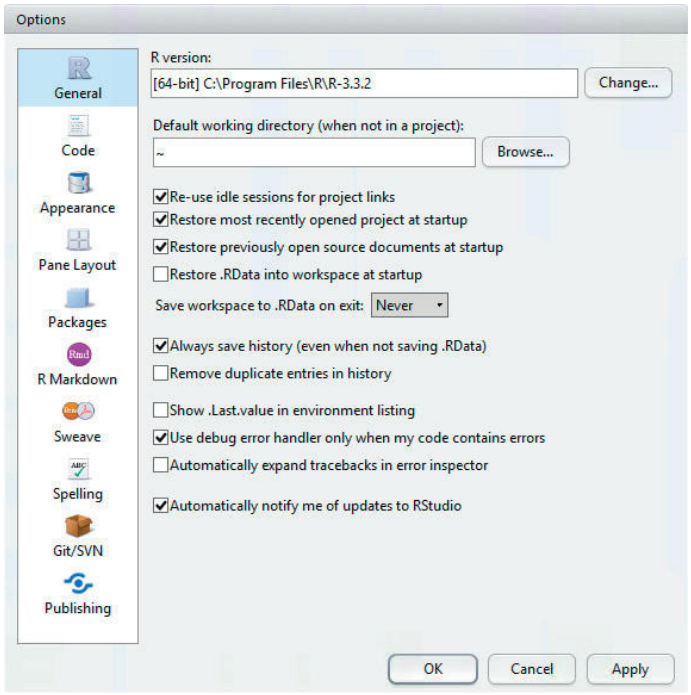
RStudio ma szerokie możliwości dostosowania. Większość opcji dostępnych jest w oknie dialogowym Options, który można wywołać klikając Tools >> Global Options, jak na rysunku 2.11.



Rysunek 2.11 Kliknięcie Tools >> Options wywołuje okno opcji RStudio

Najbardziej ogólne opcje udostępnia zakładka General, pokazana na rysunku 2.12. W systemach Windows zawiera ona kontrolkę umożliwiającą wybór wersji R, która ma być używana. Jest to bardzo przydatne narzędzie, gdy mamy zainstalowanych kilka wersji języka R. Trzeba jednak ponownie uruchomić RStudio po zmianie wersji języka.

Oczekuje się, że w przyszłości RStudio będzie oferować możliwość ustawienia różnych wersji języka R na poziomie projektu. Dobrym pomysłem jest też wyłączenie opcji wczytywania i zapisywania plików `.RData` przy uruchamianiu i zamykaniu programu*. W ten sposób za każdym razem, gdy R jest uruchamiany, tworzona jest świeża sesja bez potencjalnych uszkodzeń zmiennych lub niepotrzebnych danych zajmujących pamięć.



Rysunek 2.12 Zakładka General okna opcji RStudio

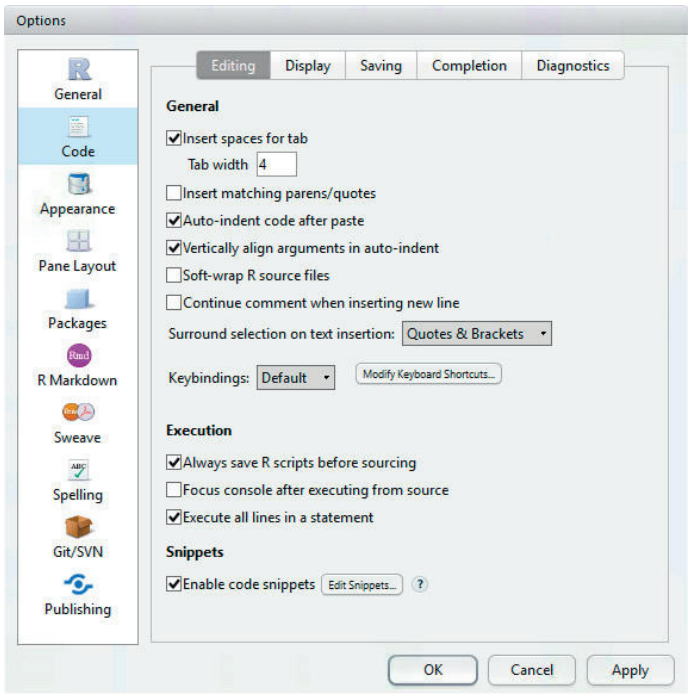
Opcje edycji kodu, dostępne w zakładce Code/Editing pokazanej na rysunku 2.13, kontrolują sposób wprowadzania kodu i jego wyświetlania w edytorze. W ogólności za dobrą praktykę uważa się zastępowanie tabulatorów spacjami (dwiema lub czterema)**, gdyż tabulatory są niekiedy różnie interpretowane w rozmaitych edytorach tekstowych. Hardkorowi programiści docenią zapewne możliwość włączenia trybu vim lub Emacs.

Opcje wyświetlania kodu (zakładka Code/Display, rysunek 2.14), kontroluje wizualne wskazówki dostępne w edytorze tekstowym i konsoli. Wyróżnianie (*highlight*) zaznaczonych słów ułatwia dostrzeżenie wielu wystąpień. Wyświetlanie numerów wierszy (*line numbers*) jest niezbędne dla wygodnego nawigowania po dłuższym kodzie. Wyświetlanie marginesu pozwala łatwo zauważyć, gdy wiersz kodu staje się zbyt długi, aby móc go łatwo przeczytać.

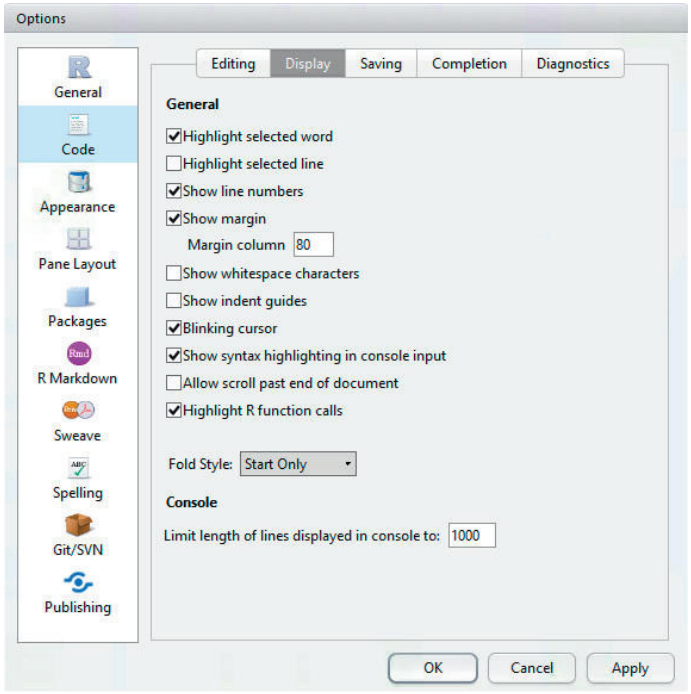
---

* Pliki RData są wygodną metodą zapisywania i udostępniania obiektów R; ich omówienie zawiera punkt 6.5.

