



BIODIVERSITY
BUILDING
BLOCKS FOR
POLICY

How to Package Your Functions: From Standalone to R Packages

Damiano Oldoni

Research Institute for Nature and Forest (INBO)



Funded by
the European Union

RESEARCH INSTITUTE
NATURE AND FOREST

4th B-Cubed Workshop/2024-11-07

Workshop material



Workshop material

Code and slides



Biodiversity Building Blocks for policy



Funded by
the European Union

Code and slides



All material is on GitHub:

github.com/b-cubed-eu/b-cubed-workshops/tree/main/workshops/04



In particular: [slides](#) (pdf)



Introduction







Introduction

Why packaging functions in R?



Why packaging your functions?

-  Code organization
-  Documentation
-  Easy to share
-  Quality control (testing)






Introduction

What is an R Package?




The cooking metaphor

-  R is our kitchen
-  R console is our food processor
-  Functions are recipes. They tell you how to:
 - Take inputs (ingredients)
 - Process them
 - Return an output (food)



The cooking metaphor

-  Define the function `make_bread()`
-  Define the recipe to make bread in the food processor settings

```
make_bread <- function(grains, yeast, water, salt) {  
  # Code to generate `bread`.  
  # The code here can be easy (easy bread recipes do exist)  
  # or quite complex (complex bread recipes do exist too)  
  bread <- grains + yeast + water + salt  
  return(bread)  
}
```

The cooking metaphor

 Add ingredients to the food processor, select your own recipe, press “Play

 Pass inputs to the function, call it, press Enter

```
# Prepare ingredients
```

```
g <- 20 # Prepare amount of grains)
```

```
y <- 1 # Prepare amount of yeast)
```

```
w <- 2 # Prepare amount of water)
```

```
s <- 3 # Prepare amount of salt)
```

```
# Add ingredients in food processor = Pass values to arguments of the function
```





```
bread <- make_bread(grains = g, yeast = y, water = w, salt = s)
```

```
# Press `Enter`
```

```
bread
```



The cooking metaphor

-  R is our kitchen
-  R console is our food processor
-  Functions are recipes. They tell you how to:
 - Take inputs (ingredients)
 - Process them
 - Return an output (food)
-  R package is well-organised bundle of recipes, a cookbook. It:
 - Bundles functions (recipes)
 - Comes with metadata (book details, title, authors, licence, ...)






The cooking metaphor

 From our notes about culinary art of grandma to a cookbook



What is an R package?

-  Fundamental unit of code in R
-  Standardized way for organizing, documenting, testing, distributing functions
-  Enhance reproducibility and reusability









Introduction

Overview of necessary software tools



Software

-  Programming language: [R](#)
-  IDE: [RStudio](#)
-  Software development package: [devtools](#) and [usethis](#)
-  Testing: [testthat](#)
-  Documentation: [Roxygen2](#) and [pkgdown](#)
-  Version control: [git](#), [GitHub](#)



R Package Essentials

The background of the slide is a close-up photograph of a colony of bees. A large, dense cluster of bees is in the lower center, with many more bees scattered across the green, textured surface. The image is semi-transparent, allowing the white text to stand out clearly.





R Package Essentials

Structure of an R package



Structure of an R package





Essential components

-  **DESCRIPTION** : file (metadata)
-  **R/** : directory (source code)
-  **man/** : directory (documentation)
-  **NAMESPACE**: file (export/import declarations)



Structure of an R package

Optional components

-  **tests/**: directory (code for testing)
-  **vignettes/**: directory (long-form documentation)
-  **data/**: directory (datasets)
-  **inst/**: directory (additional resources)













R Package Essentials

Key components



R Package essentials

DESCRIPTION file

-  **Package:** package name
-  **Title:** Very Short Description in Title Case
-  **Version:** Use x.y.z or x.y.z.9000. See [Lifecycle](#) from R Packages (2e)
-  **Authors@R:** maintainer (author and creator) and any other author/contributor
-  **Description:** Long description. One or more paragraphs.
-  **License:** MIT + file LICENSE
-  **Encoding:** UTF-8
-  **Roxygen:** instructions for Roxygen about documentation
-  **RoxygenNote:** Roxygen version, e.g. 7.3.2
-  **Imports:** dependencies (packages it depends on)

