Tworzenie pakietów w R

Tomasz Szkutnik

University of Economics in Katowice

Tworzenie pakietów w R

Czym jest pakiet R:

- podstawowym komponentem pozwalającym na współdzielenie kodu.
- zbiorem takich elementów jak: kod, dane, dokumentacja, testy, dema.

Dlaczego tworzymy pakiety w R:

- Dla siebie. Chcemy archiwizować pracę.
- Dla kogoś (wewnętrznie lub zewnętrznie):
  - Współpracownicy, Studenci, jako dodatek do artykułu. Ograniczony dostęp.
  - W formie repozytorium CRAN (Comprehensive R Archive Network). Publiczny dostęp.

Tworzenie pakietów w R

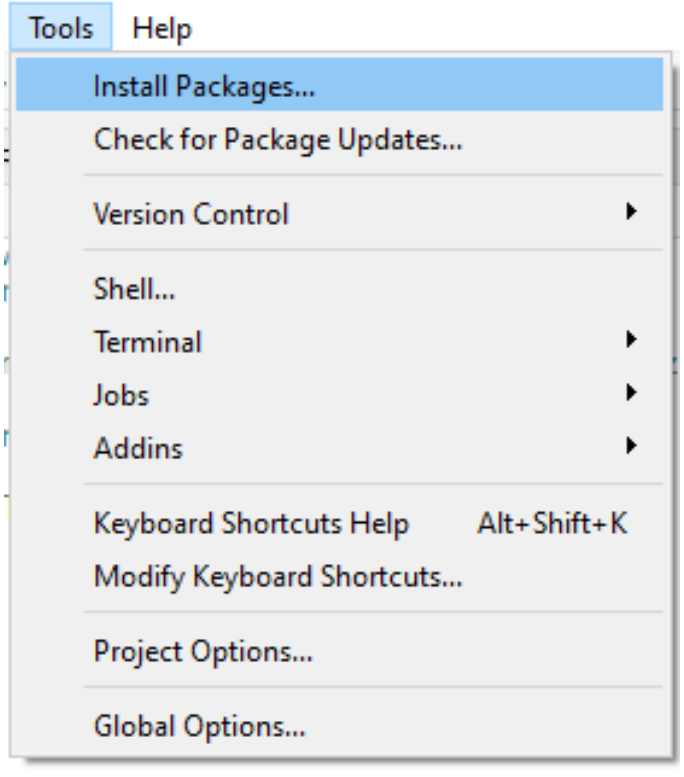
Komponenty pakietu R:

- **Kod R**- folder [R]
- **metadane**- plik DESCRIPTION- pozwala na zawarcie najważniejszych informacji o pakiecie.
- **Dokumentacja**- zawiera opis funkcji, pozwala zrozumieć co robiliśmy rok temu itd...
- **Vignettes**- zawiera szczegóły każdej funkcji w pakiecie. Pełna forma dokumentacji pakietu.
- **Tests**- pozwalają na weryfikację czy pakiet działa poprawnie- szczególnie po poprawkach. Tworzone są tu testy jednostkowe które definiują poprawne zachowanie funkcji, informują o nieprawidłowościach.
- **Przestrzeń nazw**- plik NAMESPACE- pozwala na określenie jak nasze funkcję będą działały z innymi pakietami, lub jakich zewnętrznych funkcji wymagamy w naszym pakiecie.
- **Folder [data]**- pozwala na wbudowanie danych w nasz pakiet.
- **Skompilowany kod**- folder [src]- funkcje w C lub C++.

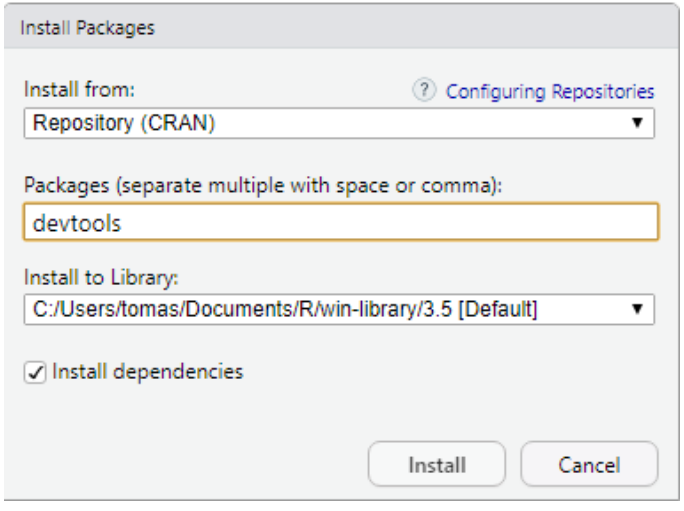
Tworzenie pakietów w R

Zamin zrobimy cokolwiek instalujemy dwa pakiety:

