Package Ontwikkeling

met devtools Spiekbriefje

RStudio

Package Structuur

Een package is een conventie om bestanden in mappen op te slaan.

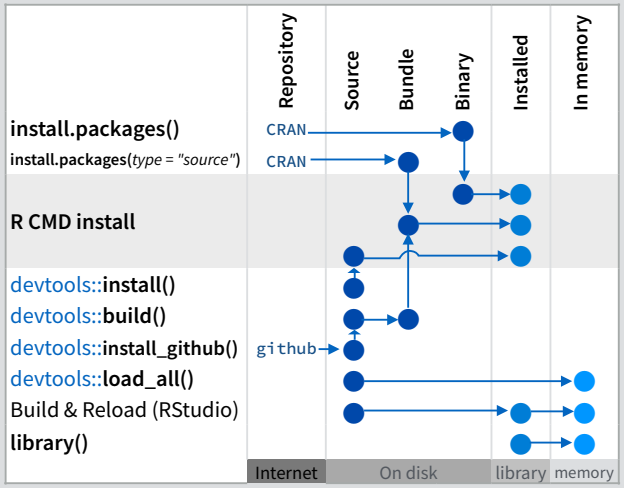
Dit document laat zien hoe om te gaan met de 7 meest voorkomende onderdelen van een R package:

```
Package
    DESCRIPTION   Opzet
    R/            Code
    tests/        Tests
    man/          Documentatie
    vignettes/    Instructie
    data/         Data toevoegen
    NAMESPACE     Organisatie
```

De inhoud van een package kan op schijf bewaard worden als een:

- **source** - een map met sub-mappen (zie boven)
- **bundle** - een gecomprimeerd bestand (*.tar.gz*)
- **binary** - een gecomprimeerd bestand geoptimaliseerd voor een specifiek besturingssysteem

De inhoud kan ook in een R bibliotheek geïnstalleerd worden (die in het geheugen geladen wordt tijdens R sessie) of online opgeslagen in een "repository". Gebruik de functies hieronder om tussen deze vormen te wisselen.



devtools::**add_build_ignore**(*"file"*)

Voegt bestanden toe aan .Rbuildignore, een lijst van bestanden die niet worden meegenomen als het package gemaakt wordt.

Opzet (DESCRIPTION)

Het DESCRIPTION bestand beschrijft je werk en regelt hoe jouw package samenwerkt met andere packages.

- Een DESCRIPTION bestand is verplicht.
- Voeg de packages de je nodig hebt toe met:
  devtools::**use_package()**
  Voegt een package toe aan het "Imports" veld (of "Suggests" veld als het tweede argument "Suggests" is).

CC0	MIT	GPL-2
Vrij om te gebruiken	Een MIT licentie is van toepassing op de code als deze opnieuw gebruikt wordt.	Een GPL-2 licentie is van toepassing op jouw code, *en alle code waarmee iemand het combineert*, als het opnieuw gebruikt wordt.

```
Package: mypackage
Title: Title of Package
Version: 0.1.0
Authors@R: person("Hadley", "Wickham", email =
    "hadley@me.com", role = c("aut", "cre"))
Description: What the package does (one paragraph)
Depends: R (>= 3.1.0)
License: GPL-2
LazyData: true
Imports:
    dplyr (>= 0.4.0),
    ggvis (>= 0.2)
Suggests:
    knitr (>= 0.1.0)
```

Importeer packages die jouw package absoluut nodig heeft. R installeert deze als het jouw package installeert.

Suggereer packages die niet perse nodig zijn. Gebruikers hebben de mogelijkheid deze handmatig te installeren

Code (R/)

Alle R code in jouw package gaat in R/. Een package met alleen een R/ map kan nog steeds heel bruikbaar zijn.

- Begin een nieuw package project met
  devtools::**create**(*"path/to/name"*)
  Maakt een sjabloon om een package te ontwikkelen.
- Bewaar je code in R/ als scripts (extension .R)

Werkwijze

1. Bewerk je code.
2. Laad je code met een van:
   devtools::**load_all()**
   Herlaadt alle *bewaarde* bestanden in R/ in het geheugen.
   Ctrl/Cmd + Shift + L (keyboard shortcut)
   Bewaart alle bestanden en roept dan load_all() aan.
3. Experimenteer in het werkscherm.
4. Herhaal.

- Volg de stijl van: **r-pkgs.had.co.nz/r.html#style**
- Klik op een functie en and druk op **F2** om de definitie te zien
- Zoek functies met **Ctrl + .**

Tests (tests/)

Gebruik tests/ om testregels te bewaren die je informatie geven als je code niet meer werkt.

- Voeg een **tests/** map toe en importeer **testthat** met
  devtools::**use_testthat()**
  Bereidt je package voor om geautomatiseerde tests te gebruiken met testthat
- Schrijf tests met **context()**, **test()**, en de verwachtte uitkomsten
- Bewaar je tests als .R bestanden in **tests/testthat/**

Werkwijze

1. Bewerk je code of tests.
2. Test je code met:
   devtools::**test()**
   Voert alle tests uit die *opgeslagen zijn* in tests/.
   Ctrl/Cmd + Shift + T
   (toetsenbord combinatie)
3. Herhaal totdat alle tests goed zijn

Voorbeeld test

```
context("Arithmetic")

test_that("Math works", {
  expect_equal(1 + 1, 2)
  expect_equal(1 + 2, 3)
  expect_equal(1 + 3, 4)
})
```

expect_equal()	is gelijk binnen een kleine numerieke tolerantie?
expect_identical()	is exact gelijk?
expect_match()	komt overeen met een gespecificeerde string of expressie?
expect_output()	drukt een gespecificeerd resultaat af?
expect_message()	toont een gespecificeerde mededeling?
expect_warning()	toont een gespecificeerde waarschuwing?
expect_error()	resulteert in gespecificeerde foutmelding?
expect_is()	het resultaat hoort in een bepaalde klasse?
expect_false()	retourneert FALSE?
expect_true()	retourneert TRUE?

