RAPID Meeting

Your first R Package

Ewoud De Troyer

universiteit hasselt | I-BioStat | KU LEUVEN

Interuniversity Institute for Biostatistics
and statistical Bioinformatics

Why make R Packages?

- Version control your code.
- Organize your code in easy-to-load packages.
- Share your methodology with others.
- Easy to share, download and install R code.
- R has a huge community.
- Can include: *Python, C, Java,...*
- **It's surprisingly easy !!**

Installation & Example

Install

- R
- R Tools
- R package: `devtools`

Example Package

- `MyFirstPackage`
  - Plotting Download Numbers of R Packages (R Studio CRAN Mirror)
- Google Drive Link: `http://bit.ly/2hTGVAU`
- Demo
  - Load
  - Example
  - Documentation
  - Vignette

Outline

General Steps

1. Generate package structure
2. Update files for your package
   - `DESCRIPTION`
   - `NAMESPACE`
   (Handled by `roxygen2`)
3. Add your R scripts
4. *Optional:* Add data
5. Write your documentation (`roxygen2`)
6. *Optional:* Make a vignette
7. *Optional:* Add other folders in `inst`
8. Use `devtools` to :
   - build + document your package
   - check your package
   - release your package

General Steps

1. Generate package structure
2. Update files for your package
   - `DESCRIPTION`
   - `NAMESPACE`
     (Handled by `roxygen2`)
3. Add your R scripts
4. *Optional:* Add data
5. Write your documentation (`roxygen2`)
6. *Optional:* Make a vignette
7. *Optional:* Add other folders in `inst`
8. Use `devtools` to :
   - build + document your package
   - check your package
   - release your package

General Steps

1. Generate package structure
2. Update files for your package
   - `DESCRIPTION`
   - `NAMESPACE`
     (Handled by `roxygen2`)
3. Add your R scripts
4. *Optional:* Add data
5. Write your documentation (`roxygen2`)
6. *Optional:* Make a vignette
7. *Optional:* Add other folders in `inst`
8. Use `devtools` to :
   - build + document your package
   - check your package
   - release your package

General Steps

1. Generate package structure
2. Update files for your package
   - `DESCRIPTION`
   - `NAMESPACE`
     (Handled by `roxygen2`)
3. Add your R scripts
4. *Optional:* Add data
5. Write your documentation (`roxygen2`)
6. *Optional:* Make a vignette
7. *Optional:* Add other folders in `inst`
8. Use `devtools` to :
   - build + document your package
   - check your package
   - release your package

General Steps

1. Generate package structure
2. Update files for your package
   - `DESCRIPTION`
   - `NAMESPACE`
     (Handled by `roxygen2`)
3. Add your R scripts
4. *Optional:* Add data
5. Write your documentation (`roxygen2`)
6. *Optional:* Make a vignette
7. *Optional:* Add other folders in `inst`
8. Use `devtools` to :
   - build + document your package
   - check your package
   - release your package

General Steps

1. Generate package structure
2. Update files for your package
   - `DESCRIPTION`
   - `NAMESPACE`
     (Handled by `roxygen2`)
3. Add your R scripts
4. *Optional:* Add data
5. Write your documentation (`roxygen2`)
6. *Optional:* Make a vignette
7. *Optional:* Add other folders in `inst`
8. Use `devtools` to :
   - build + document your package
   - check your package
   - release your package

General Steps

1. Generate package structure
2. Update files for your package
   - `DESCRIPTION`
   - `NAMESPACE`
     (Handled by `roxygen2`)
3. Add your R scripts
4. *Optional:* Add data
5. Write your documentation (`roxygen2`)
6. *Optional:* Make a vignette
7. *Optional:* Add other folders in `inst`
8. Use `devtools` to :
   - build + document your package
   - check your package
   - release your package

General Steps

1. Generate package structure
2. Update files for your package
   - `DESCRIPTION`
   - `NAMESPACE`
     (Handled by `roxygen2`)
3. Add your R scripts
4. *Optional:* Add data
5. Write your documentation (`roxygen2`)
6. *Optional:* Make a vignette
7. *Optional:* Add other folders in `inst`
8. Use `devtools` to :
   - build + document your package
   - check your package
   - release your package

Outline

Making the package structure

How to create the skeleton of the package?

Base R

```
package.skeleton(name="MyFirstPackage")
```

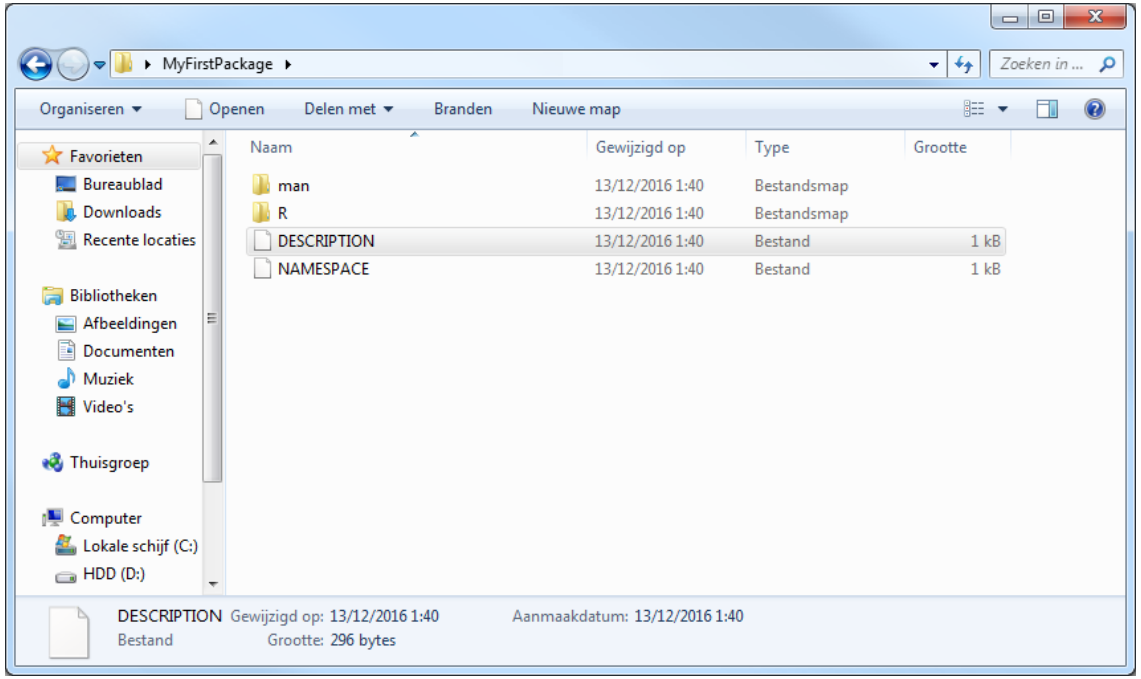
R Studio (Demo)

1. New Project
2. New Directory
3. R Package
4. Give package name + directory
5. Create Project

Package Outline

Created Folders & Files

- `R/` folder
  - `.R` scripts
- `man/` folder
  - `.Rd` documentation files
- `DESCRIPTION`
- `NAMESPACE`