1. Pakiet `devtools`
2. Pakiet `usethis`



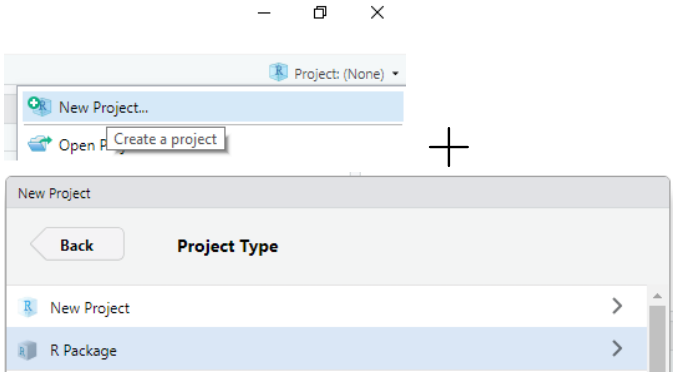
+



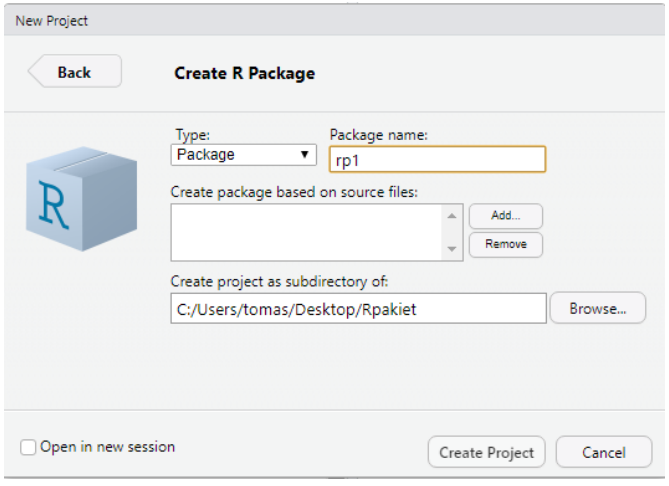
Tworzenie pakietów w R

Tworzenie pakietu w środowisku R-Studio

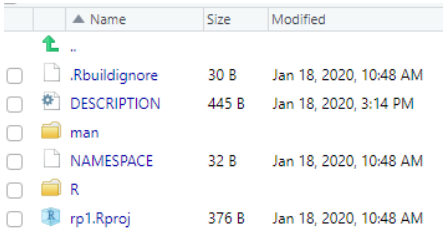
1 Wybieramy nowy projekt:



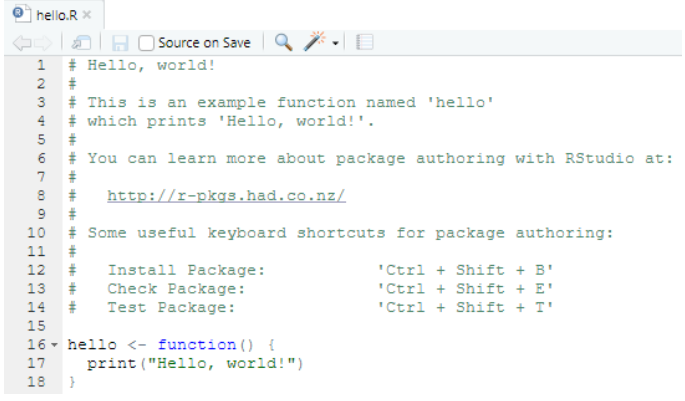
2 Podajemy podstawowe informacje:



3 Domyślna struktura projektu



4 Domyślna funkcja w folderze [R]

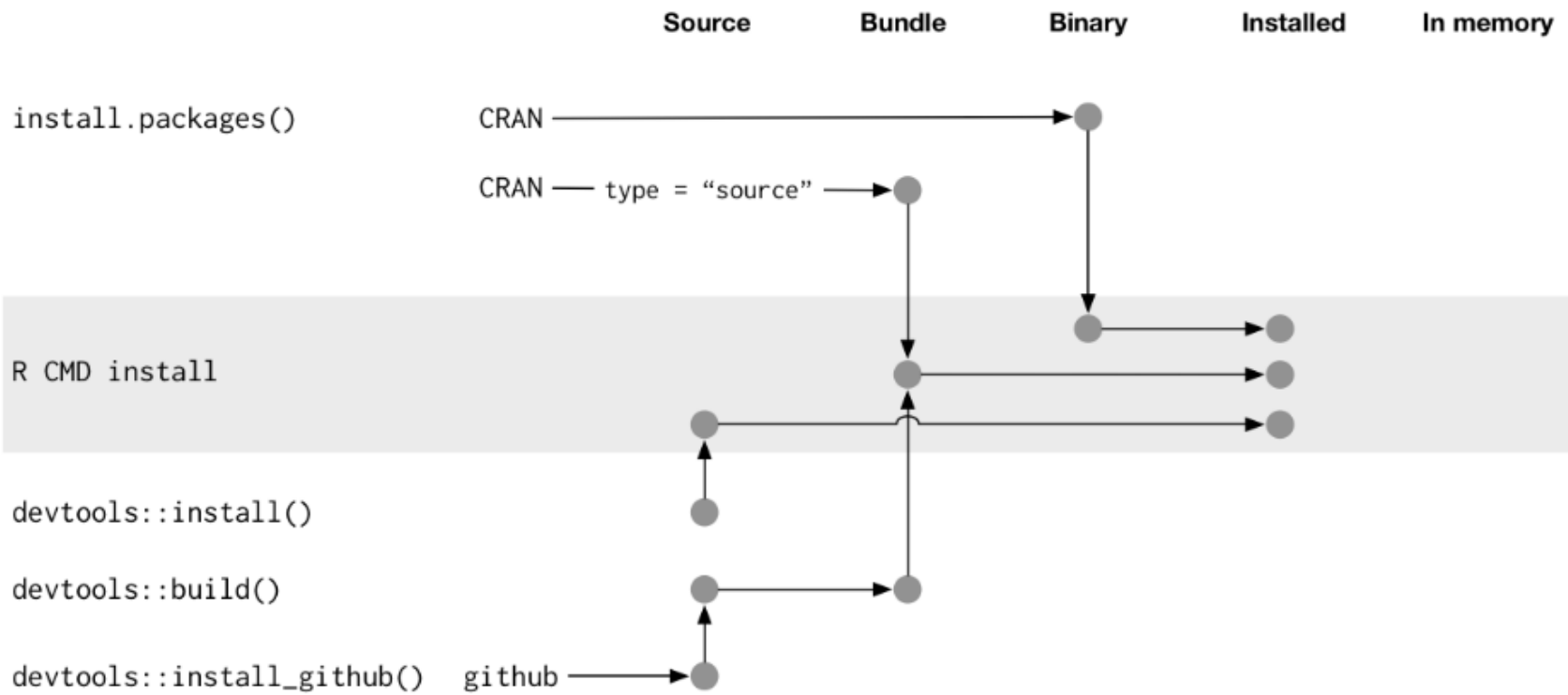


Pięć stanów pakietu

Pakiet R może się znajdować w jednym z 5 możliwych stanów:

- Source package- pakiet w trakcie tworzenia
- Bundled package- pakiet skompresowany do jednego pliku. Wymaga devtools.
- Binary package- projekt skompresowany w jednym pliku. Nie wymaga devtools.
- Instaled package- projekt zdekompresowany do biblioteki projektu. Jeszcze niedziałający w R.
- In memory package- Załadowany i działający w R.

Pięć stanów pakietu



1

[1] http://r-pkgs.had.co.nz/package.html

Uruchamianie pakietu R

Zazwyczaj z pakietów R korzystamy w następujący sposób:

- Instalujemy wprost z repozytorium CRAN za pomocą funkcji `install.package()`
- ładujemy do pamięci za pomocą funkcji `library()`

Nasz pakiet `XYZ jeszcze nie jest w repozytorium, nie jest też zainstalowany na naszym środowisku R.`
Najprostsze rozwiązanie to:

- `devtools::install()`
- `devtools::build()`

Plik DESCRIPTION- Przechowuje metadane o pakiecie.

