

The "Rmd first" method: Documenting is taking care of your future

*Action devlog :
Écoinformatique FAIR par la pratique*



Sébastien Rochette

This presentation on Github: [statnmap/course_material](https://statnmap.com/course_material)

Sébastien

Data Scientist, R expert, R trainer.

- <https://statnmap.com>
- <https://rtask.thinkr.fr>
- <https://github.com/ThinkR-open>
- https://twitter.com/thinkr_fr



**Vincent
Guyader**

Codeur Fou,
formateur et expert
logiciel R



**Diane
Beldame**

Dompteuse de
~~dragons~~ données,
formatrice logiciel R



**Colin
Fay**

Data scientist
et R hacker



**Sébastien
Rochette**

Modélisateur,
Formateur R, Joueur
de cartographies

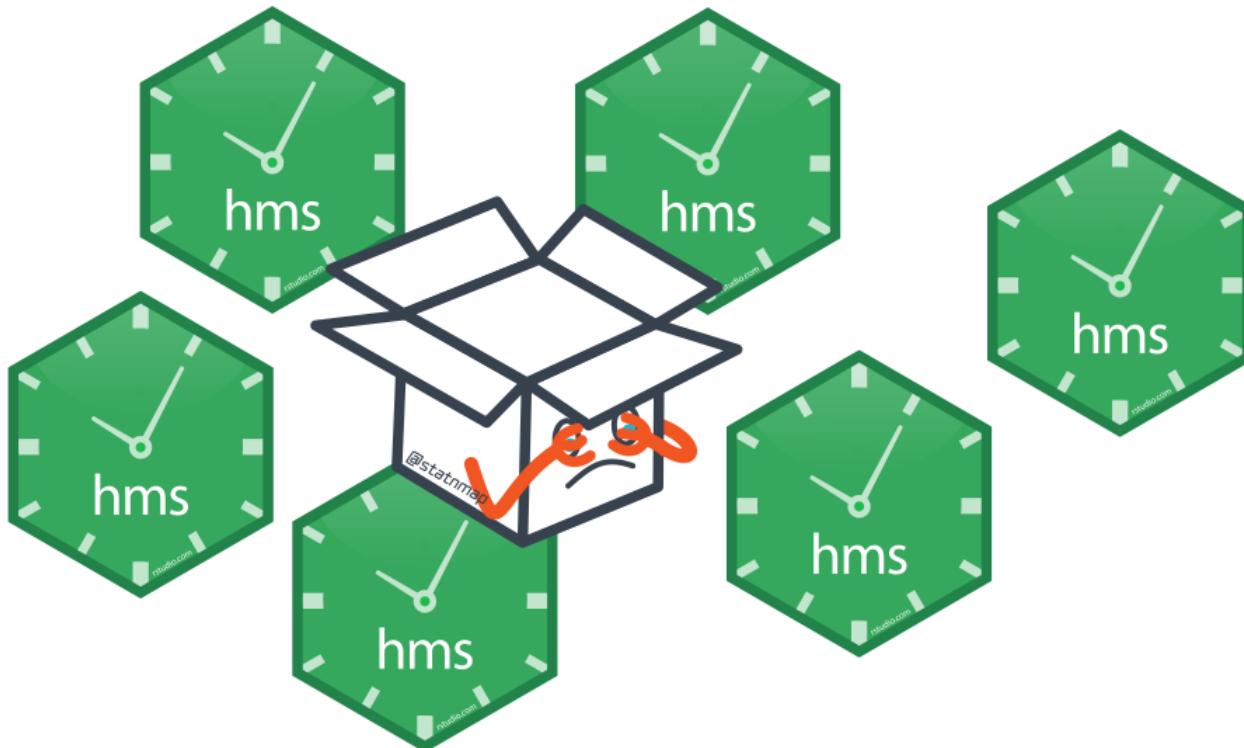


**Cervan
Girard**

Le nouveau

What is your worst enemy?

TIME?



What is your (data analyses) strategy?

- Code quickly
 - Get quick results and "nice" figures
- ⇒ *Be the first to present to colleagues/boss or to publish*

What is your real medium-term future?