Create Your First Package






Create Your First Package

Naming your package



Naming your package

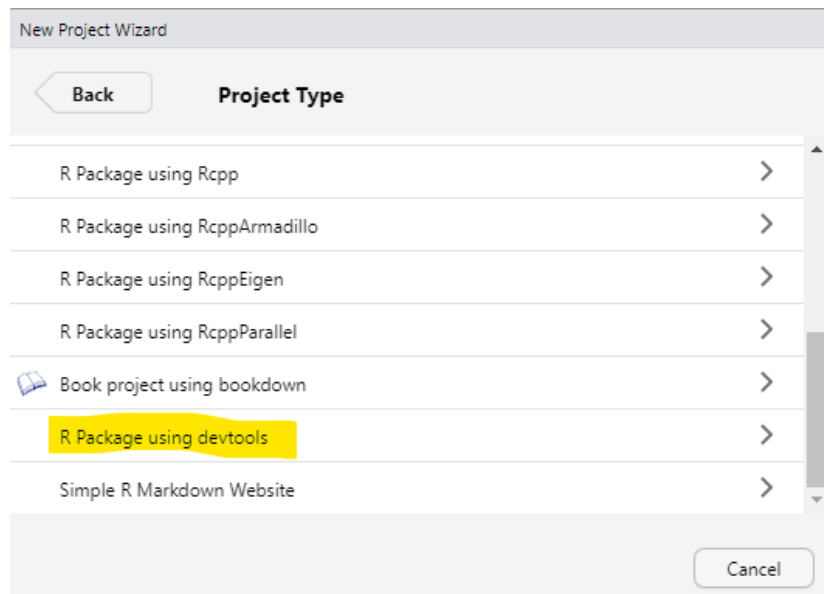
-  Check [“Naming your package”](#) from the B-Cubed software dev guide
-  Is your package name available, informative and not offensive?
`pak::pkg_name_check("mycoolpkgname")`
-  Most generic tool: `available::available("mycoolpkgname")`

Create Your First Package

Setting up a new package project in RStudio



Setting up a new package project in RStudio



Setting up a new package project in Rstudio

 Setup everything we need in a single line, isn't [usethis](#) amazing?

```
usethis::create_package("path/to/packageName")
```

 Example:

```
usethis::create_package("./workshops/04/bRead")
```

 Redirected to new Rstudio session linked to bRead.Rproj

Create Your First Package

Update the DESCRIPTION file






Update the DESCRIPTION file

```
Package: bRead
Title: What the Package Does (One Line, Title Case)
Version: 0.0.0.9000
Authors@R:
  person("First", "Last", , "first.last@example.com", role = c("aut", "cre"),
    comment = c(ORCID = "YOUR-ORCID-ID"))
Description: what the package does (one paragraph).
License: `use_mit_license()`, `use_gpl3_license()` or friends to pick a
  license
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.3.2
```



Setting up a new package project in Rstudio

-  Update **License**: set MIT licence `usethis::use_mit_license()`
-  [choosealicense](#): overview software dedicated licences
-  Update **Title**, **Authors@R** and **Description** manually

Update the DESCRIPTION file

```
Package: bRead
Title: Make Tasty Doughs with R
Version: 0.0.0.9000
Authors@R:
  c(
    person("Damiano", "Oldoni", , "damiano.oldoni@inbo.be", role = c("aut", "cre"),
      comment = c(ORCID = "0000-0003-3445-7562")),
    person("Emma", "Cartuyvels", , "emma.cartuyvels@inbo.be", role = "ctb",
      comment = c(ORCID = "0000-0001-7856-6360")))
Description: A elegant way to create doughs using R.
  Typically doughs you can create are bread and focaccia. But who knows,
  maybe in the future we will add more doughs.
License: MIT + file LICENSE
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.3.2
```










Create Your First Package

Moving functions into the package structure



Moving functions into the package structure






-  We have a package, nice! But empty
-  Add two functions: `make_bread()` and `make_focaccia()`
-  Take code from [20241107_basic_functions.R](#)
-  MUST: R files in **R/** folder
-  Best practice: one file = one function
-  Best practice: file name = function name
-  Two files: **make_bread.R** and **make_focaccia.R**

Create Your First Package

Writing documentation with roxygen2



Writing documentation with roxygen2








-  Function documentation = **.Rd** files in the **man/** folder
-  Very, very old school: write .Rd files yourself. Please, don't 🙏
-  Nowadays, documentation is written in the form of Roxygen2 documentation
-  Put the mouse in the function you want to document
-  In Rstudio, click on **Code** → **Insert Roxygen skeleton** (Alt + Ctrl + Shift + R)

Writing documentation with roxygen2

```
#' Title
#'  
#' @param grains  
#' @param yeast  
#' @param water  
#' @param salt  
#'  
#' @return  
#' @export  
#'  
#' @examples  
make_bread <- function(grains, yeast, water, salt) {  
  # Code to generate `bread`.  
  # The code here can be easy (easy bread recipes do exist)  
  # or quite complex (complex bread recipes do exist too)  
  bread <- grains + yeast + water + salt  
  return(bread)  
}
```