- Każdy pakiet R zawiera plik DESCRIPTION.
- Tworząc pakiet za pomocą narzędzi `devtools create` tworzona jest domyślna struktura pliku.

```
Package: rp1
Type: Package
Title: What the Package Does (Title Case)
Version: 0.1.0
Author: Who wrote it
Maintainer: The package maintainer <yourself@somewhere.net>
Description: More about what it does (maybe more than one line)
    Use four spaces when indenting paragraphs within the Description.
License: What license is it under?
Encoding: UTF-8
LazyData: true
```

- Linia to **Pole:Wartość** (wartości w wielu liniach muszą być wcięte)
- Dodawanie informacji o zależnych pakietach.

```
Imports: dplyr(>= 0.3.0.1),
         purrr,
         data.table
```

Tworzenie dokumentacji dla '?' oraz `help()`

W R jest wiele form dokumentacji, a jedną z nich jest dokumentacja obiektów, które są argumentami funkcji [help].

Sposoby dokumentowanie obiektów:

- za pomocą plików [.Rd] w folderze [man/]. Składnia jest oparta na LaTeX (i renderowana np do HTML, pdf)
- za pomocą pakietu `Roxygen2`, który za pomocą komentarzy zamienia się w plik [Rd.].

Roxygen2:

- Dodajemy komentarz `#'` (inne niż zwykły komentarz)
- dodajemy ewentualnie tagi: `@tag tekst`
- uruchamiamy `devtools::document()`- Komentarze `#'` sa zamieniane na [.Rd].
- przeglądamy dokumentacje za pomocą ?

Dodajemy komentarze `roxygen`

- Komentujemy funkcje

hello.R | DESCRIPTION | myFirst1.R | selectNum.R

Source on Save

```
#' Tytul- W roxygen perwsze 3 linie naglowka maja specjalne znaczenie.
#' Jesli nie zrobily pustej linii komentarza to dalej mamy tytul.
#'
#' Opis- To mozemy dac dodatkowy opis. Pamietamy, że dopiero pusta linia rozdziela formatowanie.
#'
#' Szczegoly- Tutaj podajemy szczegoly funkcji. Ponadto mozemy dac @TAGi
#'
#' @param x is the numeric vector with values to verify
#' @param y is the next parameter
#'

isNumVec <- function(x){
  ifelse(is.numeric(x),
         yes = (sum(x, na.rm = T)),
         no = (stop("Zatrzymanie")) ) )
}
```

- uruchamiamy= `devtools::document()`
- przegadamy za pomocą ?

```
> ? isNumVec
>
```

Tworzenie dokumentacji dla '?' oraz `help()`

Rezultat

isNumVec {XYZ} R Documentation

Tytul- W roxygen perwsze 3 linie naglowka maja specjalne znaczenie. Jesli nie zrobily pustej linii komentarza to dalej mamy tytul.

Description

Opis- Tu mozemy dac dodatkowy opis. Pamietamy, że dopiero pusta linia rozdziela formatowanie.

Usage

```
isNumVec(x)
```

Arguments

`x`	is the numeric vector with values to verify
`y`	is the next parameter

Details

Szczegoly- Tutaj podajemy szczegoly funkcji. Ponadto mozemy dac @TAGi

[Package *XYZ* version 0.1.0 Index]

Dodatkowe tagi

- @import- importuje zależne pakiety
- @export- udostępnia funkcje poza pakiet (funkcja musi być zaraz za @export)

hello.R | DESCRIPTION | myFirst1.R | selectNum.R

Source on Save

```
#' Find and rename the numeric variables in the data frame
#'
#' Select the variable if the datatype is numeric.
#' Change the variable name to uppercase
#' Rename variable with prefix "Num_"
#' The dplyr package is used.
#'
#'
#' The dplyr is @imported
#'
#' @param x - a data.frame
#' @import dplyr
#' @export
selectNum <- function(x){
  select_if(x, is.numeric, function(i){paste("Num",toupper(i),sep="_")} )
}
```

Dlaczego dodajemy dane do pakietu, przykłady:

- Budujemy pakiet i chcemy dodać przykłady.
- Implementujemy test statystyczny i dodajemy specyficzne tablice.