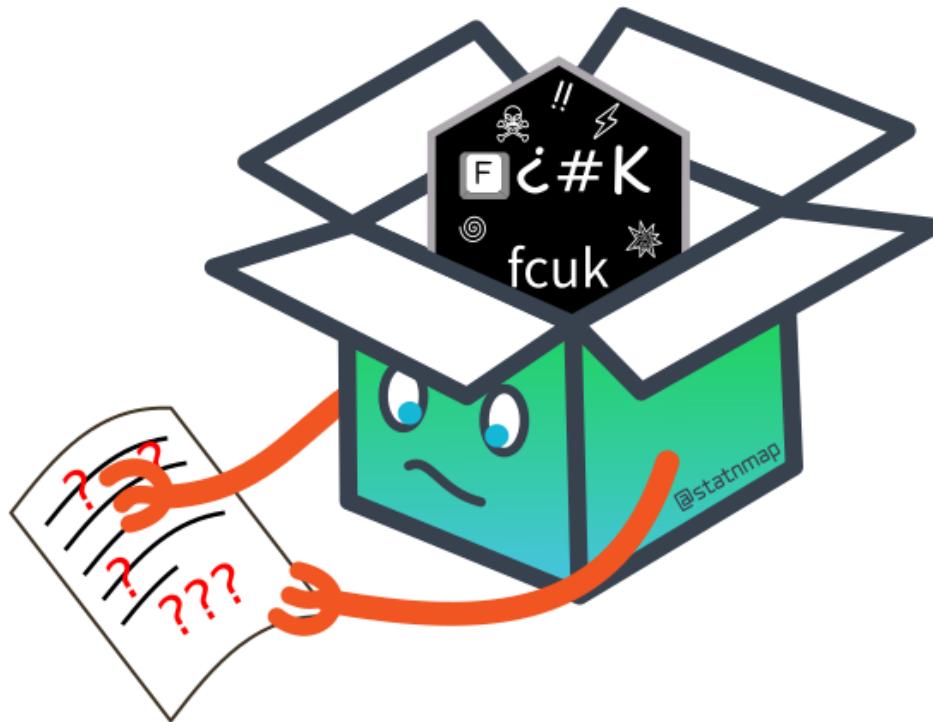
- Correction of bad raw data and filters
- Add external data from new collaborators
- Add complexity to models
- Create new data, test methods asked by reviewer #2
- Train a student on it to "accelerate" analyses
- Train a PhD student to take upon the analyses of the updated yearly survey
- Re-submit your paper with scripts asked by editor
- Share your code with the world in many ways



Will you be able to go back to your code?

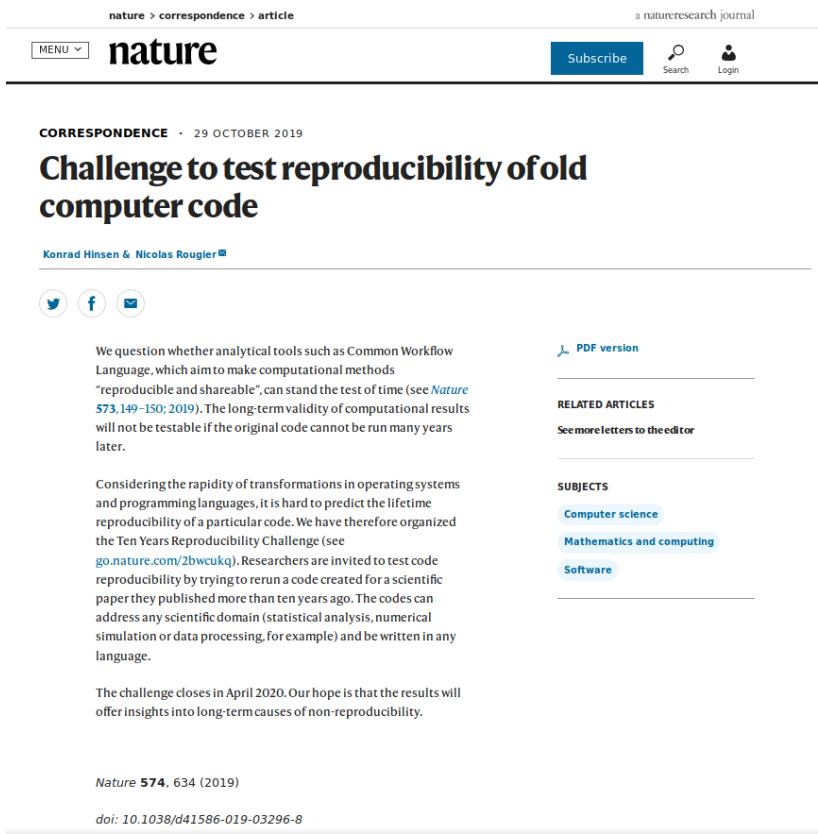
What will your "future you" think about your "past you" wasting his time?

What would have been a better use of your past time?



Will you take up the challenge?

