

R programming

... and extensions

YYT-C3002 - Application
Programming in Engineering

Marko Kallio

31 Jan 2019



Aalto University
School of Engineering



Today

I assume you already know something of R – no absolute basics!

- **Background on R**
- **Basic concepts**
 - Objects
 - Functions
- **Extending R via functions**
- **Extending R via packages**

R language

R has origins in statistics

- **Open source implementation of S language (think of matlab / octave)**
- **First version in 1993 by Ross Ihaka and Robert Gentleman**
- **Popular analysis tool among statisticians and data scientists**
- **FOSS**

R language

Like matlab, R is

- Programming language
- Programming environment
- Function library
- Application Program Interface (API)
- *Designed for interactive use*

R language

R is a procedural programming language with support for functional and object-oriented workflows.

- **Everything in R is an object**
 - Every function or data structure can be modified

R language

R has built in interoperability capabilities between languages.

- C
- C++
- FORTRAN

Through packages

- Python (e.g. via *reticulate*)
- SQL
- more...

Data structures and types

Structures

Dimension	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

Types

- Logical
- Integer
- Double (numeric)
- Character

- (Complex)
- (Raw)

Functions

Call functions with *function(arguments)*

- `sum(1:5)`
- `mean(c(1, 3, 5, 6, 8, 10))`

Find function usage help with

- `help(function)`
- `?function`



Function definition

Define the function

```
function_name <- function(arg1, arg2) {  
  do something  
}
```

Call the function

```
function_name(arg1 = data1, arg2 = data2)
```

Multiple dispatch (methods)

A function can have different behaviour depending on the input!

Test it!

```
plot(1:10)
```

```
plot(matrix(1:10, ncol=2))
```

```
plot(density(runif(100, 0,100)))
```

Multiple dispatch (methods)

New methods can be created by defining a function which looks for methods

```
function_name <- function(arg1, arg2) {  
  UseMethod("function_name")  
}
```

And defining a new method for objects of *class*

```
function_name.class <- function(arg1, arg2) { do something }
```

Extending R

R extensions

One of the best selling points of R are *user generated packages!*

- On 30.1.2019 there were 13 589 packages available in CRAN
 - + more hosted elsewhere, e.g. on GitHub

Packages are easily obtained from CRAN:

- *install.packages("package_name")*

Or (using another package, devtools) from GitHub:

- *devtools::install_github("GitHub_repository")*

CRAN

Packages in CRAN go through rigorous testing

- Package developers must ensure that the package works in different environments (mac, windows, linux...)
- Users can just use

CRAN tests are technical, *there is no scientific curation of package content!*

Extending R for your own use

- functions

1. write a function
2. save the function definition in an R script
3. source the script in a new R sessions
source("path_to/my_function.R")
4. load the libraries used in the function

Extending R for your own use - packages

Much more work than simply defining functions. However,

- Packages can be loaded regardless of your working directory!
- All required libraries will be installed on package installation!
- No need to preload libraries to use the functions!
- Can be distributed to other users!

Package example: hydrostreamer

Hydrostreamer is a package developed to downscale runoff products, and to estimate river discharge at an arbitrary river segment.

Currently distributed via GitHub:

<http://github.com/mkkallio/hydrostreamer>



(Minimal) Package structure

RStudio project internal files (not part of a package)

R code goes here

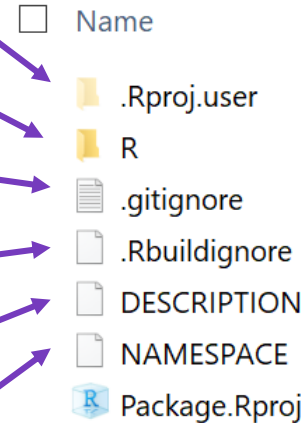
GitHub related file

List of files in the folder not part of the package

Package description

Package namespace (functions exposed to a user)

R project file (not part of a package)