Writing documentation with roxygen2

-  MUST: set title (active form)
-  SHOULD: add description (**@description** or add text under title, separated by one empty line)
-  MUST: document arguments (**@param**)
-  MUST: document output (**@return**)
-  SHOULD: add at least an example (**@example**)
-  Allow users of the package to use the function (**@export**)
-  Documented functions: [20241107_documented_functions.R](#)

Writing documentation with roxygen2

```
#' Make bread
#'
#' @param grains Amount of grains (numeric).
#' @param yeast Amount of yeast (numeric).
#' @param water Amount of water (numeric).
#' @param salt Amount of salt (numeric).
#'
#' @return Amount of bread.
#' @export
#'
#' @examples
#' make_bread(
#'   grains = 1,
#'   yeast = 2,
#'   water = 3,
#'   salt = 4
#' )
make_bread <- function(grains, yeast, water, salt) {
```







Create Your First Package

README



README

-  The **README** is the welcome sign for users
-  Use `usethis::use_readme_md()` to create a **README.md** file
-  Check "[The README file](#)" from the B-Cubed software dev guide
-  Use [20241107_readme_content.md](#)





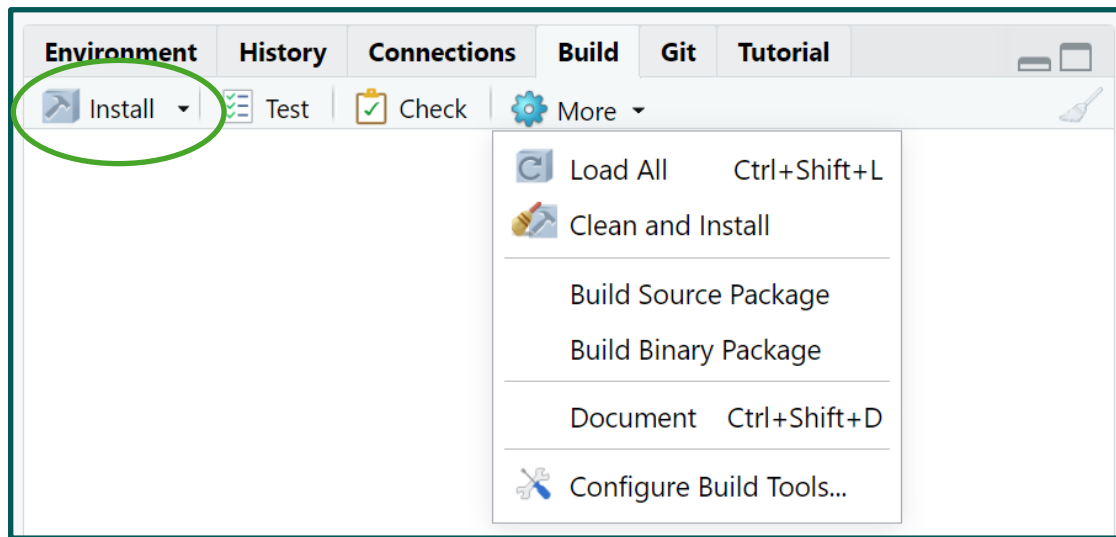
Create Your First Package

Using devtools functions



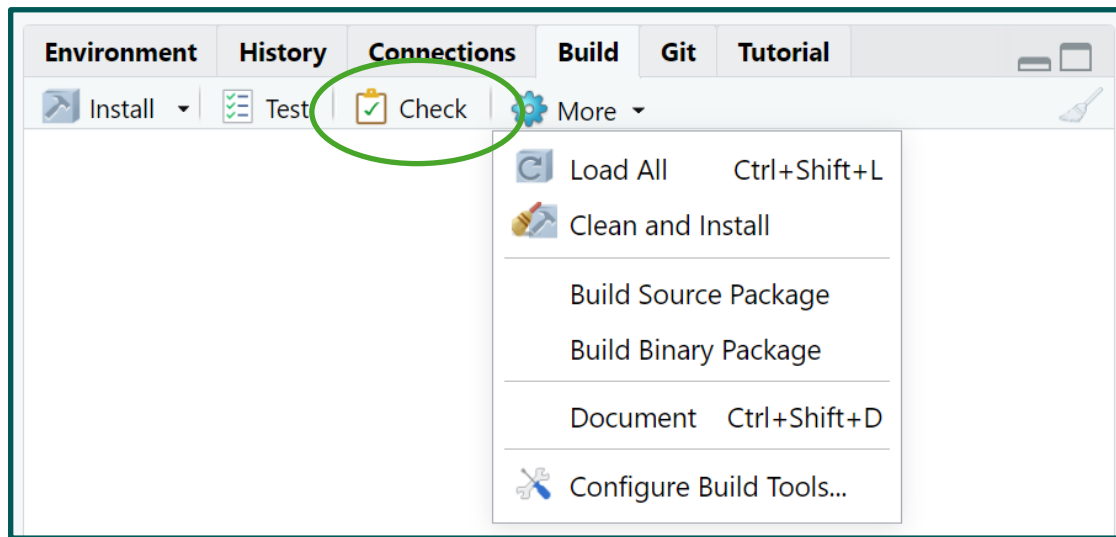
Using devtools functions

-  **Install** icon: install + restart R session + load package (**library(bRead)**)
-  Equivalent of **devtools::install()**





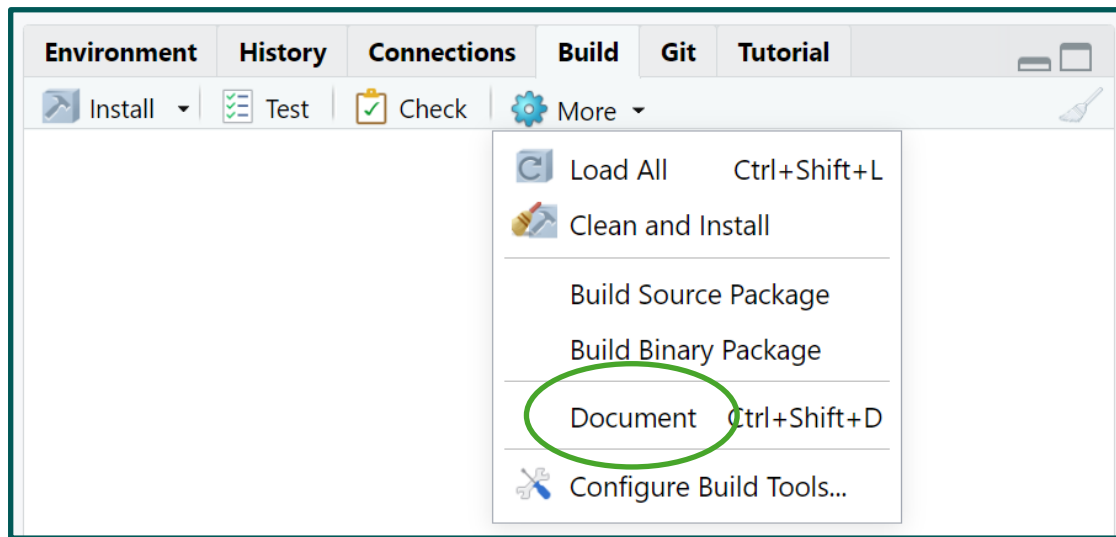
Using devtools functions

-  **Check** icon: check package (are metadata correct? Can be installed? Tests ok?)
-  Equivalent of `devtools::check()`



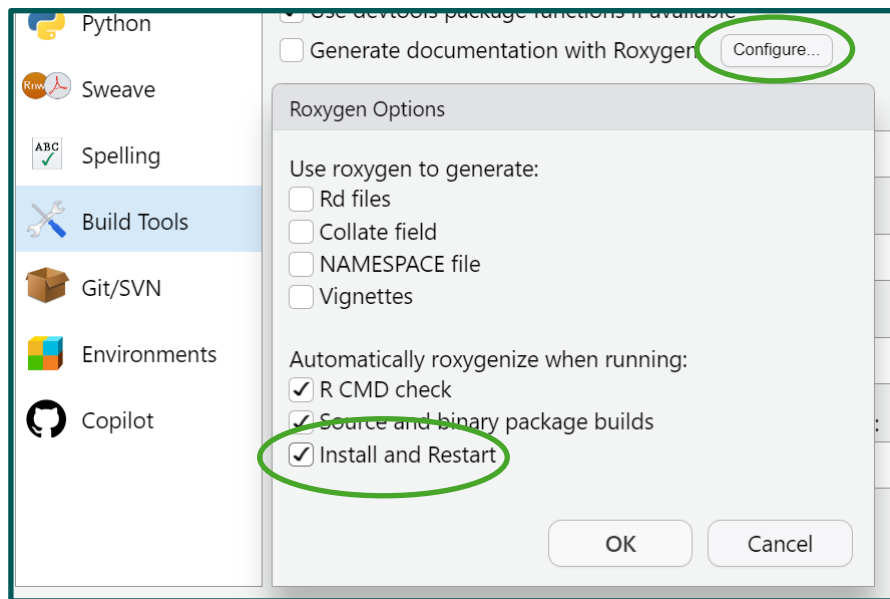
Using devtools functions

-  **Document** (Ctrl + Shift + D): create documentation = **.Rd** files in **man/** folder
-  Equivalent of `devtools::document()`