** Cztery spacje są lepszym wyborem przy pracy z dokumentami Markdown.

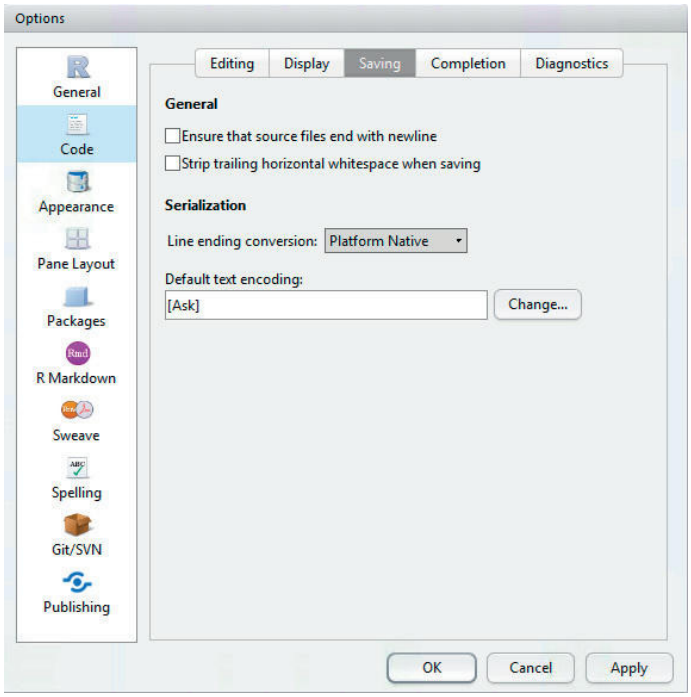


Rysunek 2.13 Opcje dostosowywania edytowania kodu



Rysunek 2.14 Opcje wyświetlania kodu

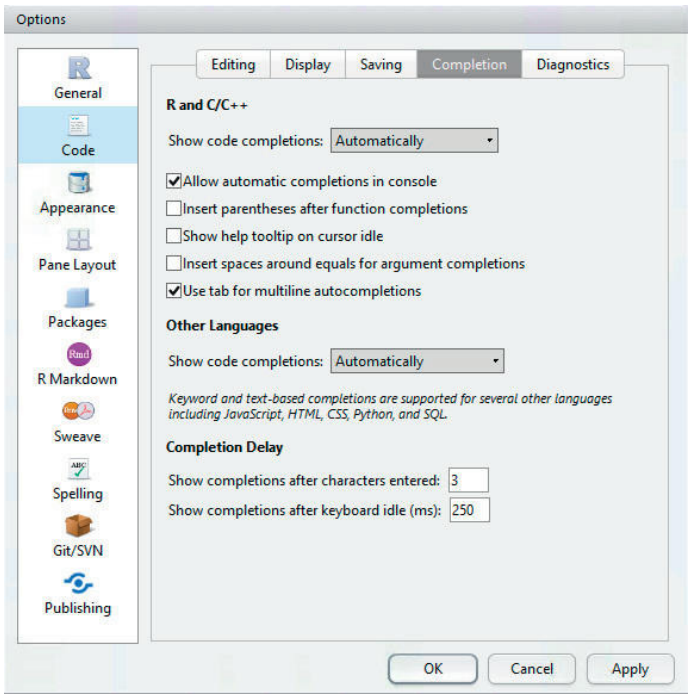
Opcje zapisywania kodu (Code/Saving), pokazane na rysunku 2.15, kontrolują sposób zapisywania plików tekstowych zawierających tworzony kod. W większości przypadków właściwym wyborem będzie użycie ustawień domyślnych, a w szczególności wybranie opcji Platform Native (natywne dla platformy) dla konwersji zakończeń wierszy w sekcji Serialization.



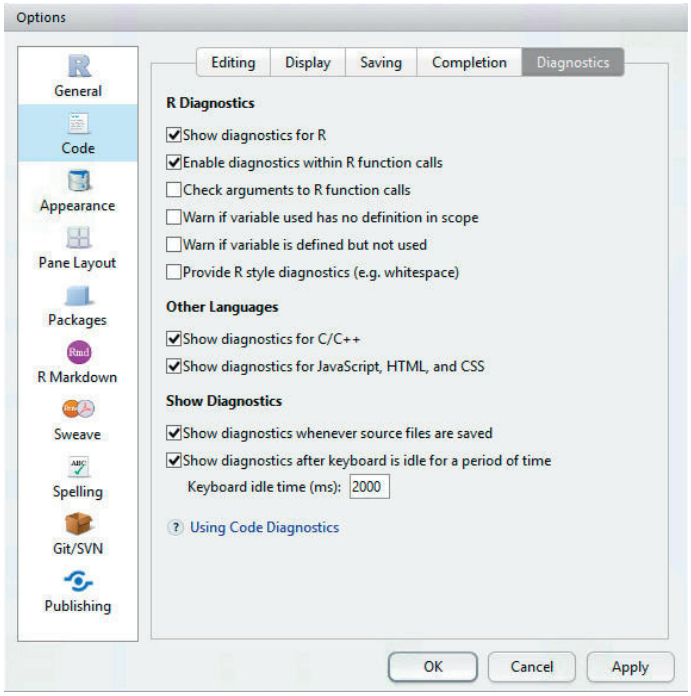
Rysunek 2.15 Opcje zapisywania kodu

Zakładka Code/Completion, pokazana na rysunku 2.16, kontroluje działanie auto-dopełniania podczas pisania programu. Niektórzy programiści lubią, gdy nawiasy są dodawane automatycznie po wpisaniu funkcji, podczas gdy inni wolą, aby tak się nie działo. Jednym ze szczególnie dzielących społeczność programistów ustawień jest to, czy należy wstawiać spacje wokół znaku równości dla nazwanych argumentów funkcji.