<https://www.nature.com/articles/d41586-019-03296-8>



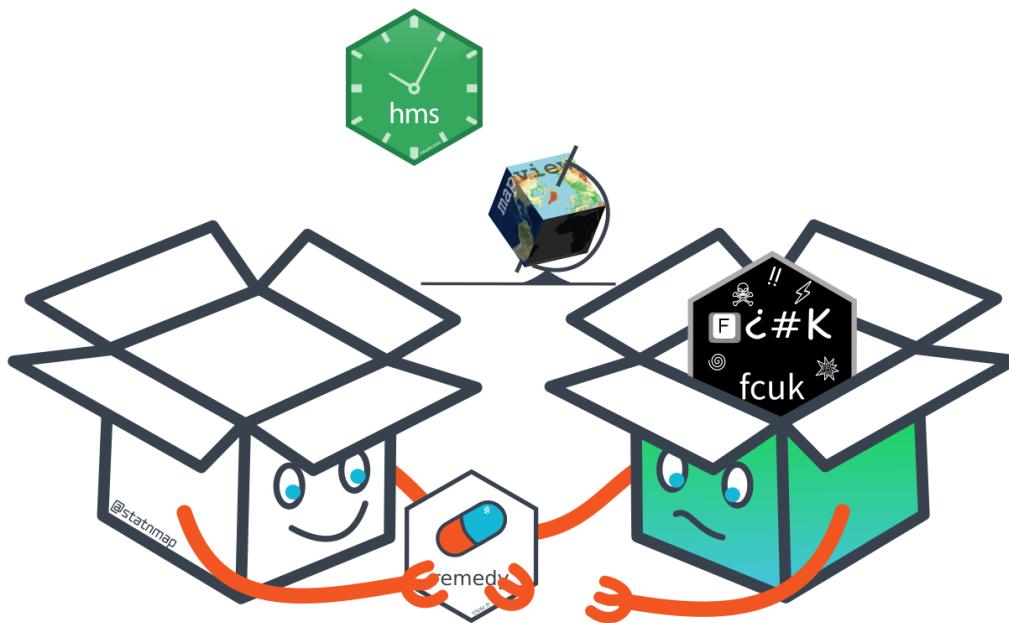
The screenshot shows a Nature journal article page. At the top, there's a navigation bar with 'nature > correspondence > article', the 'nature' logo, and links for 'Subscribe', 'Search', and 'Login'. Below the header, the article title is 'CORRESPONDENCE · 29 OCTOBER 2019 Challenge to test reproducibility of old computer code' by Konrad Hinsen & Nicolas Rougier. There are social media sharing icons for Twitter, Facebook, and Email. The main text discusses the challenge to test the reproducibility of old computer code, mentioning the Common Workflow Language and the Ten Years Reproducibility Challenge. It notes the difficulty of predicting the lifetime reproducibility of code. Researchers are invited to test code reproducibility by rerunning old scientific papers. The challenge closes in April 2020. The article is from Nature 574, 634 (2019) and has a DOI of 10.1038/d41586-019-03296-8.

⌚ The challenge closes in April 2020. Our hope is that the results will offer insights into long-term causes of non-reproducibility.

A correct workflow for reproducible science

- Documentation
- Unit tests
- Version control
- Continuous integration

Prevent code sickness and future waste of time





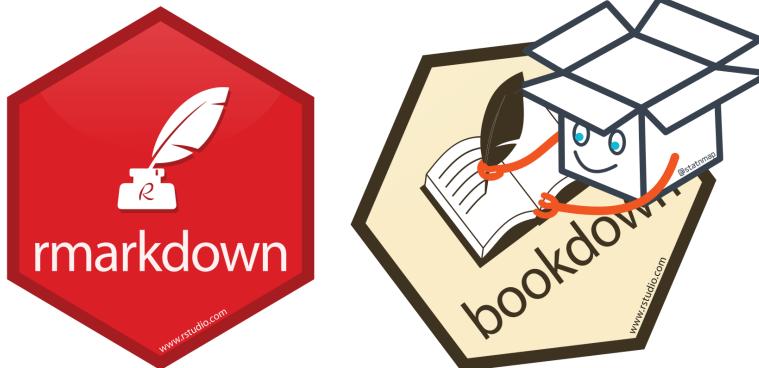
Why is documentation the last thing you do?

Why is documentation the last thing you do?

Start with the documentation !