DESCRIPTION

```
DESCRIPTION x  NAMESPACE x
1 Package: example_package
2 Title: A package with useful functions
3 Version: 0.0.0.9000
4 Authors@R: person("First", "Last", email = "first.last@example.com", role = c("aut", "cre"))
5 Description: This package does this and that!
6 Depends: R (>= 3.5.2)
7 Imports:
8   package1,
9   package2,
10  package3
11 Suggests:
12   package 4,
13   package 5
14 License: MIT
15 Encoding: UTF-8
16 LazyData: true
17 |
```

These will be installed with the package

If these packages are installed, the package can do more than just the core functionality

DESCRIPTION: hydrostreamer

Imports 15 other packages – required
for core functionality

Suggests 3 packages – not required,
but add functionality

Depends on R version $\geq 3.4.0$ –
cannot be installed for older versions

```
DESCRIPTION x  NAMESPAC x  HSflow.methods.R x  hydrostreamer.R x  News.md x  hy
1 Package: hydrostreamer
2 Type: Package
3 Title: A package for downscaling distributed runoff products on t
4   explicit river segments
5 Version: 0.3.2
6 Date: 2018-09-17
7 Author: Marko Kallio
8 Authors@R: person("Marko", "kallio", email = "marko@markokallio.f
9               role = c("aut", "cre"),
10              comment = c(ORCID = "0000-0002-6917-7790"))
11 Maintainer: The package maintainer <marko@markokallio.fi>
12 Description: hydrostreamer provides functions to downscale distri
13   into an explicitly represented river network. Downscaling is do
14   relationship between an areal unit of runoff and an overlaid ri
15   Value of the runoff unit is divided among intersecting river se
16   provides several methods for the assignment. Simple river routi
17   are also provided to estimate discharge at each segment.
18 Depends:
19   R (>= 3.4.0)
20 Encoding: UTF-8
21 Imports:
22   dplyr,
23   hydroGOF,
24   geosphere,
25   lwgeom,
26   methods,
27   sf,
28   tibble,
29   raster,
30   rgdal,
31   lubridate,
32   ggplot2,
33   quadprog,
34   ForecastComb,
35   forecast,
36   tidyr
37 Suggests:
38   knitr,
39   rmarkdown,
40   plotly
41 LazyData: true
42 License: MIT + file LICENSE
43 URL: https://mkkallio.github.io/hydrostreamer/
44 BugReports: https://github.com/mkkallio/hydrostreamer/issues
45 VignetteBuilder: knitr
46 RoxygenNote: 6.1.1
47
```

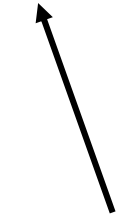
Functions in the package

```
DESCRIPTION x sum_two_vectors.R x NAMESPACE x
Source on Save Run Source
1 #' Sum two vectors separately
2 #'
3 #' This function sums two vectors separately.
4 #'
5 #' The function works so that the first vector and second vector are concatenated
6 #' and they are summed together.
7 #'
8 #' @param v1 first vector to be summed
9 #' @param v2 second vector to be summed
10 #'
11 #' @return The function sums two input vectors separately and returns their sums
12 #' as a vector.
13 #' @export Documentation
14 sum_two_vectors <- function(v1, v2) { Function name and arguments
15     s1 <- sum(v1)
16     s2 <- sum(v2)
17
18     output <- c(s1,s2) What the function does
19     return(output) What the function returns
20 }
21
```

Using functions from other packages

We must pass an explicit reference to the package!!!

example::sum_two_vectors(arg1, arg2)



Package name



Function name



Documentation using *roxygen2*

```
1 #' Sum two vectors separately ← Heading
2 #'
3 #' This function sums two vectors separately. ← Description
4 #'
5 #' The function works so that the first vector and second vector are concatenated
6 #' and they are summed together. ← Details
7 #'
8 #' @param v1 first vector to be summed ← Arguments
9 #' @param v2 second vector to be summed ← Arguments
10 #'
11 #' @return The function sums two input vectors separately and returns their sums
12 #' as a vector. ← What the function returns
13 #' @export
```

← Give the user direct access to the function

Documentation via *roxygen2*

Run `devtools::document()`

The example in the earlier slide gives the following output:

```
> library(example)
> ?sum_two_vectors
> |
```

```
sum_two_vectors {example}
```

R Documentation

Sum two vectors separately

Description

This function sums two vectors separately.

