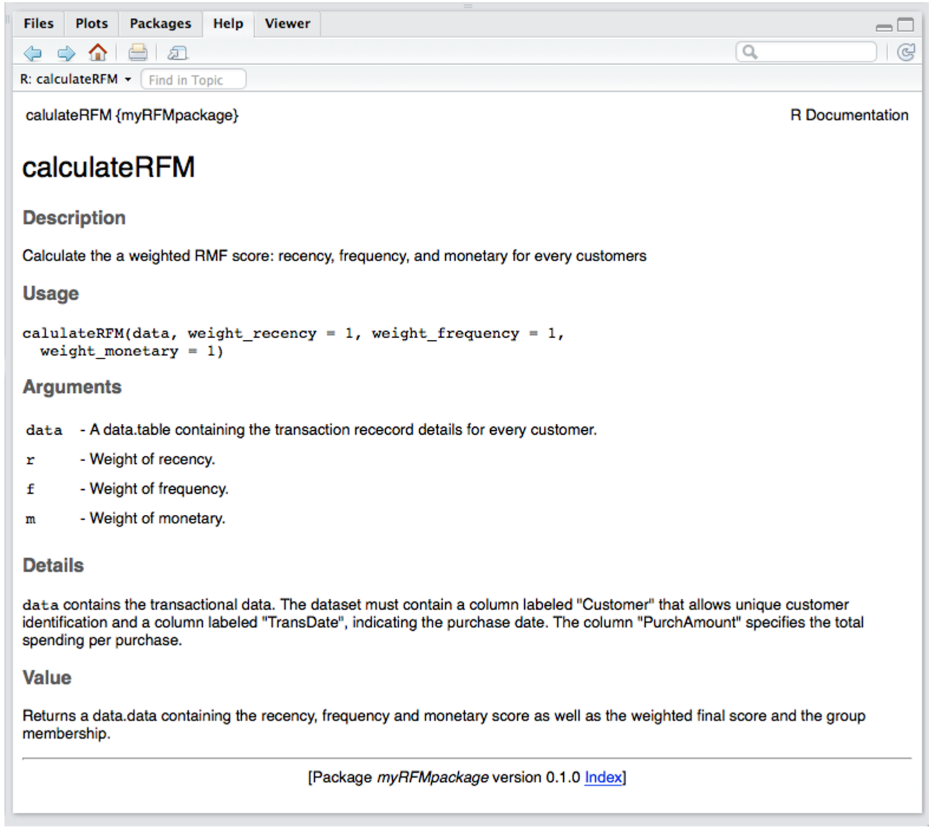
Keep calm and read the manual!

Document your package!

Provide a detailed documentation for your package

- Enable others to really use your package
- Save time when using the package later on
- In-source documentation integrates easily with your code and makes it easy to change/generate documentation



calulateRFM {myRFMpackage} R Documentation

calculateRFM

Description

Calculate the a weighted RMF score: recency, frequency, and monetary for every customers

Usage

```
calulateRFM(data, weight_recency = 1, weight_frequency = 1,
  weight_monetary = 1)
```

Arguments

`data`	- A data.table containing the transaction rececord details for every customer.
`r`	- Weight of recency.
`f`	- Weight of frequency.
`m`	- Weight of monetary.

Details

`data` contains the transactional data. The dataset must contain a column labeled "Customer" that allows unique customer identification and a column labeled "TransDate", indicating the purchase date. The column "PurchAmount" specifies the total spending per purchase.

Value

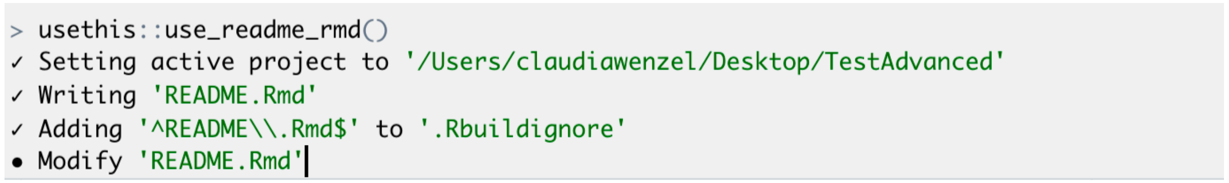
Returns a data.data containing the recency, frequency and monetary score as well as the weighted final score and the group membership.