Using devtools functions

- More → Configure Build Tools... → Configure (next “Generate documentation with Roxygen”)
- Check **Install and Restart** option ☒ under “Automatically roxygenize when running:”





Create Your First Package

Dependencies



Dependencies

-  Often our packages depend on other packages
-  Somewhere in our functions we use functions of other packages



Dependencies



Example: [20241107_functions_with_assertions.R](#)



Add some assertions in `make_bread()` and `make_focaccia()`

```
make_bread <- function(grains, yeast, water, salt) {  
  # Check that inputs are numeric scalars  
  assertthat::assert_that(assertthat::is.number(grains))  
  assertthat::assert_that(assertthat::is.number(yeast))  
  assertthat::assert_that(assertthat::is.number(water))  
  assertthat::assert_that(assertthat::is.number(salt))  
  # Make bread  
  bread <- grains + yeast + water + salt  
  return(bread)  
}
```



Dependencies



Add Imports: field in DESCRIPTION

```
Package: bRead
Title: Make Tasty Doughs With R
Version: 0.0.0.9000
Authors@R:
  c(
    person("Damiano", "oldoni", , "damiano.oldoni@inbo.be", role = c("aut", "cre"),
      comment = c(ORCID = "0000-0003-3445-7562")),
    person("Emma", "Cartuyvels", , "emma.cartuyvels@inbo.be", role = "ctb",
      comment = c(ORCID = "0000-0001-7856-6360")))
Description: A elegant way to create doughs using R.
  Typically doughs you can create are bread and focaccia. But who knows,
  maybe in the future we will add more doughs.
License: MIT + file LICENSE
Encoding: UTF-8
Roxygen: list(markdown = TRUE)
RoxygenNote: 7.3.2
Imports:
  assertthat
```










Create Your First Package

Addressing common errors and warnings



Addressing common errors and warnings

What happens if:

-  Syntax **DESCRIPTION** not correct
-  Forgotten to add **dependency** in **DESCRIPTION**
-  Function without documentation
-  **README** missing
-  Forgotten to document an argument in **@param**
-  **@export** not present
-  Function deprecated

Testing







Testing

Why writing tests?



Why writing tests?

Tests are important to check if your functions work as expected. Why testing?

-  **Fewer bugs:** you can catch bugs before they become a bug 🐛
-  **Better code structure:** you need to think about how to test your code, which often leads to better code structure
-  **Call to action:** you can write tests for your functions before you write the functions themselves. This is called test-driven development (TDD)
-  **Robust code:** if you change something in your code, you can run the tests to see if everything still works as expected.

Source: chapter "Why is formal testing worth the trouble?", from R Packages (2e), Hadley Wickham and Jennifer Bryan.