Opcje diagnostyki kodu (Code/Diagnostics), pokazane na rysunku 2.17, włączają sprawdzanie poprawności składniowej kodu. Mogą być bardzo pomocne przy identyfikowaniu literówek w nazwach obiektów, kiepskiego stylu i ogólnych pomyłek, takich jak brak zamykających nawiasów czy cudzysłowów.

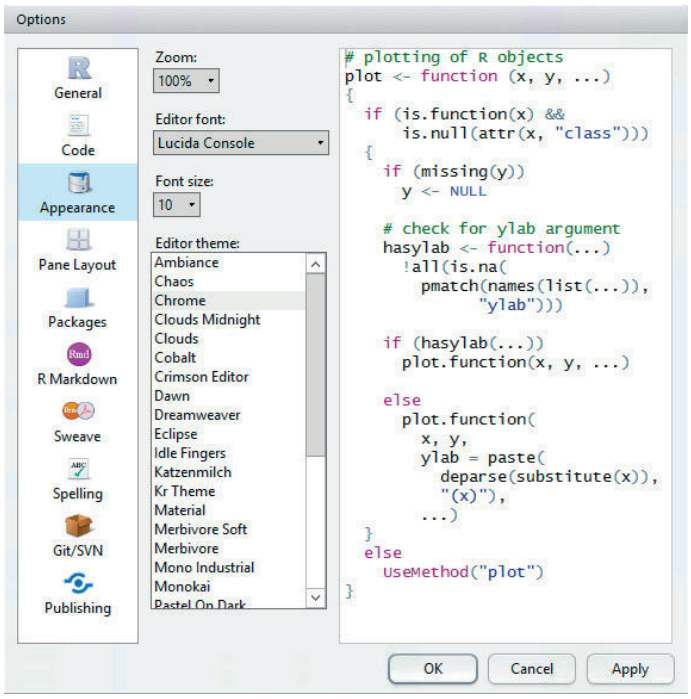


Rysunek 2.16 Opcje dostosowywania dopełniania kodu



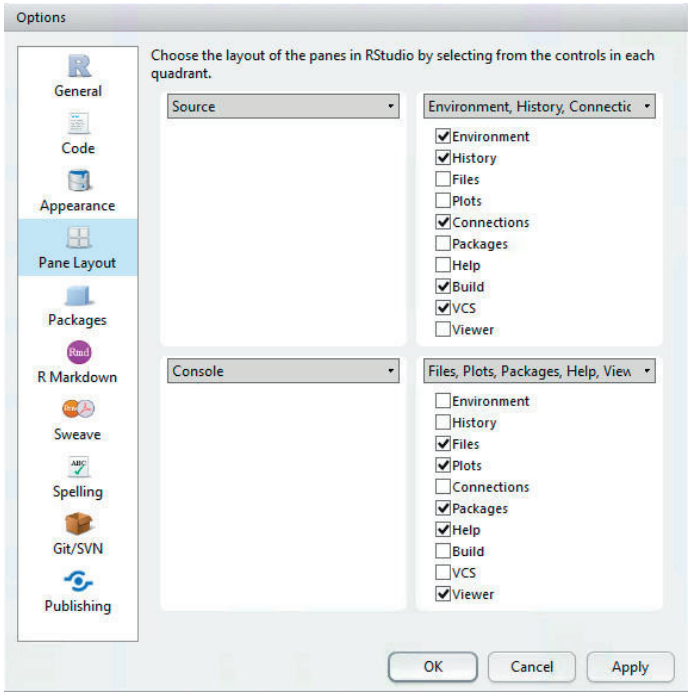
Rysunek 2.17 Opcje diagnostyki kodu

Zakładka Appearance (rysunek 2.18) pozwala określić estetykę wyglądu kodu i samego RStudio. W tym miejscu można wybrać font i jego rozmiar oraz kolory tła i tekstu, a także ogólny motyw okna programu.



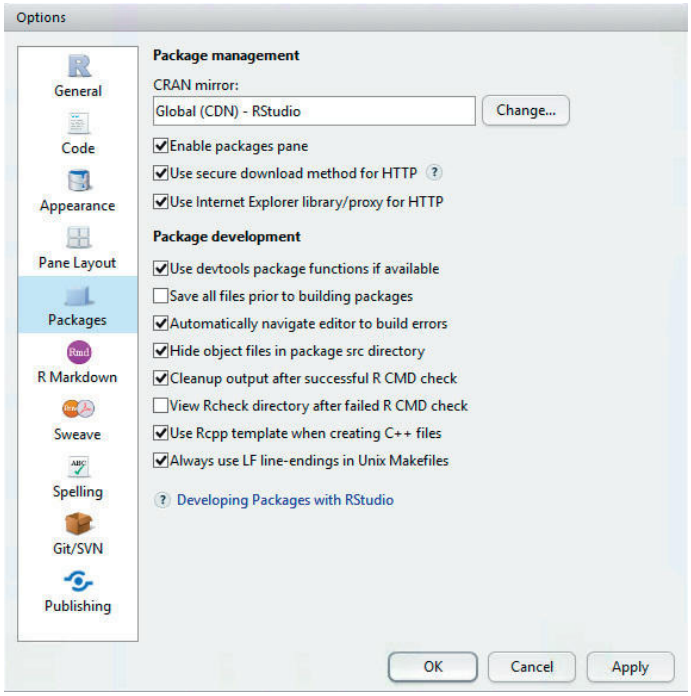
Rysunek 2.18 Opcje wyglądu kodu

Zakładka Pane Layout, pokazana na rysunku 2.19, pozwala określić rozmieszczenie paneli tworzących RStudio.



Rysunek 2.19 Te opcje kontrolują rozmieszczenie poszczególnych paneli w RStudio.