[Package *myRFMpackage* version 0.1.0 Index]

Document your package for other users
Add a README file (1/2)

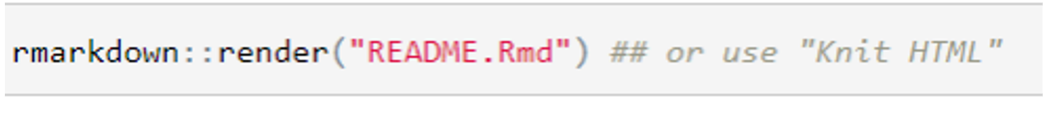
Adding a README file to your package distribution is useful when sharing your package on Github and will be shown as Description of your package.

Use the `use_readme_rmd()` function to create a README.Rmd file:



```
> usethis::use_readme_rmd()
✓ Setting active project to '/Users/claudiawenzel/Desktop/TestAdvanced'
✓ Writing 'README.Rmd'
✓ Adding '^README\\.Rmd$' to '.Rbuildignore'
● Modify 'README.Rmd'
```

Make sure to describe the basic functionality of your package and give an overview over all modules in your package. Then make sure you knit it



```
rmarkdown::render("README.Rmd") ## or use "Knit HTML"
```

Document your package for other users
Add a README file (2/2)

The goal of the README file is to answer the following questions about your package:

- Why should I use it?
- How do I use it?
- How do I get it?

https://r-pkgs.org/release.html

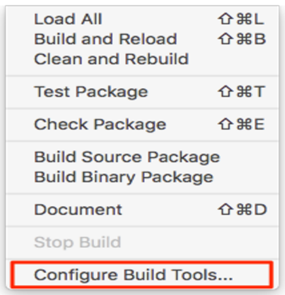
Use a package to automate some processes of the documentation work: Roxygen2

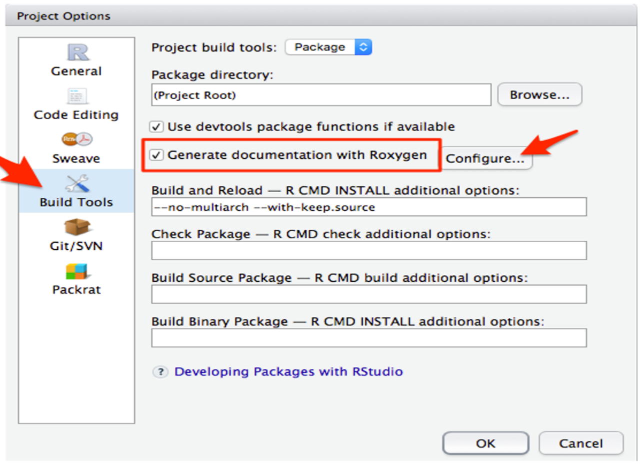
Roxygen2 is an R package that allows:

- Easy in-source documentation
- Automatic generation of help files

```
install.packages("roxygen2")
library(roxygen2)
```

Tell RStudio to use Roxygen2: Go to Build -> Configure Build Tools...





Project Options

Project build tools: Package

Package directory: (Project Root)

[✓] Use devtools package functions if available

[✓] Generate documentation with Roxygen

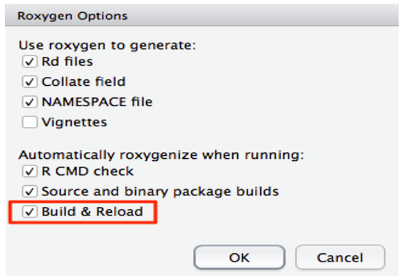
Build and Reload — R CMD INSTALL additional options: --no-multiarch --with-keep.source

Check Package — R CMD check additional options:

Build Source Package — R CMD build additional options:

Build Binary Package — R CMD INSTALL additional options:

Developing Packages with RStudio



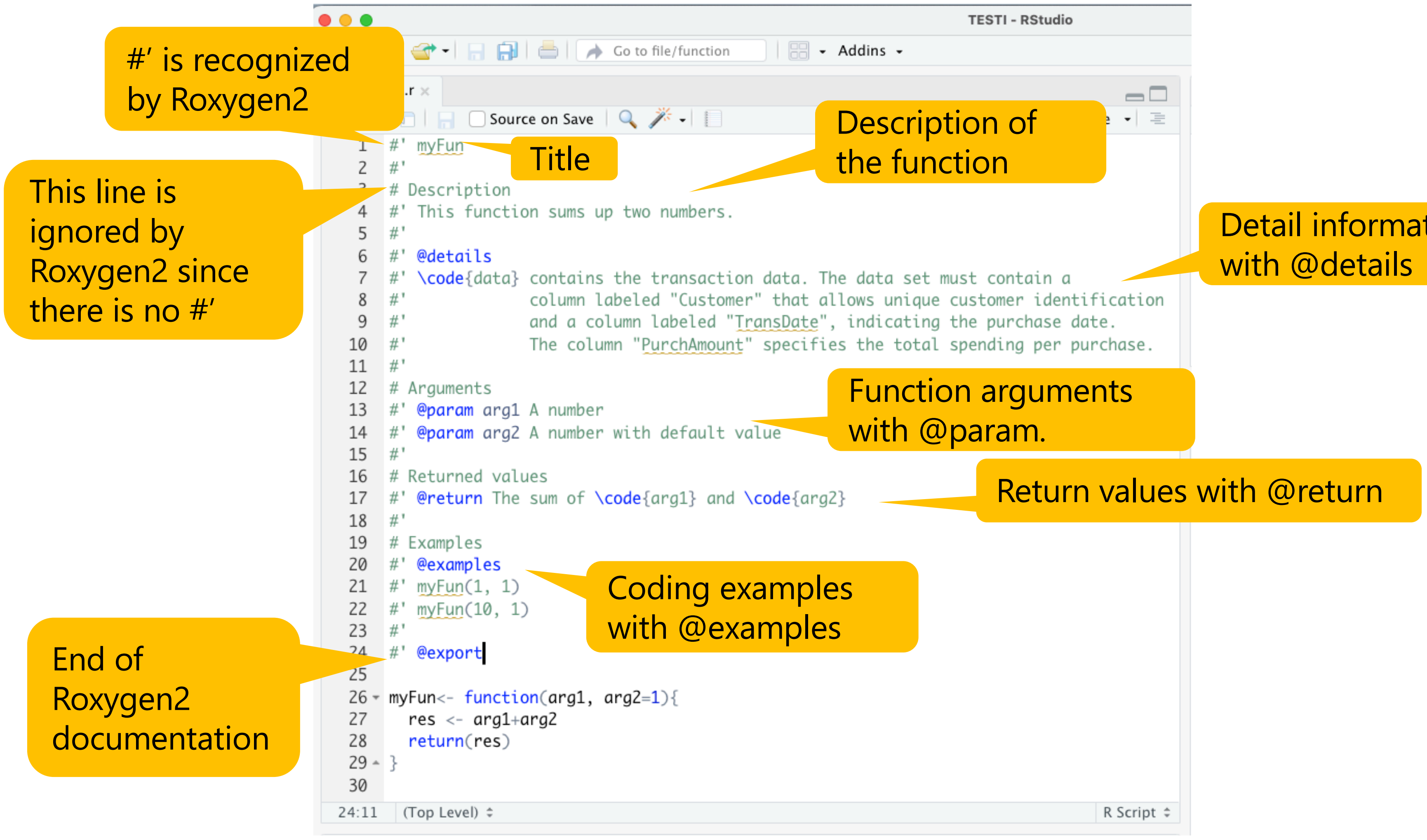
Roxygen Options

Use roxygen to generate:
- [✓] Rd files
- [✓] Collate field
- [✓] NAMESPACE file
- [ ] Vignettes

Automatically roxygenize when running:
- [✓] R CMD check
- [✓] Source and binary package builds
- [✓] Build & Reload

https://support.rstudio.com/hc/en-us/articles/200532317-Writing-Package-Documentation