Testing


Testing tools



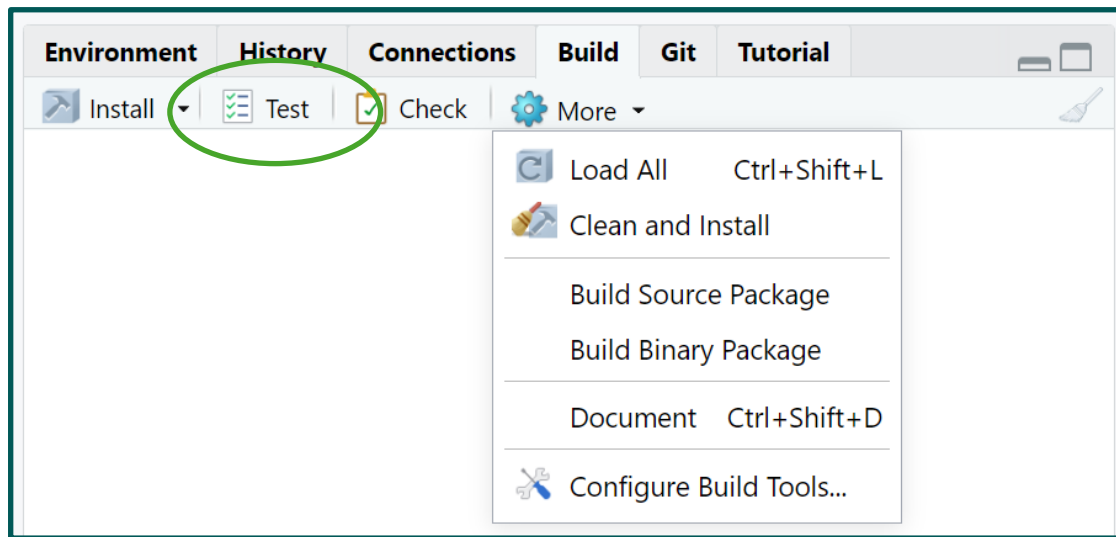
Testing tools

-  Use [testthat](#) R package
-  Run `usethis::use_testthat()`: set up the test infrastructure for our package
-  Check ["Using testthat in practise"](#) from the B-Cubed software dev guide
-  Add R file with tests using `usethis::use_test()`
-  Tests for **R/xyz.R** live in **tests/testthat/test-xyz.R**
-  Example tests: [20241107_tests_for_our_functions.R](#)
-  Run tests with `devtools::test()`

Testing tools

 **Test** (Ctrl + Shift + T): run tests for package

 Equivalent of `devtools::test()`



Version control








A large number of bees are shown in various states of flight and swarming over a light-colored wooden surface. The bees are yellow and black striped, with some showing more detail than others due to focus. The background is a soft, out-of-focus green, suggesting a natural outdoor setting. The text 'Version control' is overlaid in a clean, white, sans-serif font on the left side of the image.

Version control


Create new GitHub repository



Create new GitHub repository

-  Check [“Create a repository”](#) from the B-Cubed software dev guide
-  The **repository name** SHOULD be **lowercase**, dash-separated and short.
-  The **description** SHOULD be a **one-sentence title** (no period at the end)
-  The **visibility** MUST be set to **public**.
-  Check “Add a README file”.
-  You MUST select **.gitignore** template for **R**
-  You MUST select a **licence** and you MUST set it to **MIT License**.


Create new GitHub repository


Owner *  damianooldoni

Repository name * bread
bread is available.

Great repository names are short and memorable. Need inspiration? How about [cuddly-waddle](#) ?

Description (optional)
R package to learn how to work with packages during B-Cubed workshop of Nov 7, 2024

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:


☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
.gitignore template: R
Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: MIT License
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)




Create new GitHub repository

 **bread** Public




Pin Unwatch 1 Fork 0 Star 0

main 1 Branch Tags

t Add file <> Code

 **damianooldoni** Initial commit

08ea6c4 · now 1 Commit

 .gitignore	Initial commit	now
 LICENSE	Initial commit	now
 README.md	Initial commit	now

README MIT license

bread

R package to learn how to work with packages during B-Cubed workshop of Nov 7, 2024

About

R package to learn how to work with packages during B-Cubed workshop of Nov 7, 2024

Readme Activity 0 stars 1 watching 0 forks

Releases



No releases published

[Create a new release](#)


Packages



Add a CITATION.cff file

-  Allow users to know how to cite your package
-  See steps in [“Add a CITATION.cff file”](#) from the B-Cubed software dev guide

Adding a CITATION.cff file helps users to easily cite your software from the repository overview. [Learn more about CITATION files.](#) Insert example

 **bread** / in main Cancel changes Commit changes...

Edit Preview

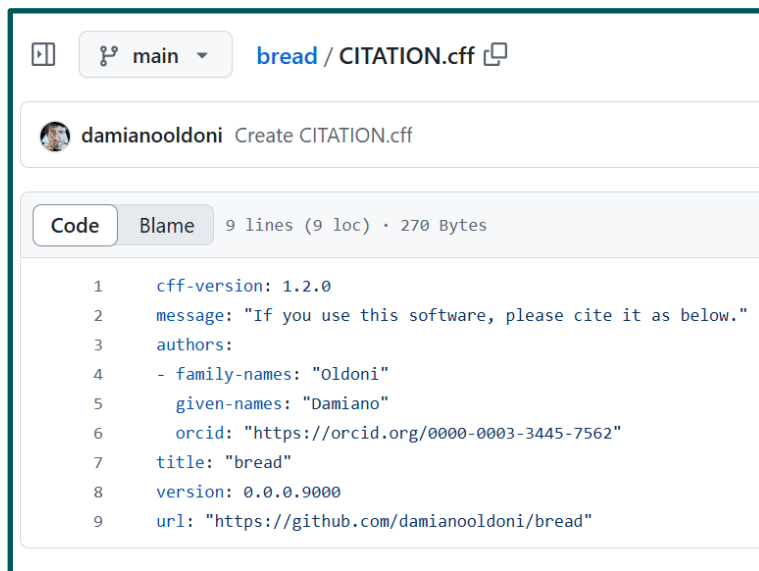
Spaces 2 No wrap

1

Enter file contents here

Add a CITATION.cff file

- 1 Include the name and [ORCID](#) of the maintainers
- 2 Remove the lines **doi** and **date-release** and commit
- 3 Update it with `cffr::cff_write()`