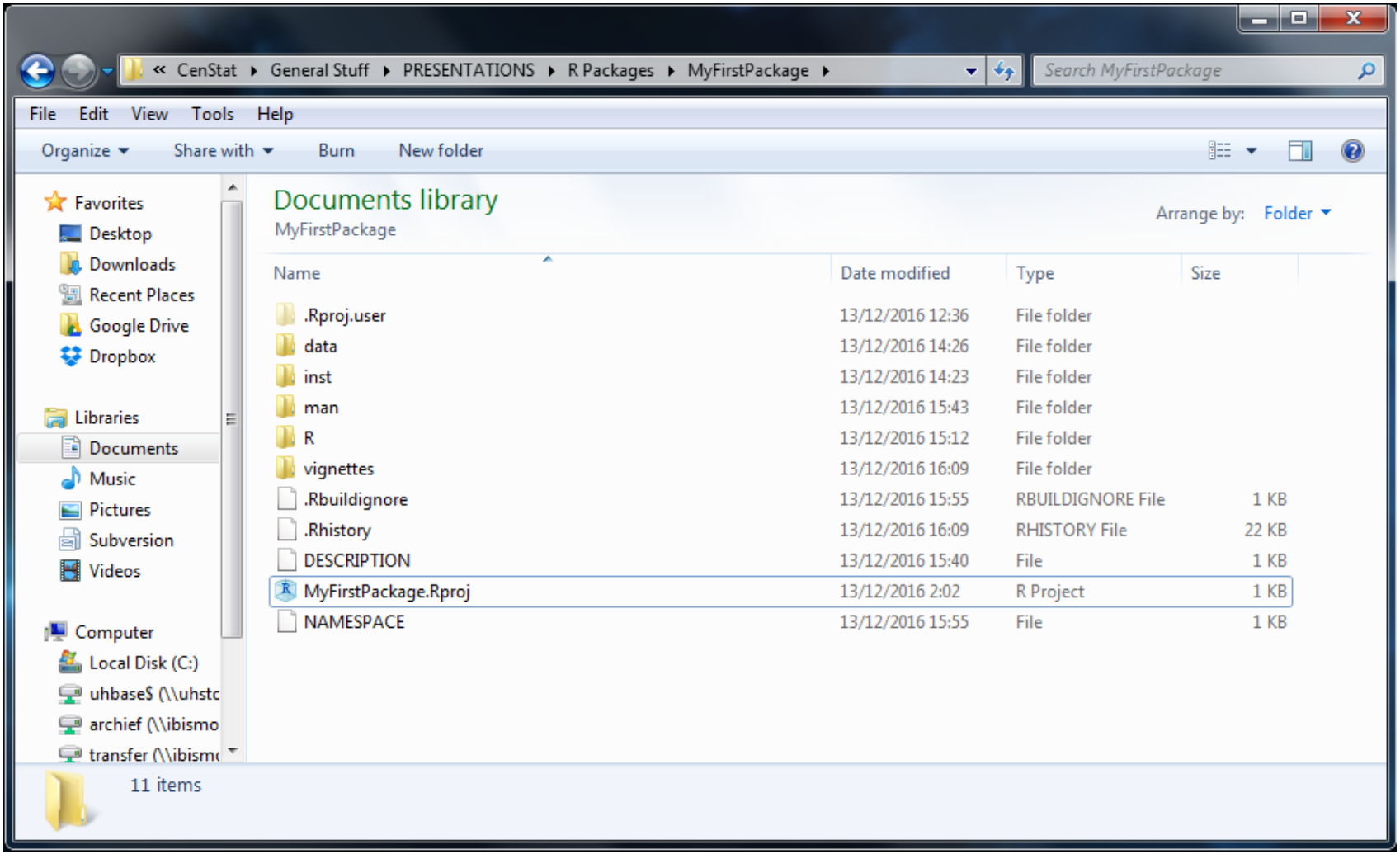


Package Outline

Other optional folders/files

- `data/` folder
- `inst/` folder
- `src/` folder
- `vignettes/` folder
- `tests/` folder
- `README` file
- `NEWS` file

Example Package - Package Outline



`R/` and `man/` folder

`R/`

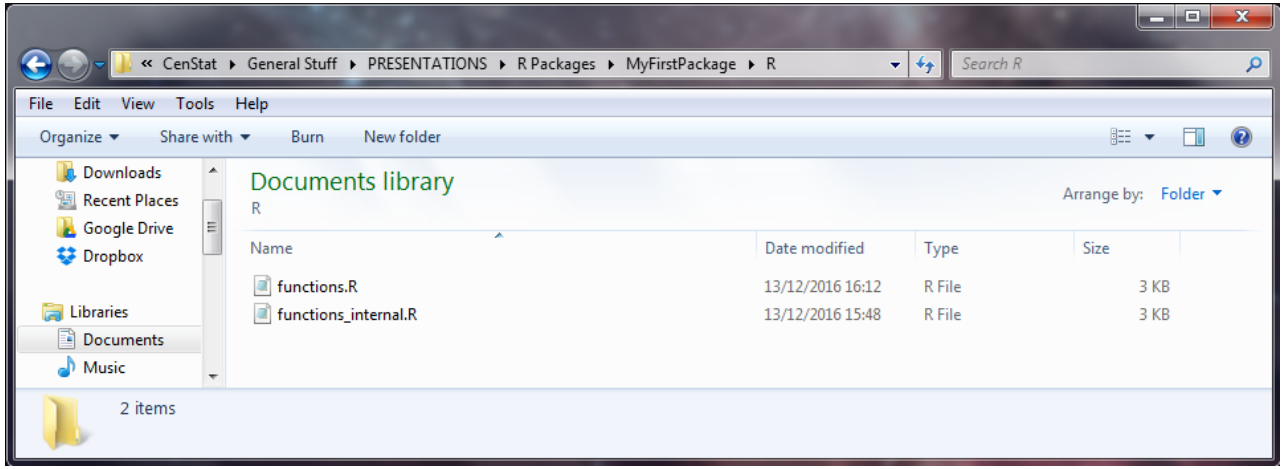
- R scripts
- As many as you want
- Contain internal and external functions

`man/`

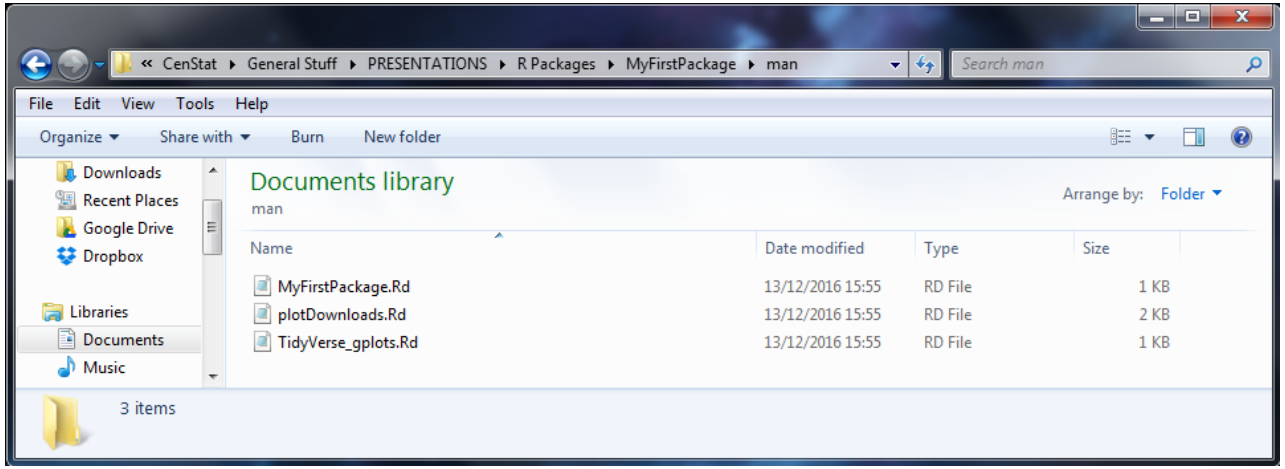
- `.Rd` files (1 per function)
- LaTeX-like syntax
- Can be done manually
- **Easier:** `roxygen2`

Example Package - R/ and man/ folder

R/



man/



Outline

DESCRIPTION - Default

→ A simple text file:

```
Package: MyFirstPackage
Type: Package
Title: What the package does (short line)
Version: 1.0
Date: 2016-12-13
Author: Who wrote it
Maintainer: Who to complain to
            <yourfault@somewhere.net>
Description: More about what it does
             (maybe more than one line)
License: What license is it under?
```

Example Package - DESCRIPTION

DESCRIPTION | MyFirstPackage.Rnw | functions.R | functions_internal.R | TidyVerse_gplots.R

```
Package: MyFirstPackage
Type: Package
Title: Plotting R Studio CRAN Mirror Download Numbers
Version: 1.0.0
Date: 2016-12-13
Author: Ewoud De Troyer
Maintainer: Ewoud De Troyer <ewoud.detroyer@uhasselt.be>
Description: Plots download numbers from R Studio CRAN Mirror with ggplot2.
License: GPL-3
Depends:
    ggplot2(>= 2.2.0)
Imports:
    cranlogs
Suggests: knitr
VignetteBuilder: knitr
RoxygenNote: 5.0.1
```

DESCRIPTION

Some CRAN Policies

- `Title`:
  Everything but conjunctions with **capital**.
- `Description`:
  Do **NOT** start with 'This package...', 'MyFirstPackage contains...', ...
- `Author`: Multiple allowed
- `Maintainer`: Only **one** allowed

Version Number x.y.z

- **x**, major release
- **y**, minor release
- **z**, patch

DESCRIPTION

- `License`: e.g. GPL-3 (= General Public License)
- `VignetteBuilder`:
  Optional, if including a knitr vignette (↔ sweave).

Including/Using other packages in your package?

- **NEVER USE `library()` OR `require()`!!!**
- `Depends`, `Imports`, `Suggests` field:
  - Add multiple packages on 1 line, separated by commas.

DESCRIPTION

- `License`: e.g. GPL-3 (= General Public License)
- `VignetteBuilder`:
  Optional, if including a knitr vignette (↔ sweave).

Including/Using other packages in your package?

- **NEVER USE `library()` OR `require()`!!!**
- `Depends`, `Imports`, `Suggests` field:
  - Add multiple packages on 1 line, separated by commas.

Outline

Depends/Imports

`Depends`

- Attaches packages before `library()`
- Loads entire package:
  - All exported functions
  - All included data
- *'Older'* → unless necessary, not preferred
  - e.g. When depending on an older package

`Imports`

- Imports only the namespace (does not attach package)
- You get option to:
  - Import **all** exported functions
  - Import **only specific** exported functions which you use
- *'More recent'* → preferred (future-proof).

Depends/Imports

`Depends`

- Attaches packages before `library()`
- Loads entire package:
  - All exported functions
  - All included data
- *'Older'* → unless necessary, not preferred
  - e.g. When depending on an older package

`Imports`

- Imports only the namespace (does not attach package)
- You get option to:
  - Import **all** exported functions
  - Import **only specific** exported functions which you use
- *'More recent'* → preferred (future-proof).

Depends/Imports

`Suggests`

- Works similar as `Depends`
- **Only** for packages used for:
  - Examples
  - Tests
  - Vignette

Example

- You can add **Version Dependency** (see below)

```
Depends:
    ggplot2(>= 2.2.0)
Imports:
    cranlogs
Suggests: knitr
```

- *Note:* `ggplot2` could be in `Imports` too

Depends/Imports

Base R Packages

- `graphics`, `grDevices`, `methods`, `stats`, `utils`,...
  - Contains functions like:
    `par`, `legend`, `new`, `write.table`, `colors`, `anova`, `coef`, `cor`,...
- **BEFORE**
  - Automatically imported
- **NOW**
  - Needs manual import
- *Checking* your package will prompt a NOTE or WARNING

Other DESCRIPTION Fields?