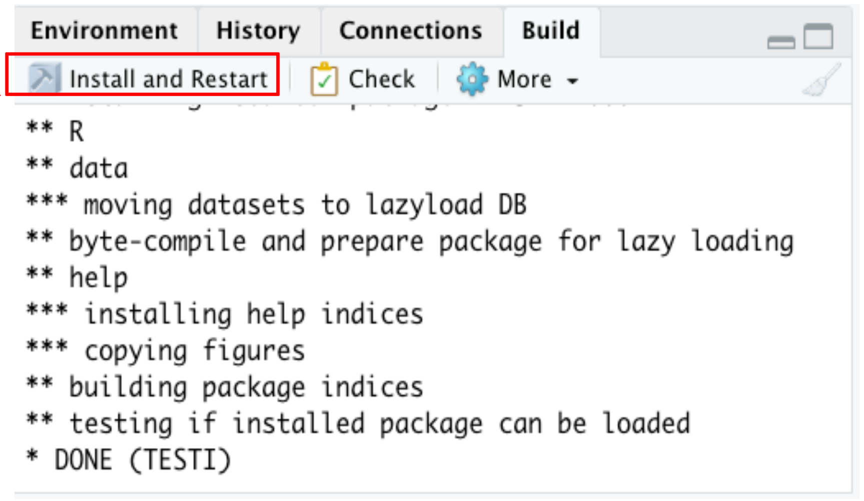
Document your package for other users - Document your functions


TESTI - RStudio
Go to file/function
Addins
Source on Save
#' is recognized by Roxygen2
This line is ignored by Roxygen2 since there is no #'
Title
Description of the function
Detail information with @details
Function arguments with @param.
Return values with @return
Coding examples with @examples
End of Roxygen2 documentation
24:11
(Top Level)
R Script

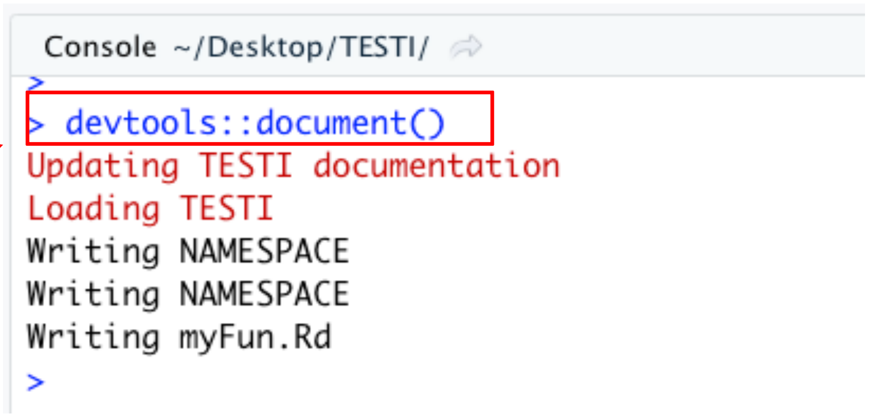
```r
#' myFun
#'
# Description
#' This function sums up two numbers.
#'
#' @details
#' \code{data} contains the transaction data. The data set must contain a
#'              column labeled "Customer" that allows unique customer identification
#'              and a column labeled "TransDate", indicating the purchase date.
#'              The column "PurchAmount" specifies the total spending per purchase.
#'
# Arguments
#' @param arg1 A number
#' @param arg2 A number with default value
#'
# Returned values
#' @return The sum of \code{arg1} and \code{arg2}
#'
# Examples
#' @examples
#' myFun(1, 1)
#' myFun(10, 1)
#'
#' @export

myFun<- function(arg1, arg2=1){
  res <- arg1+arg2
  return(res)
}
```