Usage

```
sum_two_vectors(v1, v2)
```

Arguments

v1 first vector to be summed
v2 second vector to be summed

Details

The function works so that the first vector and second vector are concatenated and they are summed together.

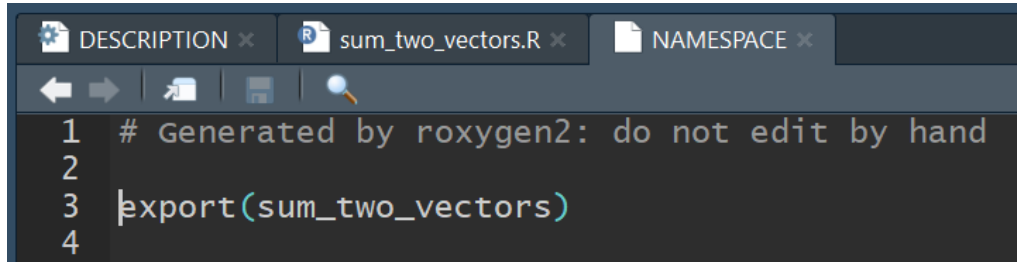
Value

The function sums two input vectors separately and returns their sums as a vector.

NAMESPACE

roxygen2 handles the namespace for you! It handles all those lines which precede function definition, and which start with #'

- Namespace gets populated based on the tag `@export` (among others)



```
DESCRIPTION x sum_two_vectors.R x NAMESPACE x
← → ↻ ⌂ 🔍
1 # Generated by roxygen2: do not edit by hand
2
3 export(sum_two_vectors)
4
```

NAMESPACE: hydrostreamer

Overall 60 different functions,
methods, or function imports from
other packages.

```
DESCRIPTION x  NAMESPACE x  HSflow.methods.R x  hydrostr
← →  ↻  📄  🔍
1  # Generated by roxygen2: do not edit by hand
2
3  s3method(HSwrite,HSflow)
4  s3method(HSwrite,HSgrid)
5  s3method(HSwrite,HSrunoff)
6  s3method(accumulate_runoff,instant)
7  s3method(accumulate_runoff,muskingum)
8  s3method(accumulate_runoff,simple)
9  s3method(combine_runoff,HSflow)
10 s3method(combine_runoff,HSgrid)
11 s3method(combine_runoff,HSrunoff)
12 s3method(combine_runoff,HSweights)
13 s3method(combine_runoff,list)
14 s3method(ensemble_summary,HSflow)
15 s3method(ensemble_summary,HSgrid)
16 s3method(ensemble_summary,HSrunoff)
17 s3method(ensemble_summary,HSweights)
18 s3method(ensemble_summary,list)
19 s3method(flow_gof,HSflow)
20 s3method(flow_gof,list)
21 s3method(summary,HSgrid)
22 export(HSwrite)
23 export(accumulate_runoff)
24 export(accumulate_runoff_instant)
25 export(accumulate_runoff_muskingum)
26 export(accumulate_runoff_simple)
27 export(add_HSgrid)
28 export(combine_runoff)
29 export(compute_HSweights)
```

Checking and installing package

R has built in program to inspect potential package.

In RStudio you can run it via menu *build/check package..*

Package can be installed via *build/install and restart..*

After installing, the package can be loaded with

library(example)

More information

Package development is complex. However, everything needed for package development in R can be found in the following (free) resources:

- R packages book - <http://r-pkgs.had.co.nz>
- Advanced R book - <http://adv-r.had.co.nz>
- ... and the proper manual: Writing R Extensions - <https://cran.r-project.org/doc/manuals/r-release/R-exts.html>