- Yes, see reference material (last slide)
- The ones shown, only ones you need most of the time

Outline

NAMESPACE

→ Initially confusing!

Why?

→ Makes your package 'play nice' with other packages

→ Package self-contained

- No interference in your code from other packages or vice versa

What?

→ Responsible for **imports** and **exports** of functions, classes and methods

How?

→ Edit manually in `NAMESPACE` file

→ Use `roxygen2`!

NAMESPACE - Default

- A simple text file:

```
exportPattern("^[[:alpha:]]+")
```

- No imported functions
- All functions starting with an alphanumerical symbol are exported

Example Package - NAMESPACE

DESCRIPTION | MyFirstPackage.Rnw | functions.R | function

```
# Generated by roxygen2: do not edit by hand

export(plotDownloads)
import(ggplot2)
importFrom(cranlogs,cran_downloads)

```

Example Package - NAMESPACE

Export

- Export `plotDownloads`
- Internal function `plot_cranlogs` not exported
  - Can not be imported in other packages
  - Can still be accessed through `:::` (Demo)
- All exported *functions/classes/methods* require **Documentation** !

Import

- Import all functions from `ggplot2` (Because we used `Depends`)
- Import `cran_downloads` function from `cranlogs` package

NAMESPACE - Attaching/Loading

More in-depth discussion of NAMESPACE

- http://r-pkgs.had.co.nz/namespace.html
- Use/Philosophy
- Example: Difference *Attaching/Loading*
  - Loading: '`::`' operator
  - Attaching: `library()` or `require()`

NAMESPACE - Manually

Common Commands

- `export(function1,function2)`
- `import(package1,package2)`
- `importFrom(package1,function1,function2)`

`import/importFrom` ?

- Dependant on `DESCRIPTION`
- `Depends: package1`
  - `import(package1)`
- `Imports: package2`
  - `import(package2`
  - `importFrom(package2,function1,function2)`

NAMESPACE - Manually

All Commands

- Exporting:
  - `export()`
  - `exportPattern()`
  - `exportClasses()`
  - `S3method()`
- Importing:
  - `import()`
  - `importFrom()`
  - `importClassesFrom()`
  - `useDynLib()` (import a function from C)
- Special functions required for importing/exporting S3/S4 methods/classes !
- S3/S4 methods/classes (see later slides) (e.g. `plot`, `summary`),...)

⇒ Use the `roxygen2` package!

NAMESPACE - Manually

All Commands

- Exporting:
  - `export()`
  - `exportPattern()`
  - `exportClasses()`
  - `S3method()`
- Importing:
  - `import()`
  - `importFrom()`
  - `importClassesFrom()`
  - `useDynLib()` (import a function from C)
- Special functions required for importing/exporting S3/S4 methods/classes !
- S3/S4 methods/classes (see later slides) (e.g. `plot`, `summary`),...)

⇒ Use the `roxygen2` package!

NAMESPACE - roxygen2

Roxygen Commands (After `#'` in your scripts)

- Exporting:
  - `@export` (functions, S3/S4 methods/classes)
  - `@exportClass` (specific cases;almost never used)
- Importing:
  - `@import pkg` (functions, S3)
  - `@importFrom pkg f1` (functions, S3)
  - `@importClassesFrom pkg classA classB` (S4 classes)
  - `@importMethodsFrom pkg GenericA GenericB` (S4 methods)
  - `@useDynLib`
- *Note:* Can call these multiple times in your script. Roxygen will make your NAMESPACE nice and tidy.
- Example → Documentation slides

NAMESPACE - roxygen2

Roxygen Commands (After `#'` in your scripts)

→ Exporting:

- `@export` (functions, S3/S4 methods/classes)
- `@exportClass` (specific cases;almost never used)

→ Importing:

- `@import pkg` (functions, S3)
- `@importFrom pkg f1` (functions, S3)
- `@importClassesFrom pkg classA classB` (S4 classes)
- `@importMethodsFrom pkg GenericA GenericB` (S4 methods)
- `@useDynLib`

→ *Note:* Can call these multiple times in your script. Roxygen will make your NAMESPACE nice and tidy.

→ Example → Documentation slides

NAMESPACE - roxygen2

Roxygen Commands (After `#'` in your scripts)

→ Exporting:

- `@export` (functions, S3/S4 methods/classes)
- `@exportClass` (specific cases;almost never used)

→ Importing:

- `@import pkg` (functions, S3)
- `@importFrom pkg f1` (functions, S3)
- `@importClassesFrom pkg classA classB` (S4 classes)
- `@importMethodsFrom pkg GenericA GenericB` (S4 methods)
- `@useDynLib`

→ *Note:* Can call these multiple times in your script. Roxygen will make your NAMESPACE nice and tidy.

→ Example → Documentation slides

NAMESPACE - roxygen2

Roxygen Commands (After `#'` in your scripts)

→ Exporting:

- `@export` (functions, S3/S4 methods/classes)
- `@exportClass` (specific cases;almost never used)

→ Importing:

- `@import pkg` (functions, S3)
- `@importFrom pkg f1` (functions, S3)
- `@importClassesFrom pkg classA classB` (S4 classes)
- `@importMethodsFrom pkg GenericA GenericB` (S4 methods)
- `@useDynLib`

→ *Note:* Can call these multiple times in your script. Roxygen will make your NAMESPACE nice and tidy.

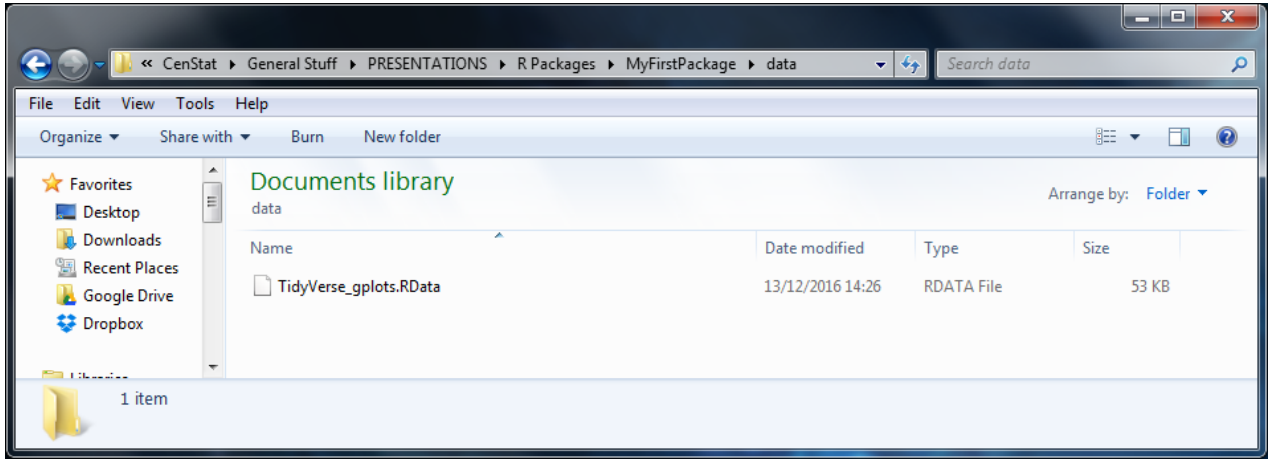
→ Example → Documentation slides

Outline

data/ folder - Example Package

data/

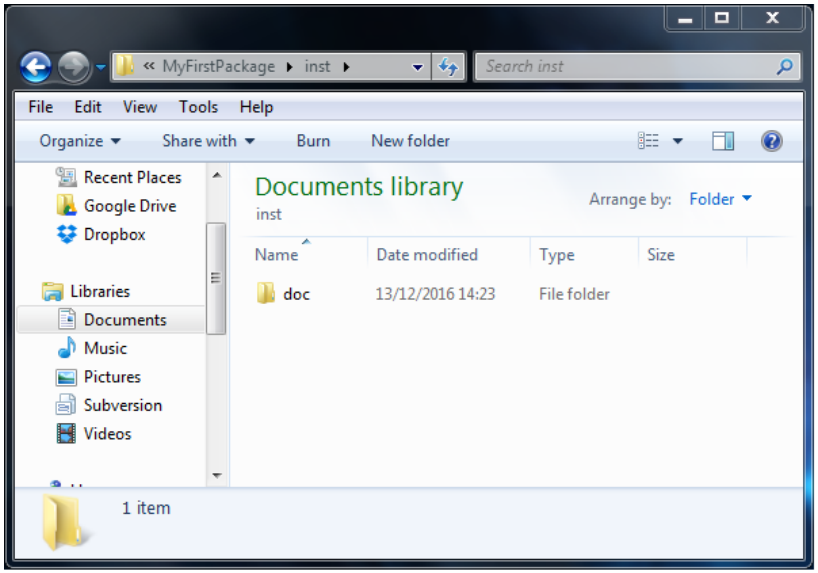
- Include datasets:
  - .R, .r, .tab, .txt, .csv (see ?data)
  - .RData, .rda (see ?save)
- Documentation required!
- Can lazy-load with LazyData field in DESCRIPTION
- Example: .RData containing list object of multiple ggplots of the tidyverse packages