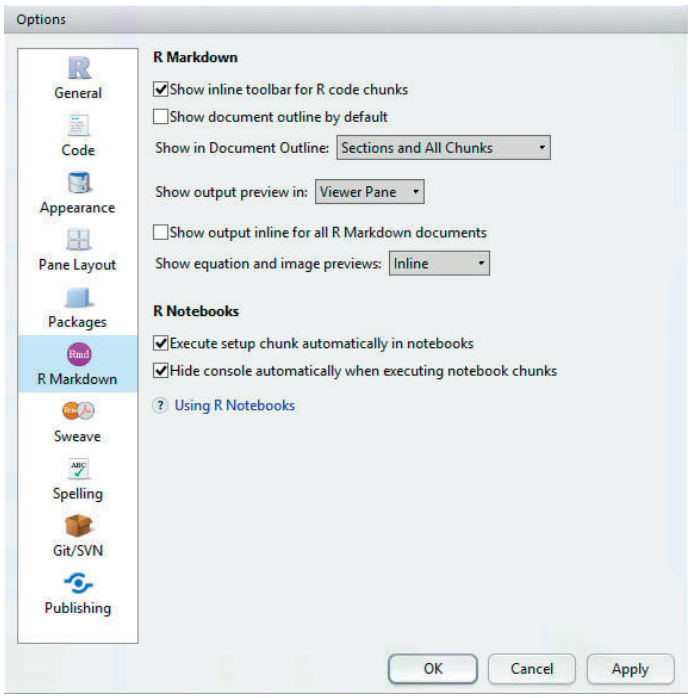
Opcje w zakładce Packages (rysunek 2.20) dotyczą pakietów, przy czym najważniejszą z nich jest używany mirror CRAN. Choć można zmienić to z poziomu konsoli, w tym miejscu określamy ustawienie domyślne. Zalecane jest wybranie takiego mirroru, który geograficznie znajduje się najbliżej.



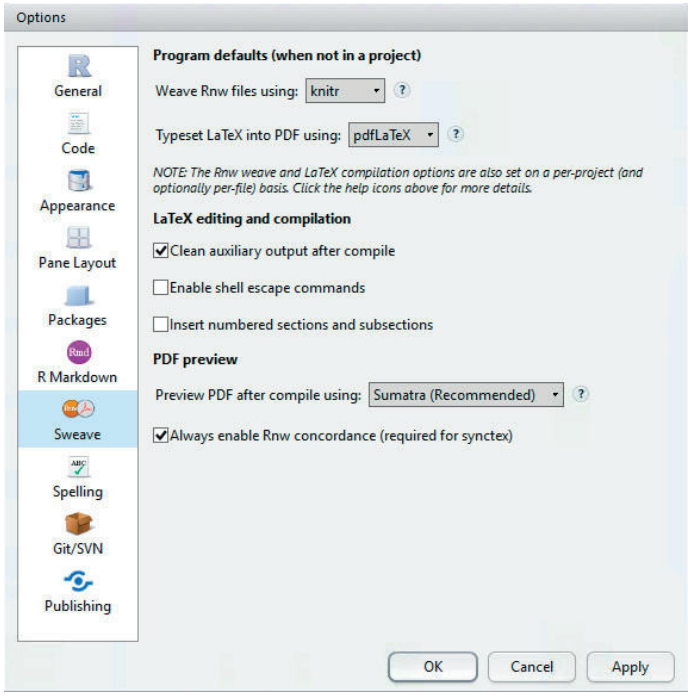
Rysunek 2.20 Opcje dotyczące pakietów. Najistotniejszy tu jest wybór odpowiedniego mirroru CRAN.

Zakładka R Markdown, pokazana na rysunku 2.21, kontroluje ustawienia dotyczące pracy z dokumentami RMarkdown. Umożliwiają one podgląd przetwarzanych dokumentów w zewnętrznym oknie lub w panelu Viewer. Dodatkowo opcje te umożliwiają traktowanie plików RMarkdown jak notatników, zawierających renderowane wyniki, obrazy i równania.

Zakładka Sweave, pokazana na rysunku 2.22, ma nieco mylącą nazwę, gdyż umożliwia ona wybór pomiędzy używaniem mechanizmu plotowania **Sweave** lub **knitr**. Obydwa służą do generowania dokumentów PDF, przy czym **knitr** umożliwia również tworzenie dokumentów HTML. Pakiet **knitr**, omówiony szczegółowo w rozdziale 27, jest zdecydowanie lepszą opcją, choć musi zostać najpierw zainstalowany, co omówimy w punkcie 3.1. Na tej zakładce można również wybrać domyślny program do przeglądania dokumentów PDF.

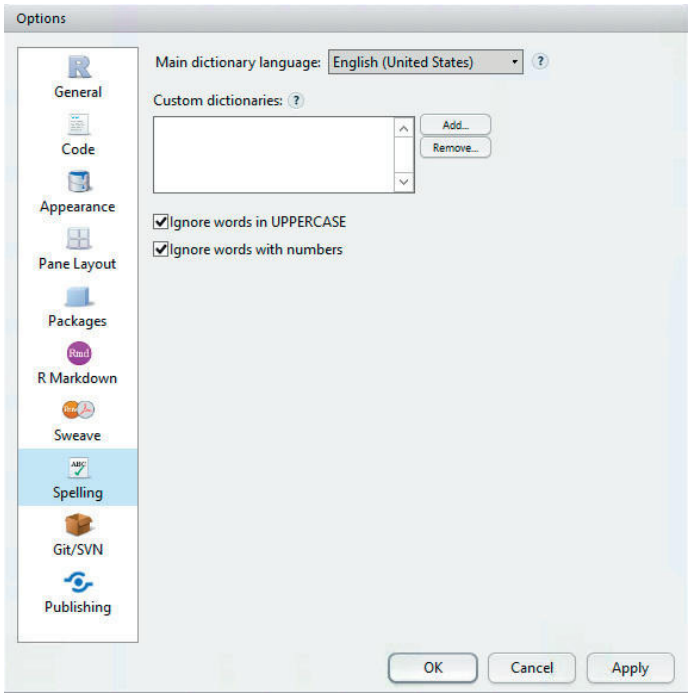


Rysunek 2.21 Opcje plików R Markdown, w tym możliwość traktowania ich jako notatników.



Rysunek 2.22 W tym miejscu wybieramy pomiędzy używaniem Sweave lub knitr oraz domyślną przeglądarkę PDF.

