

# Why write a (mini) R package?

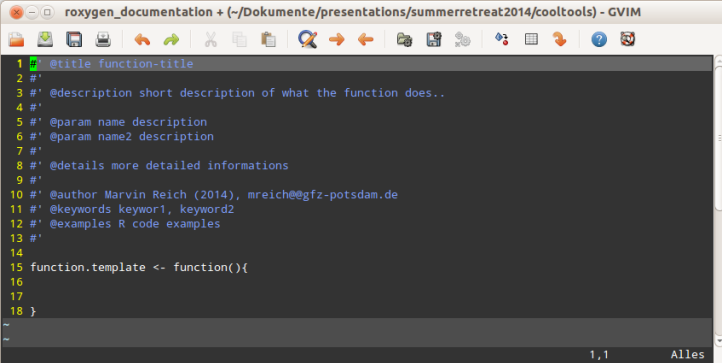
- all your messy code will be organized
- keep track of your functions used
- easily share your functions with others
- done analysis/processing in our section can easily be repeated
- help files facilitate understanding even after longer time periods

**thanks to two wonderful R packages, this is now way easy!!**

→ a mini R package in just 2 slides..

# roxygen2: documentation

- 1 `install.packages("roxygen2")`
- 2 `library(roxygen2)`
- 3 add roxygen2-formated comments **above** function declaration



The screenshot shows a GVIM editor window titled "roxygen\_documentation + (~/.Dokumente/presentations/summerretreat2014/cooltools) - GVIM". The editor displays the following R code:

```
1 #' @title function-title
2 #'
3 #' @description short description of what the function does..
4 #'
5 #' @param name description
6 #' @param name2 description
7 #'
8 #' @details more detailed informations
9 #'
10 #' @author Marvin Reich (2014), mreich@gfz-potsdam.de
11 #' @keywords keywor1, keyword2
12 #' @examples R code examples
13 #'
14
15 function.template <- function(){
16
17
18 }
```

The status bar at the bottom right indicates "1,1" and "Alles".

# devtools: package creation

- 1 `install.packages("devtools")`
- 2 `library(devtools)`
- 3 create folder (named after package) containing:
  - **man** folder
  - **R** folder
  - **DESCRIPTION** file
- 4 copy roxygen-commented function file(s) into subfolder "R"
- 5 fill mandatory fields in DESCRIPTION file:
  - Package: package name
  - Version: any number
- 6 `setwd("package folder directory")` and run: **document()**
- 7 `setwd("...")` and run: **install("package name")**
- 8 optional: **build("package name")** builds a distributable package file