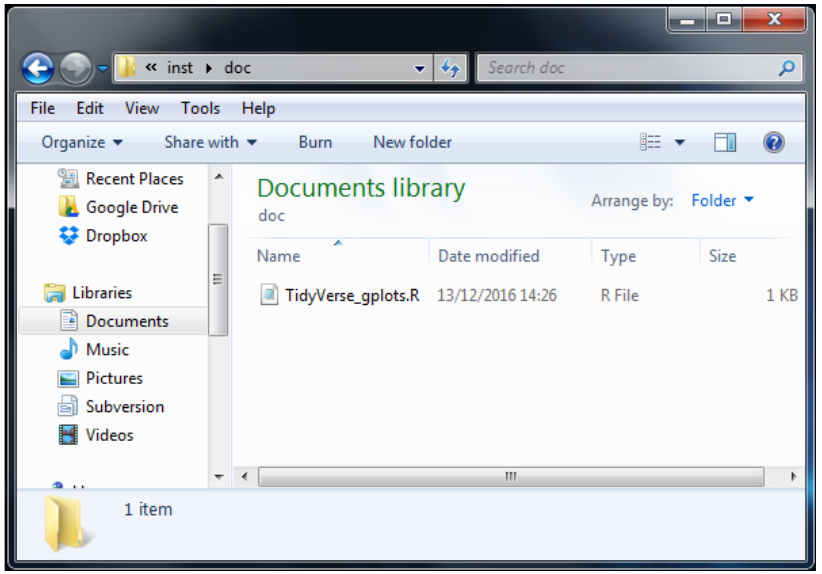


`inst/` folder - Example Package

`inst/`

- Can install other folders in the main directory of your package
- e.g. a documentation folder (`inst/doc/`)
  - After installation: `MyFirstPackage/doc/`
- Example: `.R` script containing how the example data was created

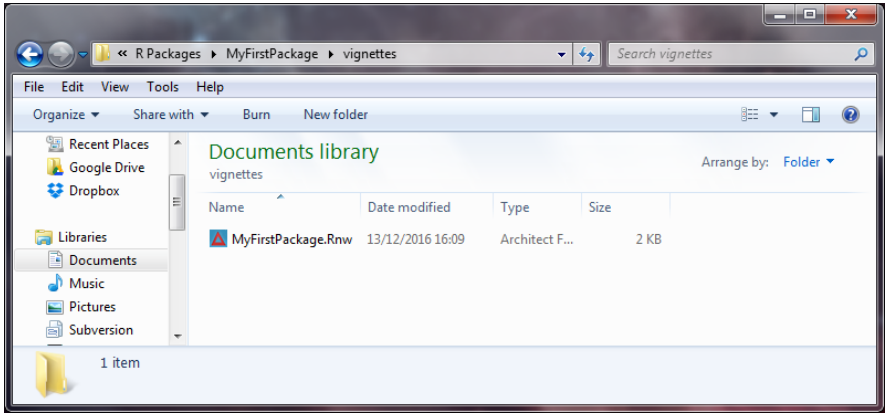




`vignettes/` folder - Example Package

`vignettes/`

- Include vignette in your package
  - Sweave/knitr
  - Extra info/documentation/workflow of package.
- Extra metadata at top of file (knitr Example):
  - `%\VignetteIndexEntry{MyFirstPackage: Vignette}`
    `%\VignetteDepends{MyFirstPackage}`
    `%\VignettePackage{MyFirstPackage}`
    `%\VignetteEngine{knitr::knitr}`
- Example: Contains some `plotDownloads` examples



`src/` and `tests` folder

`src/`

- Code that needs to be compiled
  - C
  - C++
  - FORTRAN
  - ...

`tests/`

- Contain unit tests
  - Does the code run in all standard scenarios?
  - Automate your 'testing' in a formal way
- Robust R code!
- `testthat` R package

`src/` and `tests` folder

`src/`

- Code that needs to be compiled
  - C
  - C++
  - FORTRAN
  - ...

`tests/`

- Contain unit tests
  - Does the code run in all standard scenarios?
  - Automate your 'testing' in a formal way
- Robust R code!
- `testthat` R package

README and NEWS file

README

- .txt, .md,...
- Aimed at new users
- Info about:
  - How do I get the package? (installation instructions)
  - How do I use it? (example)
  - Why should I use it? (some high-level info)

NEWS

- .txt, .md,...
- Aimed at existing users
- Info about:
  - Changes in release
  - Major changes
  - Bug fixes
- *Note:* .md not supported by CRAN, but by GitHub. (Ignore when building for CRAN!)

README and NEWS file

README

- .txt, .md,...
- Aimed at new users
- Info about:
  - How do I get the package? (installation instructions)
  - How do I use it? (example)
  - Why should I use it? (some high-level info)

NEWS

- .txt, .md,...
- Aimed at existing users
- Info about:
  - Changes in release
  - Major changes
  - Bug fixes
- *Note:* .md not supported by CRAN, but by GitHub. (Ignore when building for CRAN!)

Outline

Documentation

What to document?

- Package
- Functions
- Datasets
- Classes, Generics and Methods (S3/S4)

How?

- Manually create `.Rd` files in `man` folder.
  - Loosely based on LaTeX
- `roxygen2` package
  - Roxygen Comments
  - Start with `#'`
  - Split in *'blocks'* with *tags* (`@tag`)
  - Without tags, paragraphs automatically are: *title, description, details*

Documentation

What to document?

- Package
- Functions
- Datasets
- Classes, Generics and Methods (S3/S4)

How?

- Manually create `.Rd` files in `man` folder.
  - Loosely based on LaTeX
- `roxygen2` package
  - Roxygen Comments
  - Start with `#'`
  - Split in *'blocks'* with *tags* (`@tag`)
  - Without tags, paragraphs automatically are: *title, description, details*

Documentation

What to document?

- Package
- Functions
- Datasets
- Classes, Generics and Methods (S3/S4)

How?

- Manually create `.Rd` files in `man` folder.
  - Loosely based on LaTeX
- `roxygen2` package
  - Roxygen Comments
  - Start with `#'`
  - Split in *'blocks'* with *tags* (`@tag`)
  - Without tags, paragraphs automatically are: *title, description, details*

Documentation

What to document?

- Package
- Functions
- Datasets
- Classes, Generics and Methods (S3/S4)

How?

- Manually create `.Rd` files in `man` folder.
  - Loosely based on LaTeX
- `roxygen2` package
  - Roxygen Comments
  - Start with `#'`
  - Split in *'blocks'* with *tags* (`@tag`)
  - Without tags, paragraphs automatically are: *title, description, details*

Example Package - Documentation

DESCRIPTION | MyFirstPackage.Rnw | functions.R | functions_internal.R | TidyVerse_gplots.R | NAMESPACE | .Rbuildignore

Source on Save

```r

#######################################
## GENERAL PACKAGE DOCUMENTATION PAGE ##
#######################################


#' Plotting download numbers from R Studio CRAN Mirror
#'
#' MyFirstPackage is a test package which includes a function to plot the download numbers from the R Studio CRAN Mirror.
#'
#' @references Your submitted paper which is in revision for the last couple of months.
#'
#' @docType package
#' @name MyFirstPackage
NULL


#######################################
## IMPORTING OTHER PACKAGES/FUNCTIONS ##
#######################################


#' @import ggplot2
#' @importFrom cranlogs cran_downloads
NULL
```

Most Common Tags

→ `@title`, `@description`, `@details`, `@param`, `@examples`, `@references`, `@author`, `@export`,...

Example Package - Documentation

DESCRIPTION | MyFirstPackage.Rnw | functions.R | functions_internal.R | TidyVerse_gplots.R | NAMESPACE | .Rbuildignore

Source on Save | Run

```r
##################
## EXAMPLE DATA ##
##################
#' @title Tidyverse Plots
#'
#' @description A \code{.RData} object which contains a couple of download plots of some packages of the Hadley Tidyverse.
#'
#' @format A list with 4 ggplot objects.
#' @source R script in \code{doc/} folder
#' @name TidyVerse_gplots
NULL

###################
## MAIN FUNCTION ##
###################

#' @export
#' @title Plotting R Package Downloads by month or day.
#' @description This functions creates a ggplot2 graph of the number of downloads of a chosen package from the R Studio CRAN mirror.
#' The graph can be by month or by day.
#' @param PackageName Name of the package.
#' @param from Start date, in \code{yyyy-mm-dd} format.
#' @param to End date, in \code{yyyy-mm-dd} format.
#' @param by Plot downloads by \code{"day"} or \code{"month"}. It is advised to only use \code{"day"} for shorter periods of time to
#' avoid too much clutter.
#' @details When plotting by day, each month/year combination is assigned a different colour.
#' @return A \code{\link{ggplot2}} object based on the parameter settings.
#' @author Ewoud De Troyer
#' @examples
#' \dontrun{
#' plotDownloads("ggplot2",from="2015-01-01",to="2016-11-30")
#' plotDownloads("ggplot2",from="2016-04-01",to="2016-11-30",by="day")
#' }
#' @references Not really.
plotDownloads <- function(PackageName="ggplot2",from="2016-04-01",to="2016-11-30",by="month"){

  if(!(by %in% c("day","month"))){stop("by parameter should be either \"day\" or \"month\"")}

  x <- cran_downloads(packages=PackageName,from=from,to=to)
  gplot <- plot_cranlogs(x=x,from=from,to=to,by=by,name=PackageName)

  return(gplot)
}
```


universiteit hasselt

Documentation

Other Useful Tags

- Navigation
  - `@seealso`, `@family`, `@aliases`, `@keywords`
- `@section` (Add arbitrary sections)

Classes, Generics and Methods

- S3
  - Same as normal functions
  - *Note:* methods *can* be documented
- S4
  - Generics, methods → like normal functions
  - *Note:* methods *must* be documented
  - Classes:

```
#' An S4 class to represent a bank account
#'
#' @slot balance A length-one numeric vector
Account <- setClass("Account",
    slots = list(balance="numeric") )
```

Documentation - Formatting

- `\emph{italics}`: *italics*
- `\strong{bold}`: **bold**
- `\code{function}`: `function`
- `\url{www.xyz.com}`
- `\href{www.xyz.com}{xyz}`
- `\email{me@mail.com}`
- `\code{\link{function}}`
- `\code{\link[nlme]{lme}}`
- `\linkS4class{abc}`

Documentation - Formatting

→ `\enumerate{`
```
    \item First Item
}
```

→ `\itemize{`
```
    \item First Item
}
```

→ `\describe{`
```
    \item{One}{First Item}
}
```

→ `\eqn{a+b}`, `deqn(a+b)` (inline, block equation)

→ `\tabular{lr}{`
```
    1 \tab 2 \cr
    3 \tab 4 \cr
}
```

Documentation - Sources

More Info?