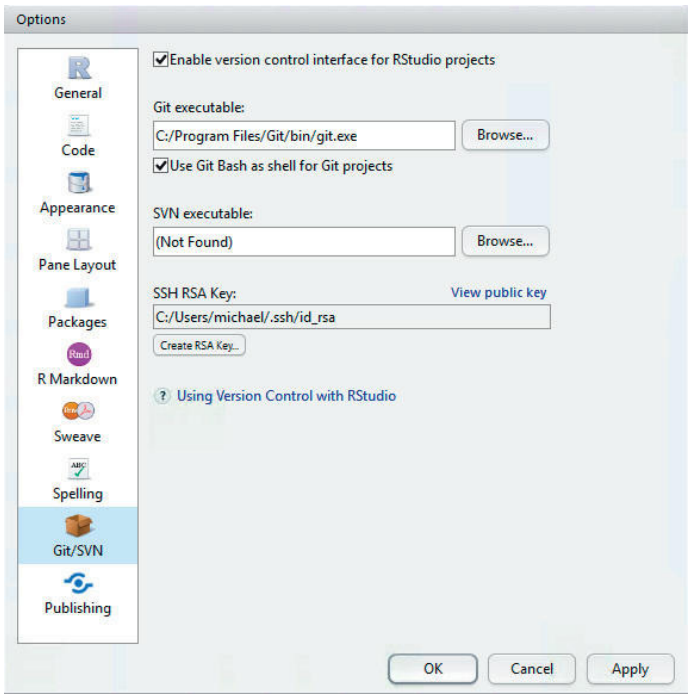
RStudio zawiera mechanizm sprawdzania pisowni dla tworzenia dokumentów LaTeX i Markdown (co preferowane, przy użyciu knitr), który można kontrolować poprzez zakładkę Spelling pokazaną na rysunku 2.23. Zasadniczo nie trzeba tu nic ustawiać.



Rysunek 2.23 Te opcje pozwalają wybrać słownik sprawdzania pisowni dla różnych języków, a także słowniki niestandardowe.

Opcje Git/SVN, pokazane na rysunku 2.24, pokazują, gdzie (i czy) obecne są pliki wykonywalne dla repozytoriów Git lub SVN. Trzeba je ustawić jedynie raz, ale są niezbędne do korzystania z kontroli wersji.

Ostatnia zakładka opcji, Publishing (rysunek 2.25), definiuje połączenia dla publikowania dokumentów w ShinyApps.io lub RStudio Connect.



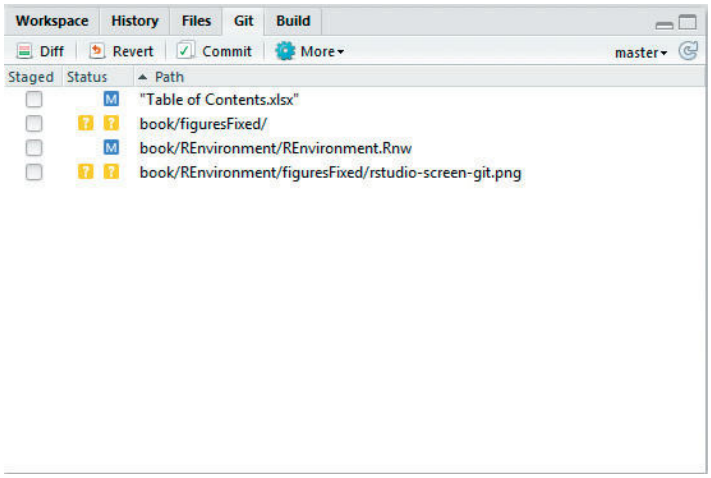
Rysunek 2.24 W tym miejscu można określić lokalizację plików wykonywalnych Git i SVN, aby mogły być używane przez RStudio.

Rysunek 2.25 Ustawianie połączenia do ShinyApps.io lub RStudio Connect

2.2.3 Integracja z Git

Korzystanie z kontroli wersji jest świetnym pomysłem z wielu powodów. Po pierwsze i przede wszystkim udostępnia ona migawki kodu z różnych punktów w czasie i umożliwia łatwy powrót do wcześniejszego stanu kodu. Dodatkowe korzyści obejmują posiadanie kopii zapasowej kodu oraz możliwość łatwego przenoszenia go pomiędzy różnymi komputerami z minimalnym wysiłkiem.

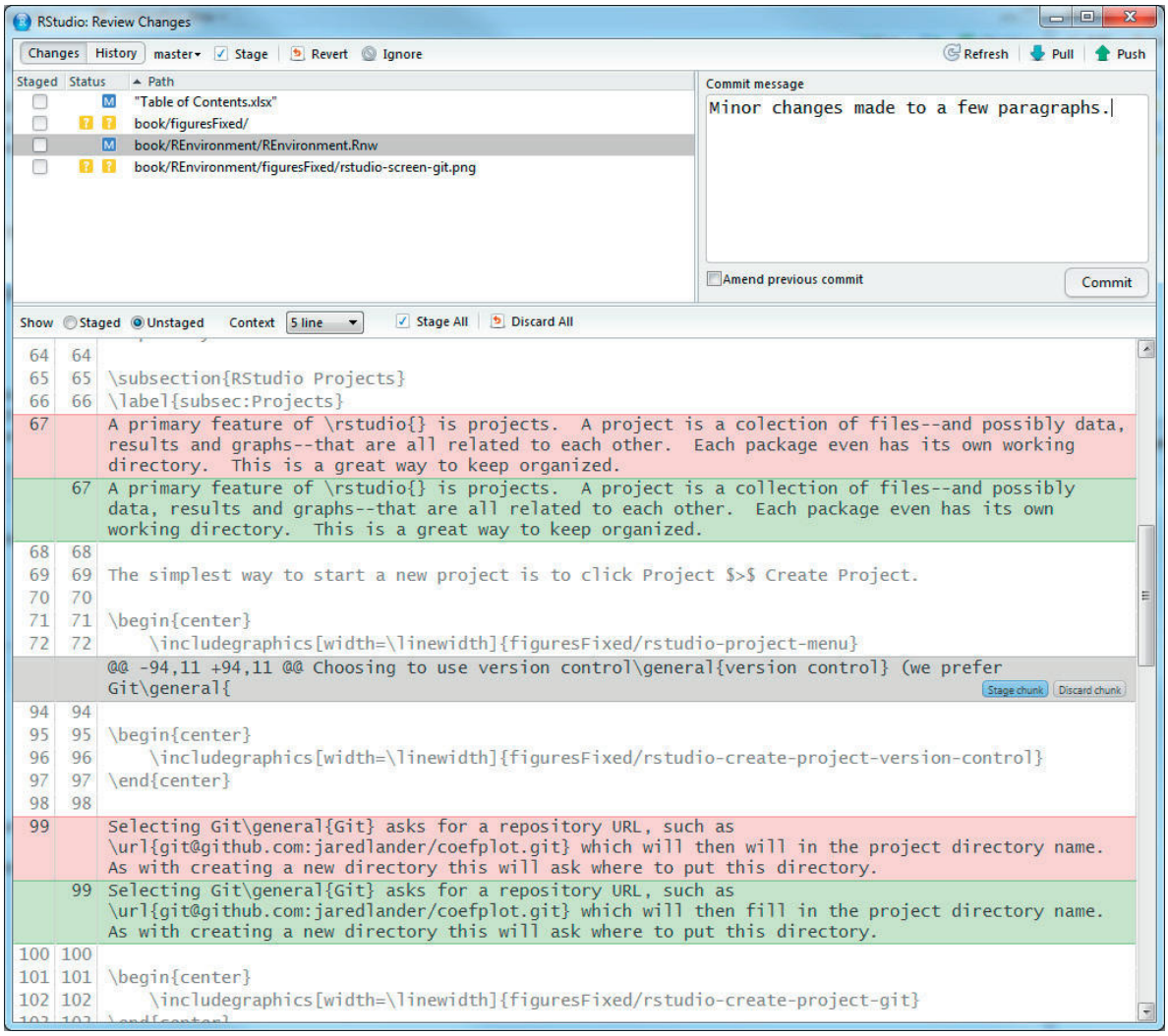
Choć SVN był uznawany za wzorcowy standard wśród mechanizmów kontroli wersji, obecnie jest coraz szerzej zastępowany przez Git, zatem skupię się na tym ostatnim. Po powiązaniu projektu z repozytorium Git* RStudio udostępnia panel Git, podobny do pokazanego na rysunku 2.26.