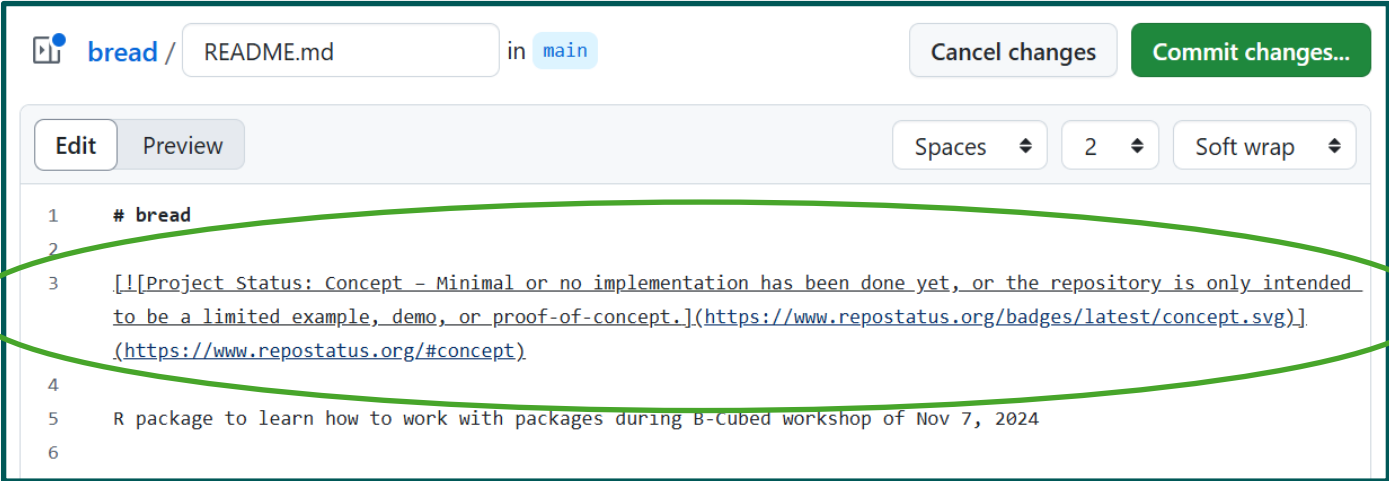
The screenshot shows a GitHub repository interface for a file named `CITATION.cff` in the `bread` repository, on the `main` branch. The file was created by `damianooldoni`. The file content is as follows:

```
1  cff-version: 1.2.0
2  message: "If you use this software, please cite it as below."
3  authors:
4    - family-names: "Oldoni"
5      given-names: "Damiano"
6      orcid: "https://orcid.org/0000-0003-3445-7562"
7  title: "bread"
8  version: 0.0.0.9000
9  url: "https://github.com/damianooldoni/bread"
```



Add a repository status badge

- See “[Badges](#)” from the B-Cubed software dev guide
- Possible statuses: [repostatus.org](https://www.repostatus.org)
- Example: [bread](#) repo is a demo = concept status