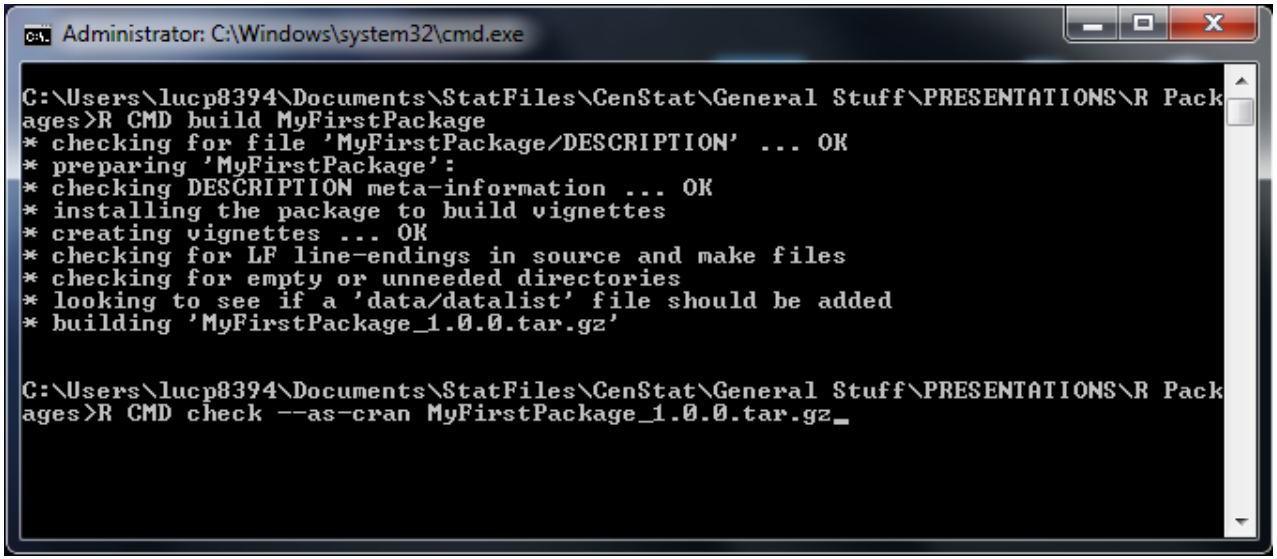
- Inheriting Parameters from Other Functions
- Documenting Multiple Functions in the Same File
- Documenting Reference Classes
- Other Roxygen Tags
- ...

Sources

- Links/books
- See last slide

Outline

Building/Checking Commands

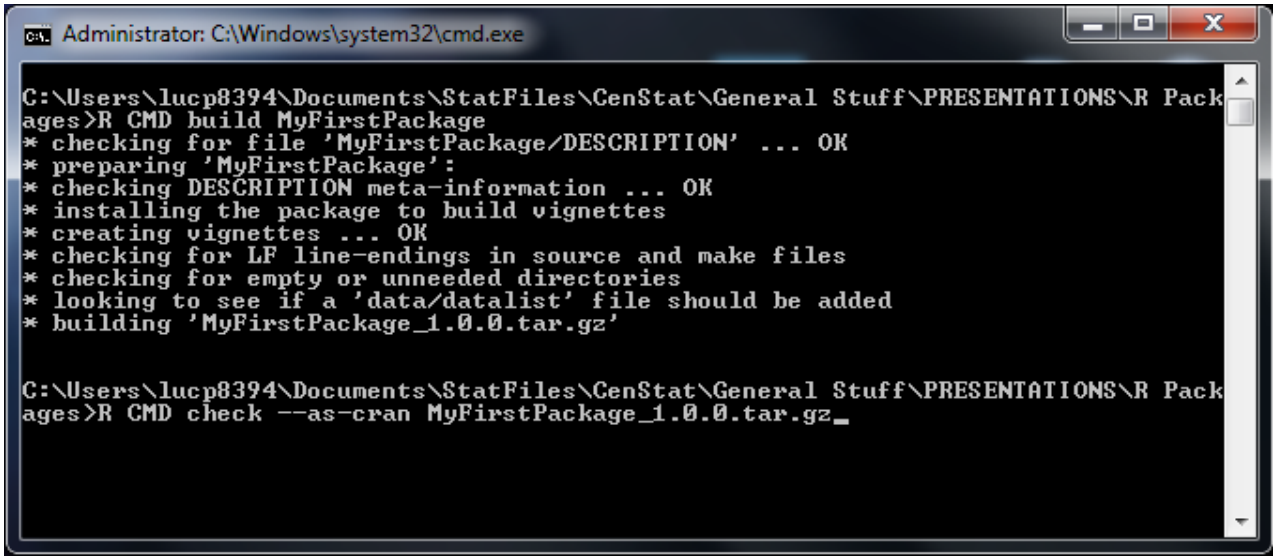


Use Commands in Windows Prompt

- Add `R` to the search path
- `R CMD build MyFirstPackage`
  `R CMD check --as-cran MyFirstPackage_1.0.0.tar.gz`
  `R CMD INSTALL --build MyFirstPackage_1.0.0.tar.gz`
  `...`

⇒ `devtools` R package to the rescue!

Building/Checking Commands



Use Commands in Windows Prompt

- Add `R` to the search path
- `R CMD build MyFirstPackage`
  `R CMD check --as-cran MyFirstPackage_1.0.0.tar.gz`
  `R CMD INSTALL --build MyFirstPackage_1.0.0.tar.gz`
  `...`

⇒ `devtools` R package to the rescue!

`devtools` R package

Documenting, Checking, Installing, Releasing Packages

- Use functions inside main package directory
  - `pkg` parameter
- `.Rbuildignore` file
  - Ignore files inside package directory (e.g. `cran-comments.md`)
  - Ignore for checking/building/installing/releasing
  - Files in 1 column (regular expressions allowed)

Other

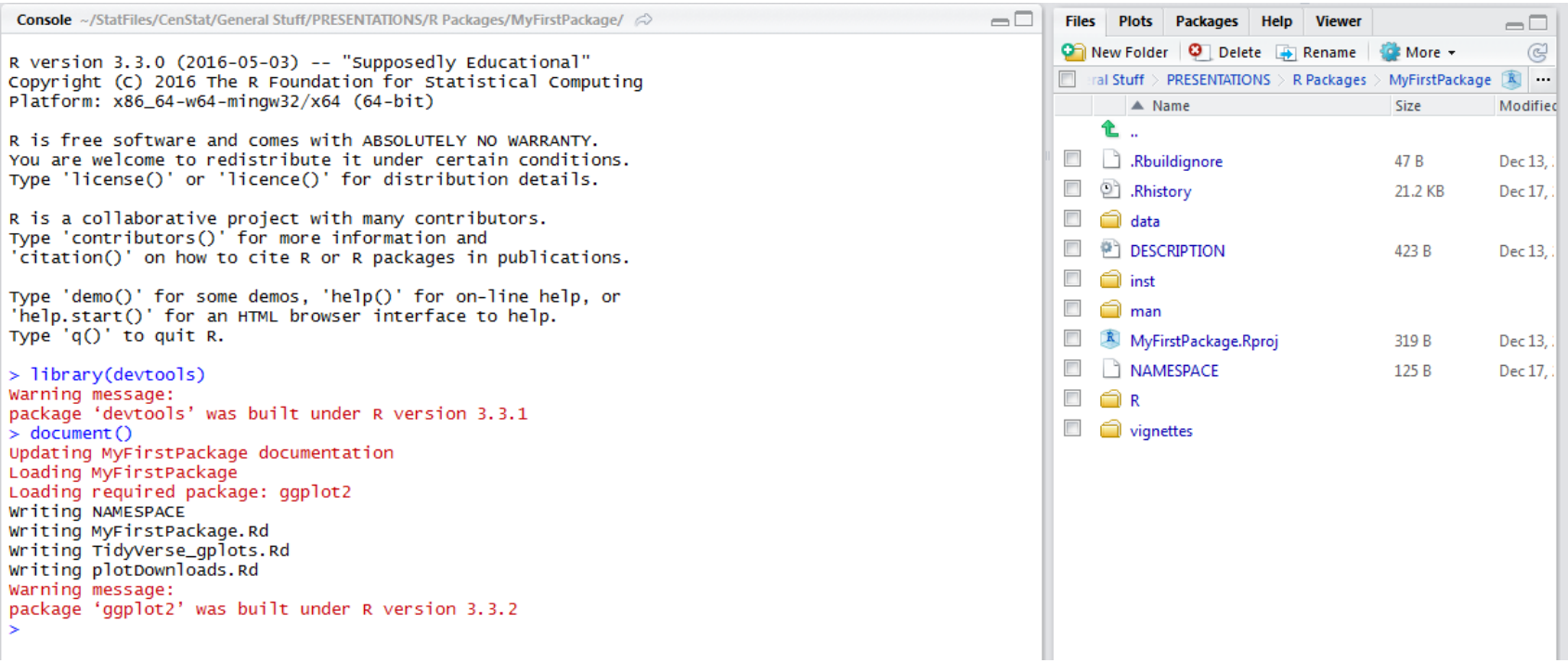
- Installing packages from GitHub
- Installing packages from Bioconductor
- ...