Rysunek 2.26 Panel Git pokazujący status Git plików podlegających kontroli wersji. Niebieski kwadrat z białą literą M sygnalizuje, że plik został zmieniony i powinien zostać zatwierdzony. Żółty kwadrat z białym znakiem zapytania wskazuje nowy plik, który jeszcze nie jest śledzony przez Git.

Główna funkcjonalność systemu to zatwierdzanie zmian i wypychanie ich na serwer oraz pobieranie z serwera zmian dokonywanych przez innych użytkowników. Kliknięcie przycisku Commit (zatwierdź) wywołuje okno dialogowe pokazane na rysunku 2.27, wyświetlające pliki zmodyfikowane oraz nowe. Kliknięcie jednego z nich pozwala zobaczyć dokonane zmiany; fragmenty usunięte są oznaczone kolorem różowym, a dopisane zielonym. Jest tu również miejsce na wpisanie komunikatu opisującego zatwierdzenie. Kliknięcie Commit utrwala zmiany, zaś kliknięcie przycisku Push (wypchnij) spowoduje wysłanie ich na serwer.

* Konto Git należy wcześniej skonfigurować albo przy użyciu GitHub (*https://github.com/*), albo Bitbucket (*https://bitbucket.org/*).



Rysunek 2.27 Pliki i dokonane w nich zmiany, przy czym kolor zielony oznacza nowe (dopisane) fragmenty, a różowy – części usunięte. W prawym górnym rogu dostępne jest miejsce na wpisanie komentarza wyjaśniającego zatwierdzenie.

2.3 Microsoft Visual Studio

Microsoft Visual Studio udostępnia narzędzia IDE do pracy z R. Choć większość użytkowników R będzie się czuła wygodniej używając RStudio, jest to przyjemna opcja dla osób przyzwyczajonych do posługiwania się Visual Studio.

2.4 Podsumowanie

Użyteczność języka R znacznie wzrosła w ciągu ostatnich kilku lat, w znaczącej części dzięki pojawieniu się RStudio. Korzystanie z IDE może znacząco poprawić wydajność i zmienić pracę w języku R z ledwo tolerowalnej na rzeczywiście przyjemną*. Dopełnianie kodu, edytor tekstowy, integracja z Git i struktura projektów udostępniane przez RStudio są nieocenione z punktu widzenia sprawnego przepływu pracy programisty.

* Jeden z moich studentów wspominał, że wolał korzystać z Matlab, niż z R, dopóki nie zaczął używać RStudio.

3
Pakiety R

Zapewne najważniejszą przyczyną fenomenalnego wzrostu popularności języka R jest przebogata kolekcja pakietów tworzonych i doskonalonych przez użytkowników. Według stanu z początku lutego 2017 roku w CRAN* dostępnych było ponad 10 000 pakietów napisanych przez przeszło 2000 różnych osób. Istnieją spore szanse, że jeśli jakaś technika statystyczna w ogóle istnieje, została już napisana w R i udostępniona w CRAN. Wrażenie robi nie tylko ogromna liczba pakietów; wiele z nich zostało napisanych przez prawdziwe autorytety na tym polu, twórców takich jak Andrew Gelman, Trevor Hastie, Dirk Eddelbuettel czy Hadley Wickham.

Pakiet jest, zasadniczo rzecz biorąc, biblioteką wstępnie przygotowanego kodu przeznaczonego do realizacji jakiegoś zadania lub zbioru zadań. Na przykład pakiet **survival** służy do analizy przeżycia (*survival analysis*), **ggplot2** umożliwia tworzenie wykresów, zaś pakiet **sp** jest przydatny przy radzeniu sobie z danymi przestrzennymi (*spatial*).

Ważne jest, aby pamiętać, że nie wszystkie pakiety prezentują tę samą jakość. Niektóre zostały zbudowane tak, aby były niezawodne i są bardzo dobrze utrzymywane, ale są też i takie, które wprawdzie sporządzono w dobrej intencji, ale często zawodzą z powodu nieprzewidzianych błędów, zaś jeszcze inne są po prostu kiepskie. Nawet w przypadku najlepszych pakietów trzeba mieć na uwadze, że większość z nich została napisana przez statystyków dla statystyków, zatem mogą się znacząco różnić od tego, czego mógłby oczekiwać inżynier oprogramowania.

