

# R-Packages

Statistics, Visualization and more using R

---

Bradley Mackay, Clemens Ehlich

NAWI PLUS

## I. The Theory

### 1. Basics

1.1 What's a package?

1.2 Why Packages?

1.3 Where can you find packages?

1.4 Install, Deinstall, Update

1.4.1 Installing packages via devtools

1.4.2 Using devtools commands for installation

## 2. Making our own package

2.1 Good Guide for starting

2.2 Components of a package

2.3 Description File

2.4 Namespace

2.5 R-Functions

2.6 Object Documentation and Manuals

2.7 Data, Vignettes, Demo

## 3 Sources

## II. Creating a Package Yourself

# I. The Theory

---

# 1. Basics



**Figure 1:** Package Universe

## 1.1 What's a package?

- R packages are collections of functions and data sets
- they extend the basic functionalities or add new ones
- mostly developed by the community itself

## 1.2 Why Packages?

- easy method for sharing your code with others
- recurring tasks - no need to reinvent the wheel
  - to load data
  - to manipulate data
  - to visualize data
  - etc. (more than 19.000 packages exist)
- packages form interfaces to:
  - other software and their file formats (foreign, caffe, RQGIS, ...)
  - databases (RODBC, RPostgreSQL, ...)
  - other programming languages (Rcpp, RPython, ...)
  - webservices (Rfacebook, RGoogleAnalytics, ...)

## 1.3 Where can you find packages?

- CRAN - The Comprehensive R Archive Network

<https://cran.r-project.org>

- BioConductor

<http://bioconductor.org/>

- GitHub

<https://github.com/>

- The most comfortable way is to use *RDocumentation*, because you can search more than 19.000 CRAN, Bioconductor and GitHub packages at once.

<https://www.rdocumentation.org/> also available as a package



## 1.4 Install, Deinstall, Update

### Install:

```
install.packages("xyz")
```

```
install.packages(c("xyz", "123"))
```

### Deinstall:

```
remove.packages("xyz")
```

### Update:

```
old.packages() → check what packages need an update
```

```
update.packages() → update packages
```

## 1.4.1 Installing packages via devtools

- one big problem: each repository has its own way to install a package from them
- to simplify this process you can use the package "devtools"
- but you might also need to install:
  - "Rtools" for Windows
  - "Xcode" for Mac
  - "r-devel and r-devel" for Linux

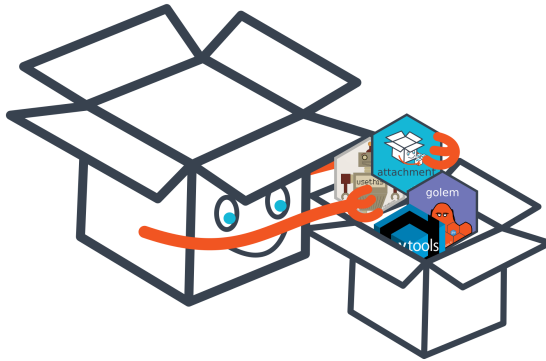
## 1.4.2 Installing packages via devtools

After devtools is installed, you will be able to use the utility functions to install another packages. Some options are:

- `install_bioc()` from Bioconductor
- `install_cran()` from CRAN
- `install_github()` from GitHub
- `install_local()` from a local file
- `install_url()` from a URL

Example: `devtools::install_github("hadley/babynames")`

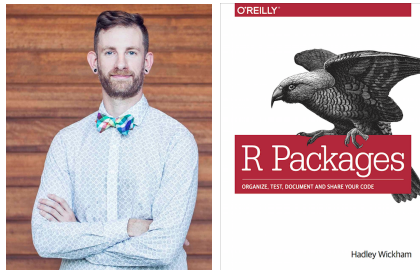
## 2. Making our own package



**Figure 2:** pack a pack

## 2.1 Making our own package

- Hadley Wickham - Chief Scientist at RStudio, Adjunct Professor of Statistics at the University of Auckland, Stanford University, and Rice University
- Free to read Book "R Packages": <http://r-pkgs.had.co.nz/>
- <https://cran.r-project.org/doc/manuals/r-release/R-exts.html>