Outline

Generating Documentation

`document()` function

- Generates documentation files from roxygen comments:
  - `.Rd` files in `man/` folder
  - `NAMESPACE`
- Example:



Checking your build

`check()` function

- Applies `document()`
- Builds + Checks build for errors or violations of policies
- CRAN Check by default
- Returns `ERRORS`, `NOTES`, `WARNINGS`
- Example:

```
Console ~/StatFiles/CenStat/General Stuff/PRESENTATIONS/R Packages/MyFirstPackage/
package 'ggplot2' was built under R version 3.3.2
> check()
Updating MyFirstPackage documentation
Loading MyFirstPackage
Writing NAMESPACE
Writing MyFirstPackage.Rd
Writing TidyVerse_gplots.Rd
Writing plotDownloads.Rd
Setting env vars ----------------------------------------------------------------------------
CFLAGS  : -Wall -pedantic
CXXFLAGS: -Wall -pedantic
Building MyFirstPackage ---------------------------------------------------------------------
```

OMITTED OUTPUT

```
* checking examples ... OK
* checking for unstated dependencies in vignettes ... OK
* checking package vignettes in 'inst/doc' ... OK
* checking re-building of vignette outputs ... OK
* DONE

Status: 1 WARNING, 1 NOTE
See
  'C:/Users/lucp8394/AppData/Local/Temp/RtmpKgfzxH/MyFirstPackage.Rcheck/00check.log'
for details.


R CMD check results
0 errors | 1 warning  | 1 note
checking whether package 'MyFirstPackage' can be installed ... WARNING
Found the following significant warnings:
  Warning: package 'ggplot2' was built under R version 3.3.2
See 'C:/Users/lucp8394/AppData/Local/Temp/RtmpKgfzxH/MyFirstPackage.Rcheck/00install.out' for det
ails.

checking sizes of PDF files under 'inst/doc' ... NOTE
Unable to find GhostScript executable to run checks on size reduction
>
```

Files | Plots | Packages | Help | Viewer

New Folder | Delete | Rename | More

ral Stuff > PRESENTATIONS > R Packages > MyFirstPackage

Name	Size	Modified
..		
.Rbuildignore	47 B	Dec 13,
.Rhistory	21.2 KB	Dec 17,
data		
DESCRIPTION	423 B	Dec 13,
inst		
man		
MyFirstPackage.Rproj	319 B	Dec 13,
NAMESPACE	125 B	Dec 17,
R		
vignettes		

Checking your build

ERRORS

- Severe problems!
- Fix regardless if you submit to CRAN or not

WARNINGS

- *Likely* problems
- Fix if submitting to CRAN, if necessary

NOTES

- Mild problems
- Strive to eliminate if submitting to CRAN
  - False positives?
  - Add comments to your submission

Types of Issues?

- *Meta Data, Package Structure, Description, Namespace, R Code, Data, Documentation, Demos, Compiled Code, Tests, Vignettes*: e.g.
  - Undeclared global variables, Linewidth of `.Rd` files,...

Checking your build

ERRORS

- Severe problems!
- Fix regardless if you submit to CRAN or not

WARNINGS

- *Likely* problems
- Fix if submitting to CRAN, if necessary

NOTES

- Mild problems
- Strive to eliminate if submitting to CRAN
  - False positives?
  - Add comments to your submission

Types of Issues?

- *Meta Data, Package Structure, Description, Namespace, R Code, Data, Documentation, Demos, Compiled Code, Tests, Vignettes*: e.g.
  - Undeclared global variables, Linewidth of `.Rd` files,...

Checking your build

ERRORS

- Severe problems!
- Fix regardless if you submit to CRAN or not

WARNINGS

- *Likely* problems
- Fix if submitting to CRAN, if necessary

NOTES

- Mild problems
- Strive to eliminate if submitting to CRAN
  - False positives?
  - Add comments to your submission

Types of Issues?

- *Meta Data, Package Structure, Description, Namespace, R Code, Data, Documentation, Demos, Compiled Code, Tests, Vignettes*: e.g.
  - Undeclared global variables, Linewidth of `.Rd` files,...

Checking your build

ERRORS

- Severe problems!
- Fix regardless if you submit to CRAN or not

WARNINGS

- *Likely* problems
- Fix if submitting to CRAN, if necessary

NOTES

- Mild problems
- Strive to eliminate if submitting to CRAN
  - False positives?
  - Add comments to your submission

Types of Issues?

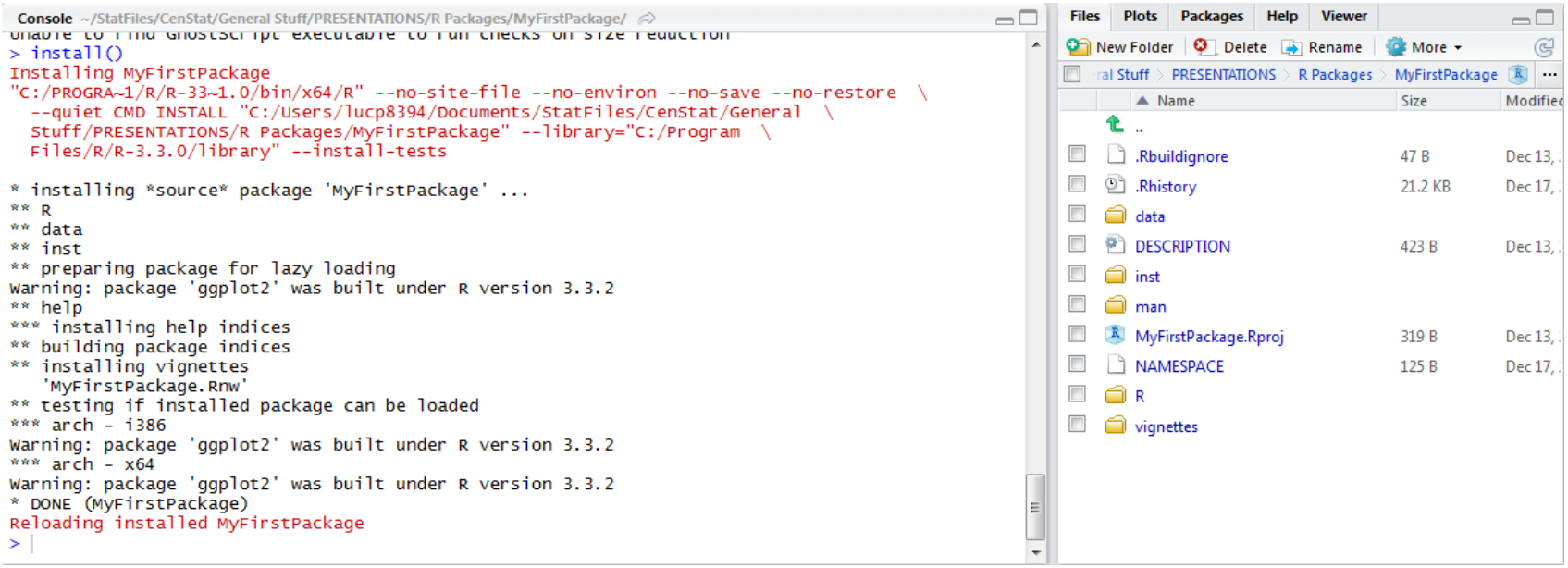
- *Meta Data, Package Structure, Description, Namespace, R Code, Data, Documentation, Demos, Compiled Code, Tests, Vignettes*: e.g.
  - Undeclared global variables, Linewidth of `.Rd` files,...