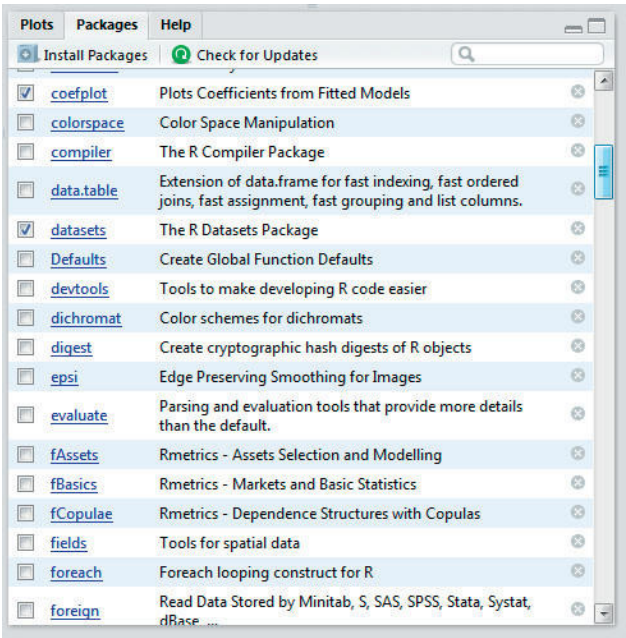
W tej książce nie podejmuję się przedstawić wyczerpującej listy dobrych pakietów, których warto używać, gdyż ulega ona nieustannym zmianom. Tym niemniej jest kilka pakietów, które są tak rozpowszechnione, że używam ich w tej książce, jakby były częścią samego języka R. Należą do nich **ggplot2**, **tidyr** oraz **dplyr** autorstwa Hadleya Wickhama; **glmnet** napisany przez Trevora Hastie, Roberta Tibshirani i Jerome Friedmana; **Rcpp**, którego autorem jest Dirk Eddelbuettel oraz **knitr**, dzieło Yihui Xie. Nie mogę też pominąć własnych pakietów **coefplot**, **useful** i **resumer** – i to na pewno nie jest koniec.

---

* *http://cran.r-project.org/web/packages/*

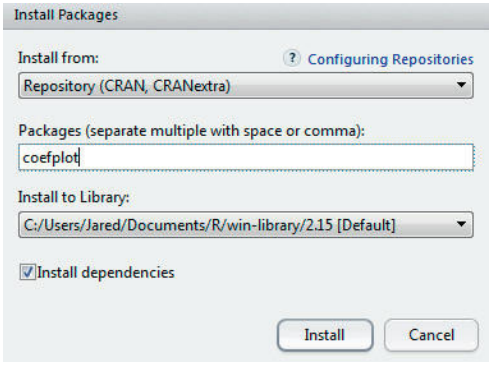
3.1 Instalowanie pakietów

Podobnie jak w przypadku wielu innych zadań w R, istnieje kilka sposobów instalowania pakietów. Najprostsza jest instalacja przy użyciu GUI udostępnianego przez RStudio (rysunek 3.1). Dostęp do panelu Packages widocznego na tym rysunku można uzyskać klikając jego zakładkę lub naciskając skrót klawiszowy Ctrl+7.



Rysunek 3.1 Panel Packages w RStudio

W lewym górnym rogu klikamy przycisk Install Packages, aby wyświetlić okno dialogowe pokazane na rysunku 3.2.



Rysunek 3.2 Okno dialogowe instalacji pakietów w RStudio

W oknie tym wpisujemy nazwę pakietu (przydaje się tu funkcja auto-dopełniania RStudio) i klikamy Install. Można wyspecyfikować wiele pakietów, rozdzielając je przecinkami. Spowoduje to pobranie i zainstalowanie pożądanych pakietów, które następnie są dostępne do użycia. Zaznaczenie pola wyboru Install dependencies (instaluj

wymagane) spowoduje automatyczne pobranie i zainstalowanie wszystkich pakietów, których wybrany pakiet potrzebuje do działania. Dla przykładu mój pakiet **coefplot** jest zależny od pakietów **ggplot2**, **plyr**, **dplyr**, **useful**, **stringr** i **reshape2**, przy czym każdy z nich może mieć swoje własne wymagane pakiety.

Alternatywnie można wpisać bardzo proste polecenie w konsoli:

```
> install.packages("coefplot")
```

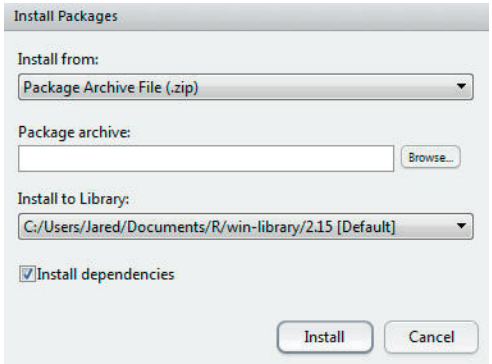
Spowoduje to identyczne działanie, jak opisane wyżej operacje przy użyciu GUI.

Od pewnego czasu można zauważyć rosnącą popularność instalowania pakietów bezpośrednio z repozytoriów GitHub lub BitBucket, szczególnie w przypadku, gdy chcemy uzyskać rozwojową wersję pakietu. Można to osiągnąć przy użyciu pakietu **devtools**, omówionego w punkcie 3.2.

```
> library(devtools)
> install_github(repo="coefplot/jaredlander")
```

Jeśli pakiet instalowany z repozytorium zawiera kod źródłowy języka kompilowanego – zazwyczaj C++ lub FORTRAN – muszą być zainstalowane odpowiednie kompilatory. Więcej informacji na ten temat zawiera punkt 30.7.

Niekiedy zachodzi potrzeba zainstalowania pakietu z pliku lokalnego w postaci archiwum zip zawierającego wstępnie skompilowany pakiet lub archiwum tar.gz w przypadku kodu pakietu. Można to zrealizować przy użyciu tego samego okna dialogowego, co omówione wcześniej, ale trzeba przełączyć opcję Install from: na Package Archive File (rysunek 3.3). Następnie należy wskazać plik do zainstalowania. Trzeba jednak zauważyć, że ta procedura nie instaluje wymaganych pakietów i jeśli nie są one obecne, instalacja się nie powiedzie. Trzeba się upewnić, że wymagane pakiety zostały zainstalowane wcześniej.



Rysunek 3.3 Okno dialogowe instalacji pakietów w RStudio umożliwia instalację z pliku archiwum.

Analogicznie, zadanie to można wykonać przy użyciu `install.packages`.

```
> install.packages("coefplot_1.1.7.zip")
```

3.1.1 Odinstalowywanie pakietów

W rzadkiej i nietypowej sytuacji, gdy pakiet musi zostać odinstalowany (usunięty), najłatwiejszą metodą jest kliknięcie szarego kółka z białym X na prawo od opisu pakietu w panelu Packages RStudio (patrz rysunek 3.1). Alternatywnie można to zrealizować przy użyciu funkcji **remove.packages**, przy czym pierwszym argumentem jest `character vector` zawierający nazwy pakietów do usunięcia.