Bezoek **r-pkgs.had.co.nz** voor meer informatie

Meer weten? http://r-pkgs.had.co.nz • devtools 1.6.1 • Updated: 1/15

info@rstudio.com • 844-448-1212 • rstudio.com

Meer weten? http://r-pkgs.had.co.nz • devtools 1.6.1 • Updated: 1/15

Documentatie (man/)

man/ bevat de documentatie voor jouw functies en de help pagina's in jouw package.

- [x] Gebruik roxygen commentaar om elke functie, naast zijn definitie, te beschrijven
- [x] Documenteer de naam van elke geëxporteerde data set
- [x] Voeg duidelijke voorbeelden voor elke functie toe

Werkwijze

1. Voeg roxygen commentaar toe aan je .R bestanden
2. Converteer roxygen commentaar in documentatie met één van:

   devtools::**document()**
   Converteert roxygen commentaar naar .Rd bestanden en plaatst ze in man/. Builds NAMESPACE.

   Ctrl/Cmd + Shift + D (Toetsenbord combinatie)
3. Open de help pagina's met **?** om de documentatie te controleren
4. Herhaal

.Rd format labels

```
\emph{italic text}
\strong{bold text}
\code{function(args)}
\pkg{package}

\dontrun{code}
\dontshow{code}
\donttest{code}

\deqn{a + b (block)}
\eqn{a + b (inline)}

\email{name@@foo.com}
\href{url}{display}
\url{url}

\link[=dest]{display}
\linkS4class{class}
\code{\link{function}}
\code{\link[package]{function}}

\tabular{lcr}{
  left \tab centered \tab right \cr
  cell \tab cell     \tab cell  \cr
}
```

Het **roxygen** package

Met roxygen kun je, in een verkorte vorm, je documentatie in de .R bestanden schrijven.

- Voegt roxygen documentatie als commentaar zinnen toe die beginnen met **#'**.
- Zet de commentaar zinnen direct boven de code die het gedocumenteerde object definieert.
- Zet een roxygen **@** label (rechts) na **#'** om een specifieke documentatie sectie te maken.
- Niet-gelabelde zinnen worden gebruikt om een titel, een beschrijving, en een details sectie te maken (in die volgorde)

```
#' Add together two numbers.
#'
#' @param x A number.
#' @param y A number.
#' @return The sum of \code{x} and \code{y}.
#' @examples
#' add(1, 1)
#' @export
add <- function(x, y) {
  x + y
}
```

Algemene roxygen labels

@aliases	@inheritParams	**@seealso**
@concepts	@keywords	@format
@describeIn	**@param**	@source (data)
@examples	@rdname	@include
@export	**@return**	@slot (S4)
@family	@section	@field (RC)

Instructie(vignettes/)

vignettes/ bevat documenten die de gebruikers leren echte problemen met de hulpmiddelen op te lossen.

- [x] Maak een vignettes/ map en een vignette sjabloon met

  devtools::**use_vignette()**
  Voegt een vignette sjabloon als as vignettes/my-vignette.Rmd toe.
- [x] Voeg YAML kopteksten aan je vignettes toe (zie rechts)
- [x] Schrijf de inhoud van je vignettes in R Markdown (rmarkdown.rstudio.com)

```
---
title: "Vignette Title"
author: "Vignette Author"
date: "`r Sys.Date()`"
output: rmarkdown::html_vignette
vignette: >
  %\VignetteIndexEntry{Vignette Title}
  %\VignetteEngine{knitr::rmarkdown}
  \usepackage[utf8]{inputenc}
---
```

Data toevoegen (data/)

De data/ map maakt het mogelijk om data aan jouw package toe te voegen.

- [x] Sla data op in een van **data/**, **R/Sysdata.rda**, **inst/extdata**
- [x] Gebruik altijd **LazyData: true** in je DESCRIPTION bestand.
- [x] Bewaar data als .Rdata bestanden (suggestie)

devtools::**use_data()**
Voegt een data object toe aan data/ (R/Sysdata.rda als **internal = TRUE**)

devtools::**use_data_raw()**
Voegt een R Script om een dataset op te schonen toe aan data-raw/. Voegt data-raw/ toe aan .Rbuildignore.

Sla data op in

- **data/** om de data beschikbaar te maken voor de gebruikers
- **R/sysdata.rda** houdt data intern voor jouw functies.
- **inst/extdata** maakt ruwe data beschikbaar om voorbeelden te laden en te ontleden. Benader deze data met **system.file()**

Organiseren (NAMESPACE)

Het NAMESPACE bestand helpt om je package op zichzelf staand te maken: daardoor heeft het geen invloed op andere packages en vice versa.

- [x] Exporteer functies voor gebruikers door **@export** in het roxygen commentaar van deze functies te zetten
- [x] Importeer objecten van andere packages met **package::object** (aanbevolen) of **@import**, **@importFrom**, **@importClassesFrom**, **@importMethodsFrom** (niet altijd aanbevolen)

Werkwijze

1. Bewerk je code of tests
2. Documenteer je package (devtools::**document()**)
3. Controleer NAMESPACE
4. Herhaal totdat NAMESPACE correct is

Stuur je package in

r-pkgs.had.co.nz/release.html