Outline

Installing your package

`install()` function

- Builds + Installs your package
- To build a vignette: `install(build_vignettes=TRUE)`
- *Note:* Could skip `check()`, but not advised
- Example:



Outline

Releasing your package

`release()` function

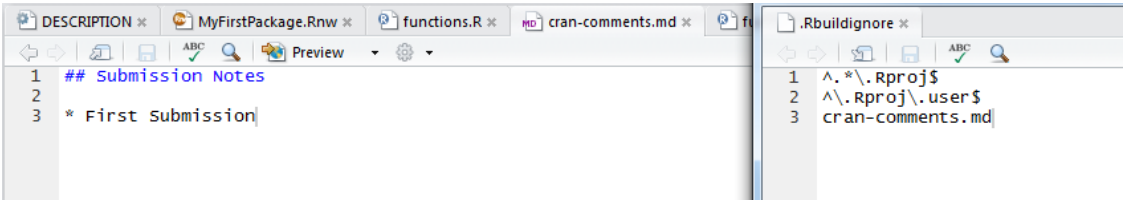
1. Builds + Checks Package
2. Asks yes/no questions to verify good practice
3. Upload package bundle to CRAN
   - Can include comments in `cran-comments.md`
   - `cran-comments.md` should be added to `.Rbuildignore`

→ Example:

Releasing your package

When your submission fails

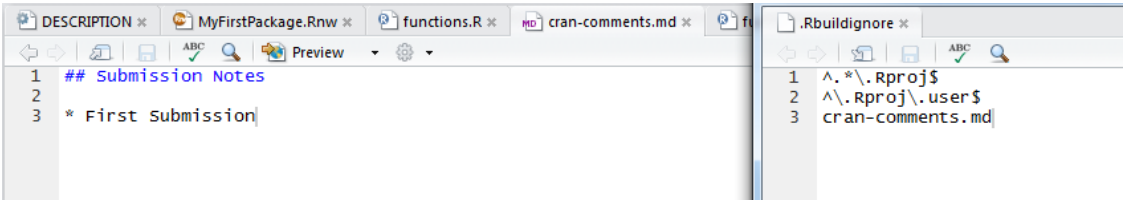
- Frustrating!
- Curt feedback, may feel insulting!
- Don't argue, breathe, have a cup of tea, cool down for a few days.
  - Everybody experiences this, even R-core members!
- Unless discussion merited:
  - Don't respond to mail directly!
  - Fix problems + make recommended changes
  - Change `cran-comments` to 'Resubmission' and list the changes.
- Example cran-comments:



Releasing your package

When your submission fails

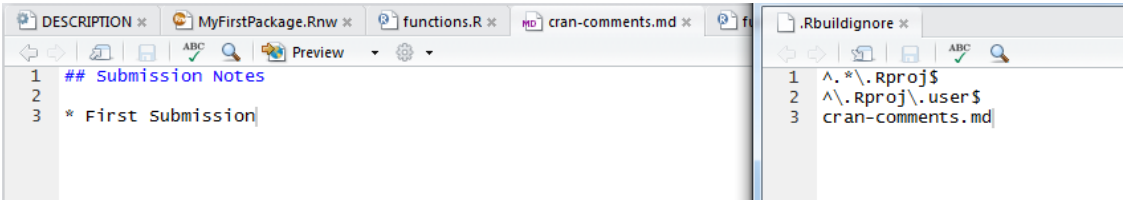
- Frustrating!
- Curt feedback, may feel insulting!
- Don't argue, breathe, have a cup of tea, cool down for a few days.
  - Everybody experiences this, even R-core members!
- Unless discussion merited:
  - Don't respond to mail directly!
  - Fix problems + make recommended changes
  - Change `cran-comments` to 'Resubmission' and list the changes.
- Example cran-comments:



Releasing your package

When your submission fails

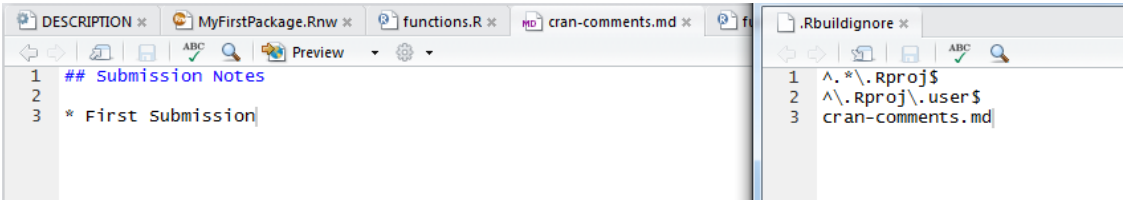
- Frustrating!
- Curt feedback, may feel insulting!
- Don't argue, breathe, have a cup of tea, cool down for a few days.
  - Everybody experiences this, even R-core members!
- Unless discussion merited:
  - Don't respond to mail directly!
  - Fix problems + make recommended changes
  - Change `cran-comments` to 'Resubmission' and list the changes.
- Example cran-comments:



Releasing your package

When your submission fails

- Frustrating!
- Curt feedback, may feel insulting!
- Don't argue, breathe, have a cup of tea, cool down for a few days.
  - Everybody experiences this, even R-core members!
- Unless discussion merited:
  - Don't respond to mail directly!
  - Fix problems + make recommended changes
  - Change `cran-comments` to 'Resubmission' and list the changes.
- Example cran-comments:



Building/Checking/Installing/Releasing - Overview

1. `document()` (If Step 2 is skipped)
2. `check()`
3. `install()` (installs package on your device)
   - If package includes vignette:
     `install(build_vignettes=TRUE)`
4. `release()`

Outline

S3 & S4 Objects

- Introduces structure and design
- Robust R code (validity checking)
- Functions respond differently to different input!
  - `plot`, `summary`, `print` (S3 Methods)
  - `ExpressionSet` (S4 Class)
  - *Note:* when using `@` instead of `$`, it's an S4 object!

Difference S3 & S4?

- S3
  - Quick & Easy!
  - Method only looks on type of *first* argument
- S4
  - More complex!
  - Method can look at types of *multiple* arguments.
  - Extra: check validity, convert objects,...

⇒ Short Example Introduction (Details in tutorials, last slide)

S3 & S4 Objects

- Introduces structure and design
- Robust R code (validity checking)
- Functions respond differently to different input!
  - `plot`, `summary`, `print` (S3 Methods)
  - `ExpressionSet` (S4 Class)
  - *Note:* when using `@` instead of `$`, it's an S4 object!

Difference S3 & S4?

- S3
  - Quick & Easy!
  - Method only looks on type of *first* argument
- S4
  - More complex!
  - Method can look at types of *multiple* arguments.
  - Extra: check validity, convert objects,...

⇒ Short Example Introduction (Details in tutorials, last slide)

S3 & S4 Objects

- Introduces structure and design
- Robust R code (validity checking)
- Functions respond differently to different input!
  - `plot`, `summary`, `print` (S3 Methods)
  - `ExpressionSet` (S4 Class)
  - *Note:* when using `@` instead of `$`, it's an S4 object!

Difference S3 & S4?

- S3
  - Quick & Easy!
  - Method only looks on type of *first* argument
- S4
  - More complex!
  - Method can look at types of *multiple* arguments.
  - Extra: check validity, convert objects,...

⇒ Short Example Introduction (Details in tutorials, last slide)

Outline

S3 - Example

S3.R

Source on Save Run Source

```
########
## S3 ##
########


## Examples ##

methods("summary")
#   [1] summary.aov                  summary.aovlist*             summary.aspell*              summary.check_packages_in_dir*
#   [5] summary.connection           summary.data.frame           summary.Date                 summary.default
#   [9] summary.ecdf*                summary.factor               summary.glm                  summary.infl*
#   [13] summary.lm                   summary.loess*               summary.manova               summary.matrix
#   [17] summary.mlm*                 summary.nls*                 summary.packageStatus*       summary.PDF_Dictionary*
#   [21] summary.PDF_Stream*          summary.POSIXct              summary.POSIXlt              summary.ppr*
#   [25] summary.prcomp*              summary.princomp*            summary.proc_time            summary.srcfile
#   [29] summary.srcref               summary.stepfun              summary.stl*                 summary.table
#   [33] summary.tukeysmooth*
methods(class="lm")
# [1] add1           alias          anova          case.names     coerce         confint        cooks.distance deviance
# [9] dfbeta         dfbetas        drop1          dummy.coef     effects        extractAIC     family         formula
# [17] hatvalues      influence      initialize     kappa          labels         logLik         model.frame    model.matrix
# [25] nobs           plot           predict        print          proj           qr             residuals      rstandard
# [33] rstudent       show           simulate       slotsFromS3    summary        variable.names vcov
```

→ `methods()`

- Look on which classes a method works
- Look which methods exist for a class

universiteit hasselt

S3 - Example

S3.R

Source on Save

```r

## Make S3 Class ##

CreateLunch <- function(food="sandwhiches",drink="water"){

  person <- list(food=food,drink=drink) # Person is a list class

  class(person) <- "Lunch"
  # or
  # class(person) <- append(class(person),"Lunch")
  return(person)
}

Nolen <- CreateLunch()
Nolen
# $food
# [1] "sandwhiches"
#
# $drink
# [1] "water"
#
# attr(,"class")
# [1] "Lunch"


## Extend existing method ##

summary.Lunch <- function(x){
  cat("This person ate",x$food,"and drank",x$drink)
}

summary(Nolen)
# This person ate sandwhiches and drank water

```

S3 - Example

S3.R

Source on Save

```r

## Create a new S3 Method ##

# Step 1 - Create a Generic (default)
ChangeLunch <- function(LunchObject,newFood,newDrink,...){
  print("We start at the base Generic")
  UseMethod("ChangeLunch",LunchObject)
  print("This is never executed!")
}


# Step 2 (Optional) - A default execution (For example: if the input is not what you expect it to be)
ChangeLunch.default <- function(LunchObject,newFood,newDrink,...){
  print("What happened? How did I get here?")
  return(LunchObject)
}

# Step 3 - Execute method on your class
ChangeLunch.Lunch <- function(LunchObject,newFood,newDrink){
  print("This person changed his mind what to eat and drink!")
  LunchObject$food <- newFood
  LunchObject$drink <- newDrink
  return(LunchObject)
}


Nolen <- ChangeLunch(Nolen,newFood="cake",newDrink="coffee")
# [1] "We start at the base Generic"
# [1] "This person changed his mind what to eat and drink!"

Nolen
# $food
# [1] "cake"
#
# $drink
# [1] "coffee"
#
# attr(,"class")
# [1] "Lunch"
summary(Nolen)
# This person ate cake and drank coffee


test <- 1:10
test <- ChangeLunch(test,newFood="cake",newDrink="coffee")
# [1] "We start at the base Generic"
# [1] "What happened? How did I get here?"
test
# [1]  1  2  3  4  5  6  7  8  9 10
```