3.2 Ładowanie pakietów

Teraz, gdy pakiety zostały już zainstalowane, są niemal gotowe do użycia i jedynie muszą zostać załadowane. Dostępne są dwa polecenia, które realizują to zadanie: **library** oraz **require**. Obydwa wykonują to samo: ładują pakiet. Użycie **require** zwróci `TRUE`, jeśli zakończy się powodzeniem, oraz `FALSE` wraz z ostrzeżeniem, jeśli polecenie nie będzie mogło odnaleźć pakietu. Ta zwracana wartość jest użyteczna, jeśli ładujemy pakiet z wnętrza funkcji. Praktyka taka jest uważana za akceptowalną przez część programistów, ale za niewłaściwą przez innych. Wywołanie **library** wobec pakietu, który nie jest zainstalowany, powoduje błąd, co może być przydatne, gdy kod uruchamiany jest w skrypcie. W użyciu interaktywnym nie ma większej różnicy, ale przy pisaniu skryptów preferowane jest stosowanie **library**. Obowiązkowym argumentem każdej z tych funkcji jest nazwa pożądanego pakietu, ujęta w cudzysłowy lub bez nich.

Tak więc załadowanie pakietu **coefplot** może wyglądać następująco:

```
> library(coefplot)
Loading required package: ggplot2
```

Funkcja wypisuje również informacje o ładowanych pakietach wymaganych (o ile istnieją). Tę informację zwrotną można pominąć, ustawiając argument `quietly` na wartość `TRUE`.

```
> library(coefplot, quietly=TRUE)
```

Pakiet musi być załadowany tylko po rozpoczęciu nowej sesji R. Po załadowaniu pozostaje dostępny do ponownego uruchomienia R lub do usunięcia pakietu z pamięci, zgodnie z opisem w punkcie 3.2.1.

Alternatywą do ładowania pakietu poprzez kod jest zaznaczenie pola wyboru obok nazwy pakietu w panelu Packages RStudio, widocznego na rysunku 3.1. Spowoduje to załadowanie pakietu – w istocie RStudio wywołuje kod pokazany powyżej.

3.2.1 Usuwanie pakietów z pamięci

Niekiedy pakiet musi zostać usunięty z pamięci (na przykład z powodu kolizji z nowszą wersją lub konieczności jego zmodyfikowania). Można to osiągnąć, czyszcząc pole wyboru w panelu Packages przy nazwie pakietu albo przy użyciu funkcji **detach**. Funkcja przyjmuje jako argument nazwę pakietu poprzedzoną słowem kluczowym `package:` (całość w cudzysłowach).

```
> detach("package:coefplot")
```

Nie jest niczym nadzwyczajnym, że funkcje zawarte w różnych pakietach noszą tę samą nazwę. Dla przykładu funkcja **coefplot** obecna jest zarówno w pakiecie **arm** (autorstwa Andrew Gelmana), jak i **coefplot**[*]. Jeśli obydwa pakiety są załadowane, przy wywołaniu tej funkcji zostanie użyta wersja pochodząca z ostatnio załadowanego pakietu. Istnieje jednak sposób obejścia tego zachowania, a mianowicie poprzedzenie nazwy funkcji nazwą pakietu, oddzielonej dwoma dwukropkami (`::`).

```
> arm::coefplot(object)
> coefplot::coefplot(object)
```

Nie tylko zapewni to wywołanie właściwej funkcji; dodatkowo pozwoli to na wywołanie funkcji bez wcześniejszego załadowania pakietu (zostanie on załadowany przy pierwszym użyciu funkcji).

3.3 Budowanie pakietu

Tworzenie pakietów to jeden z najbardziej satysfakcjonujących elementów pracy z językiem R, a zwłaszcza udostępnienie tego pakietu w społeczności za pośrednictwem CRAN. Cały proces omówiony jest szczegółowo w rozdziale 30.

---

* Ten konkretny przypadek wynika z faktu, że utworzyłem **coefplot** jako usprawnienie funkcji dostępnej w **arm**. Istnieją jednak sytuacje, gdy funkcje o identycznych nazwach nie mają ze sobą nic wspólnego.

3.4 Podsumowanie

Pakiety tworzą szkielet społeczności R. Często są uważane za coś, dzięki czemu praca z R jest tak pożądana. W ten sposób społeczność sprawia, że jej praca i tak wiele technik statystycznych jest dostępna dla całego świata. Przy tak wielkiej liczbie pakietów znalezienie tego jednego może być przytłaczające. Strona CRAN Task Views (*http://cran.r-project.org/web/views/*) udostępnia wyczyszczoną listę pakietów do różnych zastosowań. Niekiedy jednak najlepszą metodą znalezienia potrzebnego pakietu jest po prostu spytanie społeczności. Dodatek A zawiera kilka zasobów pozwalających zrealizować to zadanie.

4
Podstawy R

Język R jest potężnym narzędziem wykonywania wszelkiego rodzaju kalkulacji, manipulowania danymi i obliczeń naukowych. Zanim jednak przejdziemy do złożonych operacji dostępnych w R, musimy zacząć od podstaw. Podobnie jak większość innych języków, także R ma specyficzne dla siebie możliwości wykonywania działań matematycznych, zmienne, funkcje i typy danych.

4.1 Podstawowe działania arytmetyczne

Będąc językiem programowania ukierunkowanym na statystykę, R oczywiście może zostać wykorzystany do wykonywania podstawowych działań matematycznych i właśnie od tego miejsca rozpoczniemy poznawanie jego możliwości.

Zacznijmy od „hello, world!" podstawowej arytmetyki: 1 + 1. W konsoli możemy zauważyć znak zachęty – prawy kątowy nawias (inaczej mówiąc, znak większości), czyli **>** – to w tym miejscu należy wpisać kod. Na początku sprawdźmy, że R w ogóle działa, wpisując

```
> 1 + 1
[1] 2
```

Jeśli w wyniku otrzymaliśmy 2, wszystko jest świetnie; jeśli nie, coś poszło bardzo, ale to bardzo źle. Zakładając, że zadziałało poprawnie, spróbujmy kilku nieco bardziej złożonych wyrażeń:

```
> 1 + 2 + 3
[1] 6

> 3 * 7 * 2
[1] 42

> 4 / 2
[1] 2
```