**Figure 3:** H. Wickham and his Book

## 2.2 Components of a package

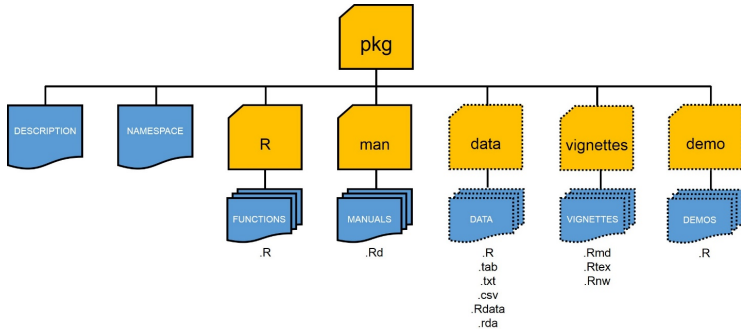
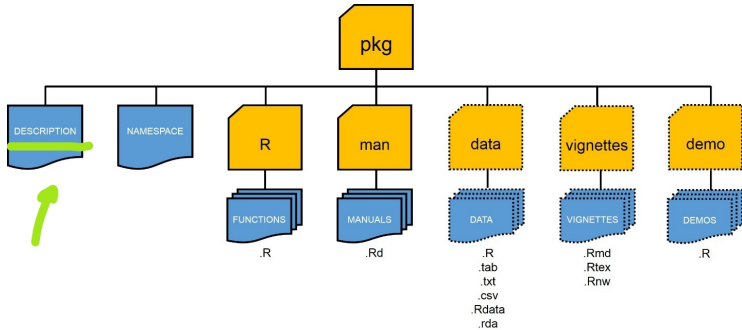


Figure 4: package structure

## 2.3 Description File



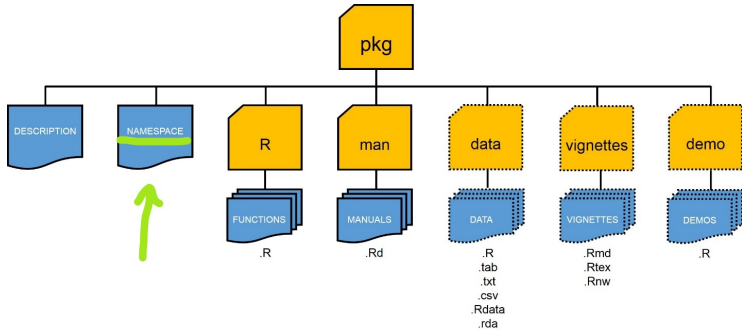
**Figure 5:** Description File

## 2.3 Description File

- Title
- Description
- Dependencies
  - list of necessary packages (and also package versions)
  - "LinkingTo" a Package - if you want to use c or c++ code from another package
- Author
- License (MIT, GPL-2, ...)



## 2.4 Namespace

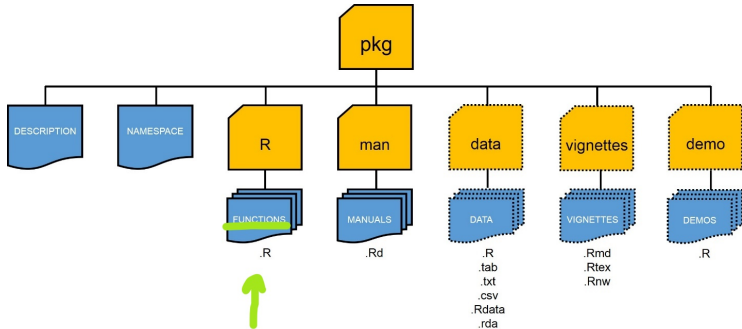


**Figure 6:** Namespace

## 2.4 Namespace

- With NAMESPACE you define the way in which your package interacts with other packages. For Example, to:
  - ensure that other packages won't interfere with the code you include
  - your code won't interfere with other packages
  - and that your package works regardless of the environment in which it's run
- Practical example: You are using two packages with ":: operator" and both have the summarize() function. Then it does matter in which order the packages are loaded in your code.