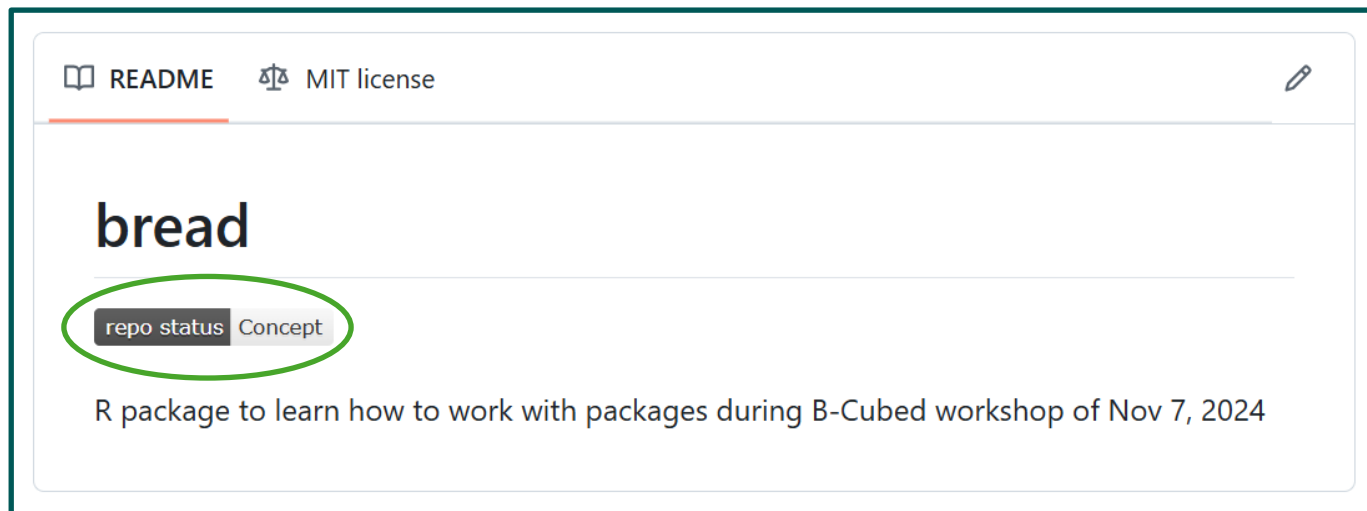


```
1 # bread
2
3 [!Project Status: Concept - Minimal or no implementation has been done yet, or the repository is only intended
4 to be a limited example, demo, or proof-of-concept.](https://www.repostatus.org/badges/latest/concept.svg)
   (https://www.repostatus.org/#concept)
5
6 R package to learn how to work with packages during B-Cubed workshop of Nov 7, 2024
```



Add a repository status badge

- See [“Badges”](#) from the B-Cubed software dev guide
- Possible statuses: repostatus.org
- Example: [bread](#) repo is a demo = concept status




Report issues

- See [“Report issues”](#) from the B-Cubed software dev guide
- Issues can be used to report and discuss a bug, idea or task
- Add issue template: `usethis::use_tidy_issue_template()`

Issue: Bug report or feature request

Describe a bug you've seen or make a case for a new feature. If this doesn't look right, [choose a different type](#).

 **Add a title**

Add a description

Write

Preview

Please briefly describe your problem and what output you expect. If you have a question, please don't use this form. Instead, ask on <https://stackoverflow.com/> or <https://community.rstudio.com/>.

Please include a minimal reproducible example (AKA a reprex). If you've never heard of a [reprex](<http://reprex.tidyverse.org/>) before, start by reading <https://www.tidyverse.org/help/#reprex>.

For more advice on how to write a great issue, see <https://code-review.tidyverse.org/issues/>.

Brief description of the problem

```
""r
# insert reprex here
""
```

☒ Markdown is supported

☐ Paste, drop, or click to add files

Submit new issue

Resources, Best Practices, Q&A









The background of the slide features a close-up photograph of a large cluster of bees on a light-colored, textured surface, possibly a piece of fabric or paper. The bees are in various states of activity, with some in sharp focus and others blurred. The entire image is overlaid with a semi-transparent teal or green tint, which serves as a backdrop for the white text.

Resources, Best Practices, Q&A

Useful links



Useful links

-  The [B-Cubed software development guide](#)
-  The main resource for package development: the [R packages book, 2nd edition](#)
-  The [rOpenSci Packages: Development, Maintenance, and Peer Review](#) guide
-  The [tidyverse style guide](#): B-Cubed's official style guide
-  Packages: [Roxygen2](#) and [pkgdown](#) (documentation), [devtools](#) and [usethis](#) (software development), [testthat](#) (tests)
-  The [checklist](#) package: the "usethis" on steroids 💪
-  [Naming Things](#): Blogpost about naming R packages
-  [choosealicense](#): overview software dedicated licences

Dedicated support channels – updates



[rOpenSci Slack](#)



For B-Cubed people: Slack Channel **#wp3-software-helpdesk**



Subscribe to the monthly [rOpenSci Newsletter](#)



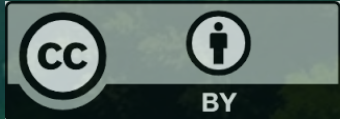
Thank you!

Damiano Oldoni

damiano.oldoni@inbo.be

Open science lab for biodiversity ([oscibio](#))

Research Institute Nature and Forest ([INBO](#))



b-cubed.eu



[@BCubedProject](#)



[B-Cubed Project](#)

Photo by Viridiflavus - Own work, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=4956453>



Biodiversity Building Blocks for policy

This project receives funding from the European Union's Horizon Europe Research and Innovation Programme (ID No 101059592). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the EU nor the EC can be held responsible for them.



**Funded by
the European Union**