https://cran.r-project.org/web/packages/roxygen2/vignettes/roxygen2.html

Document your package for other users - Re-Build your documented package



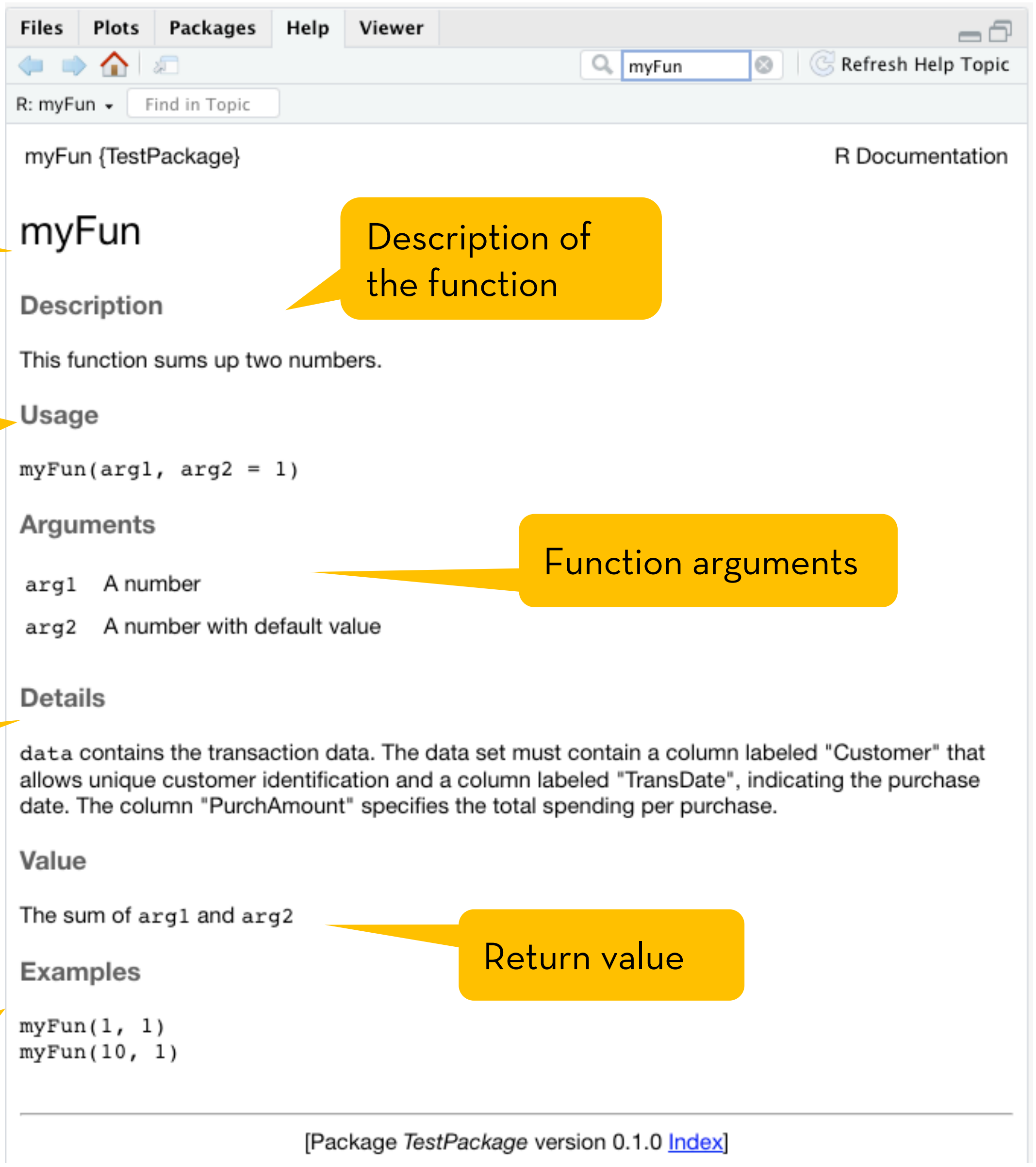
```
** R
** data
*** moving datasets to lazyload DB
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
*** copying figures
** building package indices
** testing if installed package can be loaded
* DONE (TESTI)
```



```
>
> devtools::document()
Updating TESTI documentation
Loading TESTI
Writing NAMESPACE
Writing NAMESPACE
Writing myFun.Rd
>
```

The help files are automatically generated.

… and get a well-documented package



myFun {TestPackage} R Documentation

myFun

Description

This function sums up two numbers.

Usage

```
myFun(arg1, arg2 = 1)
```

Arguments

`arg1`	A number
`arg2`	A number with default value

Details

`data` contains the transaction data. The data set must contain a column labeled "Customer" that allows unique customer identification and a column labeled "TransDate", indicating the purchase date. The column "PurchAmount" specifies the total spending per purchase.

Value

The sum of `arg1` and `arg2`

Examples

```
myFun(1, 1)
myFun(10, 1)
```

[Package *TestPackage* version 0.1.0 Index]

Now it’s your turn!