S3 - Example

S3.R

Source on Save

```
## Inheritance ##
# General Idea: go more specialised
# Here: Toy Example

class(Nolen)
# [1] "Lunch"
class(Nolen) <- append(class(Nolen),"data.frame")
class(Nolen)
# [1] "Lunch"      "data.frame"
summary(Nolen)
# This person ate cake and drank coffee
class(Nolen) <- c("data.frame","Lunch")
summary(Nolen)
# food               drink
# Length:1            Length:1
# Class :character    Class :character
# Mode  :character    Mode  :character
```

- `class` attribute can be a vector
- `R` will look for a method for each element in turn until it finds one
- Use `NextMethod()` to call the next method in the inheritance.
- *Note:*
  - Inheritance not used much
  - Statisticians extend by generalization, not specialization

S3 - Problems

Simple and powerful system... BUT

- Not a real class system, more set of naming conventions
- Classes attached to simple attributes
- Method dispatch looks at **one** (often first) argument!

- No validation if objects are valid for a certain class!

S3.R

Source on Save

```
## No Validation ##
Ewoud <- "I am just a string in disguise"
class(Ewoud) <- "Lunch"
summary(Ewoud)
# Error in x$food : $ operator is invalid for atomic vectors
```

S3 - Problems

Simple and powerful system... BUT

- Not a real class system, more set of naming conventions
- Classes attached to simple attributes
- Method dispatch looks at **one** (often first) argument!
- No validation if objects are valid for a certain class!

S3.R

Source on Save

```
## No Validation ##
Ewoud <- "I am just a string in disguise"
class(Ewoud) <- "Lunch"
summary(Ewoud)
# Error in x$food : $ operator is invalid for atomic vectors
```

S3 - Problems

Simple and powerful system... BUT

- Not a real class system, more set of naming conventions
- Classes attached to simple attributes
- Method dispatch looks at **one** (often first) argument!
- No validation if objects are valid for a certain class!

S3.R

Source on Save

```
## No Validation ##
Ewoud <- "I am just a string in disguise"
class(Ewoud) <- "Lunch"
summary(Ewoud)
# Error in x$food : $ operator is invalid for atomic vectors
```

S3 - Problems

Simple and powerful system... BUT

- Not a real class system, more set of naming conventions
- Classes attached to simple attributes
- Method dispatch looks at **one** (often first) argument!

- No validation if objects are valid for a certain class!

S3.R

Source on Save

```
## No Validation ##
Ewoud <- "I am just a string in disguise"
class(Ewoud) <- "Lunch"
summary(Ewoud)
# Error in x$food : $ operator is invalid for atomic vectors
```

S3 - Problems

Simple and powerful system... BUT

- Not a real class system, more set of naming conventions
- Classes attached to simple attributes
- Method dispatch looks at **one** (often first) argument!

- No validation if objects are valid for a certain class!

S3.R

Source on Save

```
## No Validation ##
Ewoud <- "I am just a string in disguise"
class(Ewoud) <- "Lunch"
summary(Ewoud)
# Error in x$food : $ operator is invalid for atomic vectors
```

Outline

Access Registry / Find Methods & Classes

- `showClass()`
- `showMethods()`
- `getMethod()`
- `selectMethod()`
- `existsMethod(),hasMethod()`
- `removeClass(),removeMethod()`,...

S4 - Example

S3.R S4.R

Source on Save Run Source

```
########
## S4 ##
########

## Examples ##


library(Biobase)
showClass("ExpressionSet")
# Class "ExpressionSet" [package "Biobase"]
#
# Slots:
#
#   Name:      experimentData          assayData          phenoData        featureData         annotation       protocolData
# Class:              MIAME          AssayData AnnotatedDataFrame AnnotatedDataFrame          character AnnotatedDataFrame
#
# Name:   .__classVersion__
# Class:           Versions
#
# Extends:
#   Class "eSet", directly
# Class "VersionedBiobase", by class "eSet", distance 2
# Class "Versioned", by class "eSet", distance 3
```

S4 - Example

S3.R × S4.R ×

Source on Save

```r
## Make S4 Class ##


FoodOrderClass <- setClass(
  # Set the name for the class
  Class="FoodOrder",

  # Define the slots
  slots = c(
    food = "character",
    number = "numeric"
  ),

  # Set the default values for the slots. (optional)
  prototype=list(
    food = "sandwich",
    number = 10
  ),

  # Make a function that can test to see if the data is consistent.
  # This is not called if you have an initialize function defined!
  validity=function(object)
  {
    food_available <- c("sandwich","salad","salmon","mozarella","tomato")

    if(!all(object@food %in% food_available)){
      return("Not all ordered food is available!")
    }

    if(length(object@food)!=length(object@number)){
      return("Not all food items have an associated number!")
    }

    return(TRUE)
  }
)
```

S4 - Example

S3.R × S4.R ×

Source on Save

```
# Define it with the function
Nolen <- FoodOrderClass(food=c("sandwich","salmon"),number=c(10,25))
Nolen
# An object of class "FoodOrder"
# Slot "food":
#   [1] "sandwich" "salmon"
#
# Slot "number":
#   [1] 10 25
Nolen@food
# [1] "sandwich" "salmon"

# Or define it with new() function
Ewoud <- new("FoodOrder",food=c("sandwich","mozarella","tomato"),number=c(10,15,5))
Ewoud
# An object of class "FoodOrder"
# Slot "food":
#   [1] "sandwich"  "mozarella" "tomato"
#
# Slot "number":
#   [1] 10 15  5


# Validity Checking:
test <- FoodOrderClass(food=c(1,2,3),number=c(1,2,3))
# Error in validObject(.Object) :
#   invalid class "FoodOrder" object: invalid object for slot "food" in class "FoodOrder":
#   got class "numeric", should be or extend class "character"
test <- FoodOrderClass(food=c("sandwich","beer"),number=c(1,2))
# Error in validObject(.Object) :
#   invalid class "FoodOrder" object: Not all ordered food is available!
```


universiteit hasselt

S4 - Example

~/StatFiles/CenStat/General Stuff/PRESENTATIONS/R Packages/MyFirstPackage - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

S3.R S4.R

Source on Save

```r

## Make S4 Method ##


# Make the generic (This is skipped if you extend existing method, e.g. pData() )
setGeneric(name="ComputePrice",
           def=function(OrderObject,discount=0)
           {
             standardGeneric("ComputePrice")
           }
)

# Set a method for the new generic
setMethod(f="ComputePrice",
          signature=c("FoodOrder","numeric"),

          definition=function(OrderObject,discount=0)
          {

            food_prices <- c(sandwich=1.75,salad=1.25,salmon=2.25,mozarella=1.5,tomato=0.75)
            price <- 0
            for(i in 1:length(OrderObject@food)){
              price <- price +  food_prices[OrderObject@food[i]]*OrderObject@number[i]
            }
            price <- round((1-discount)*price,2)
            names(price) <- NULL

            cat("Total Price of Food Order:",price,"euro")
            return(price)
          }
)


price <- ComputePrice(Nolen,discount=0.1)
# Total Price of Food Order: 66.38 euro
price
# [1] 66.38

```


universiteit hasselt

S4

Other S4 Options

- Multiple Dispatch - Special Signatures
  - `ANY` (Any class, like S3 default)
  - `MISSING`
- Making new classes which include existing classes
- `is(object,"class")` tests whether `object` inherits from `class`
- `as(object,"class")`
- Object Conversion (`setAs()`)
- Class Inheritance
- Group generics/methods
- Replacement methods
- Calling ‘next’ method
- Class unions

S4

S4 Problems

- Documentation/Reference material
- Slower than S3

Examples

- CRAN: `biclust`, `Matrix`, `lme4`,...
- Bioconductor: `Biobase`, `DESeq`,...

Namespace

- Import functions from `methods` R package

Outline

Other Topics

What's more?

- Unit Tests
  - `testthat` R package
- Including compiled code from C, Python, Java,...
- Demos
- External Data
- Creating/Using Environments in your package
  - Global variables not allowed
  - Make variables in package environment
- Using repositories:
  - Git
  - GitHub
  - R-Forge
- ...

Links/Tutorials/Books

Reference Material

- *'R Packages'*, Hadley Wickham (`http://r-pkgs.had.co.nz/`)
  - e.g. Namespace explanation
- *'Writing R Extensions'*, in R -> help -> manuals (Extensive, but difficult!)
- Roxygen2 Introduction, `https://cran.r-project.org/web/packages/roxygen2/vignettes/roxygen2.html`
- `https://www.r-bloggers.com/`
- `http://stackoverflow.com/`
- S3/S4
  - 2 presentations in `doc/S3_S4/` folder
  - `http://www.cyclismo.org/tutorial/R/s3Classes.html`
  - `http://www.cyclismo.org/tutorial/R/s4Classes.html`