Trzy sposoby na dodanie danych do pakietu:

- Folder [data/]- przechowywanie binarnych danych dostępnych dla użytkownika pakietu jako typ: `.RData`. Dodaj do DESCRIPTION- `LazyData: true`- wtedy zbiory będą dodane dopiero w momencie użycia.
- Folder [R/sysdata.rda]- przechowywanie danych dla funkcji- np. testy statystyczne `devtools::use_data ,internal = TRUE`
- Folder [inst/extdata]- zawiera surowe dane- jako przykład pracy z danymi w różnych formatach: .csv, .txt, sas7bat.

Tworzenie i dokumentowanie danych

- Tworzenie zbioru danych

```
> mydf <- data.frame(x= rnorm(10), y=rt(10,df = 5))
> usethis::use_data(mydf)
✓ Setting active project to 'C:/Users/tomas/Desktop/PackageXYZ/XYZ'
✓ Creating 'data/'
✓ Saving 'mydf' to 'data/mydf.rda'
>
```

- Dokumentowanie zbioru danych- tak jak w przypadku funkcji

hello.R | myFirst1.R | DESCRIPTION | selectNum.R | mydf.R

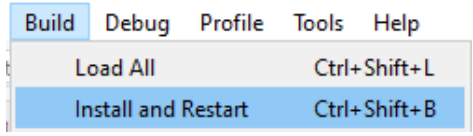
Source on Save

```
1  #' Tytul: Opis danych testowych zapisujemy w folderze [R]
2  #'
3  #' Opis danych: Nasze dane zawieraja tylko 20 obserwacji
4  #'
5  #' W celu poprawnego zapisania opisu danych musimy zrobic 2 rzeczy
6  #'
7  #' [1] podac nazwe zbioru danych na koncu na koncu tj. w tym przypadku "mydf"
8  #'
9  #' [2] uruchomic devtools::document()
10 #'
11 #' Efekty mozemy zobaczyc wpisujac ?mydf
12 #'
13 #' @format 10 na 2 gdzie:
14 #' \describe{
15 #' \item{x}{Losowe x- rozklad normalny standaryzowany}
16 #' \item{y}{Losowe y- rozklad t-Student df=5}
17 #' }
18 #'
19 #' @source \url{ue.katowice.pl}
20 #'
21 #' @author Tomasz Szkutnik
22 #'
23 "mydf"
24
```

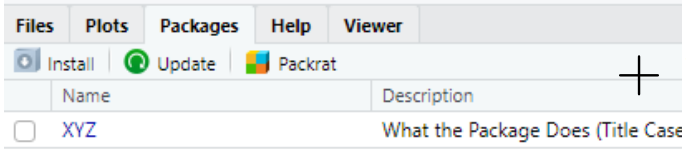
Aktualizacja pakietu

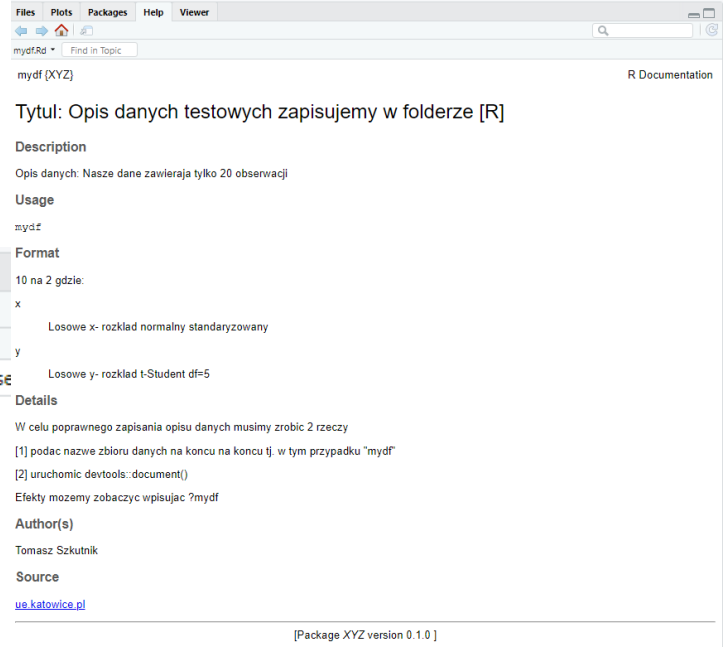
- Aktualizacja po zmianach

```
> devtools::document()
Updating XYZ documentation
Writing NAMESPACE
Loading XYZ
Writing NAMESPACE
Writing mydf.Rd
>
```



- Końcowa dokumentacja





Dziękuje