## 2.5 R-Functions

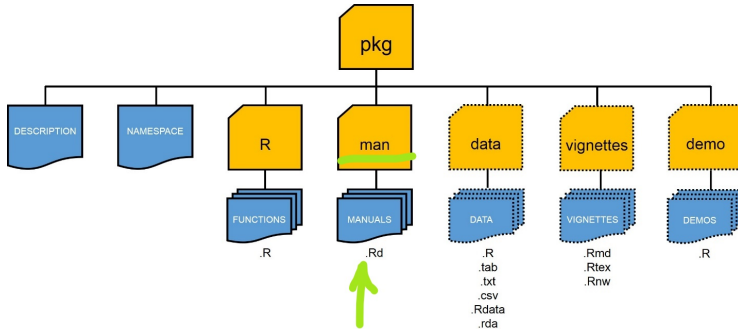


**Figure 7:** R-Functions

## 2.5 R-Functions

- include the functions and the R-Code itself
- with rcpp-package its posible to write C or C++ Code

## 2.6 Object Documentation and Manuals

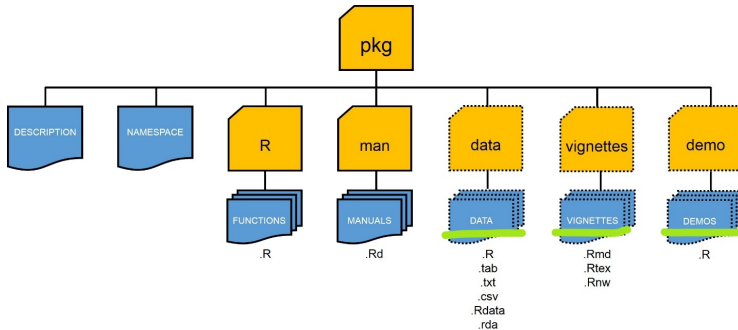


**Figure 8:** Object Documentation and Manual

## 2.6 Object Documentation and Manuals

- Good Documentation is one of the most important aspects of a good package
  - Documenting Functions, Classes, Generics and Methods
  - Documenting Datasets
  - Documenting Packages
  - ...
- R provides a standard way of documenting the objects in a package: you write .Rd files in the man/ directory.
- These files use a custom syntax, loosely based on LaTeX, and are rendered to HTML, plain text, and PDF for viewing.
- there are two ways to do this:
  - writing these files by hand
  - use the package "roxygen2" (recommended)

## 2.7 Data, Vignettes, Demo



**Figure 9:** Data, Vignettes, Demo

## 2.7 Data - External Input (optional)

- to include own data in a package
- three main ways to include data in your package:
  - store binary data (available for the user) in **data/**
  - store parsed data (not available for the user) in **R/sysdata.rda**
  - store raw data in **inst/extdata**



## 2.7 Vignettes (optional)

- vignettes are a long-form guide to your package
- describes a problem like a book chapter or an academic paper
- For Example: <https://cran.r-project.org/web/packages/dplyr/vignettes/colwise.html>
- The elegant way to build a vignette is to use RMarkdown and Knitr.
- Workflow:
  1. Create a vignettes/ directory.
  2. Add the necessary dependencies to DESCRIPTION
  3. Draft a vignette, vignettes/my-vignette.Rmd.

## 2.7 Demo (optional)

---

- Demos are like examples
- A demo is an .R file that lives in **demo/**
- Fuseful to the introduction of vignettes

### 3. Sources

---

Fig.1: <https://www.elasticfeed.com/ab4da588234cfa0cab5373f0b69ae5e/>

Fig.2: [https://rtask.thinkr.fr/wp-content/uploads/user2019\\_create\\_package\\_with\\_pkg.png](https://rtask.thinkr.fr/wp-content/uploads/user2019_create_package_with_pkg.png)

Fig.3: <http://r-pkgs.had.co.nz/cover.png>

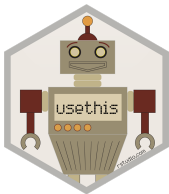
Fig.3: [https://www.mango-solutions.com/wp-content/uploads/2018/03/Hadley-Wickham\\_web.png](https://www.mango-solutions.com/wp-content/uploads/2018/03/Hadley-Wickham_web.png)

Fig.4-9: <https://methodsblog.files.wordpress.com/2015/11/stott-1.jpg?w=1024&h=464>

## **II. Creating a Package Yourself**

---

## Packages we will be using



## Packages we will be using