"Rmd first" method

- Forces coding in a reproducible way
- Does not use too much of your present time
- Highly saves time of "future you"



Overview of the "Rmd first" method

Only develop in a Rmarkdown file

- Build your analyses
- Transform appropriate code blocks as functions
- Document your functions
- Test your functions

⇒ Apply directly on small reproducible examples and on your data



Let's try a workflow for data analysis

Download course material on https://github.com/statnmap/course_material/

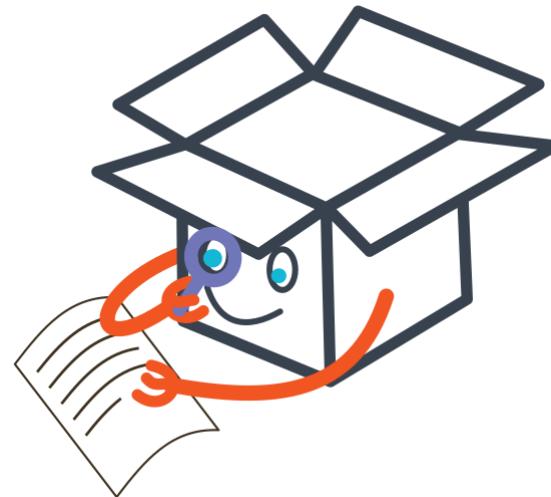
Directory: "2019-11-06_rmd_first_method"



Let's try a workflow for data analysis

- dataset: NYC Squirrel Census
 - source: <https://github.com/rfordatascience/tidytuesday>

```
#> -----
#> Interested in my life?
#> -----
#>          \
#>          \
#>          \
#>          . .
#>          | \ | \_
#>          /   @ \ 
#>          /   _-/_°
#>          \\\\\\\\\\\\\\   /   / \
#>          ///////////////   /   \   / |
#>          \\\\\\\\\\\\\\\\\\\\\\   /   /\\ \ \\
#>          ////////////////   /   \\ \ \\
#>          \\\\\\\\\\\\\\\\\\\\\\   /   / `` ``
#>          ///////////////   \   /   \
#>          ML      \\\\\\\\\\   /   /   /
```

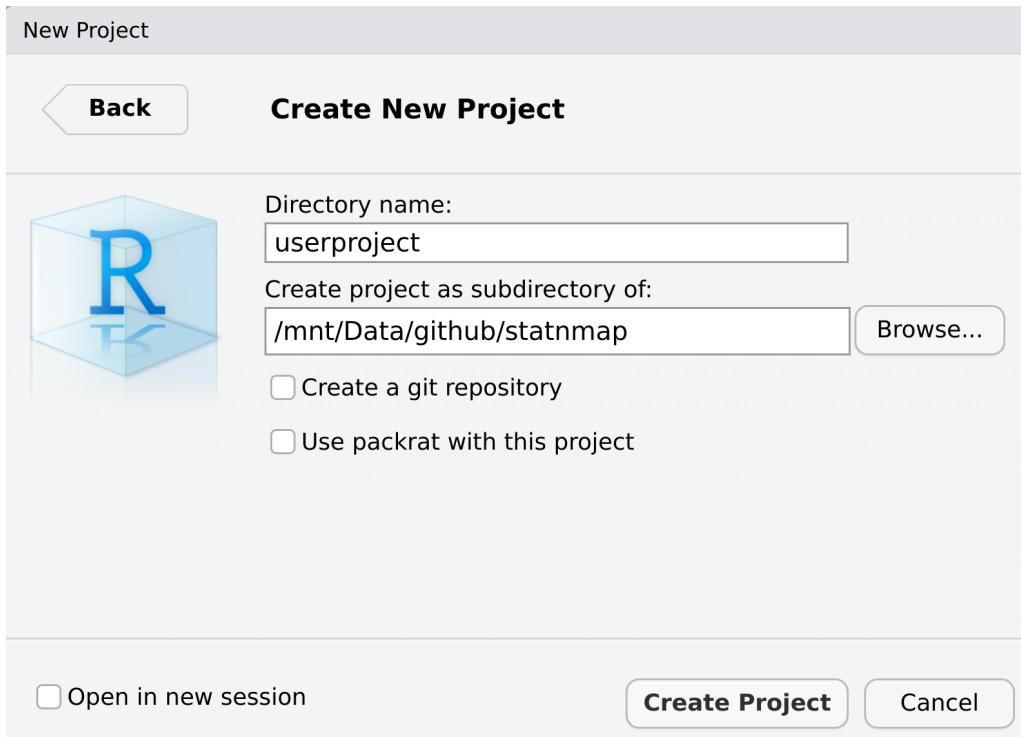


⇒ Here starts the live demo...
Download course material on
https://github.com/statnmap/course_material
Directory: "2019-11-06_rmd_first_method"

Always work in a project

```
one_analysis <- one_project <- one_folder
```

- Create *baby* project in Rstudio

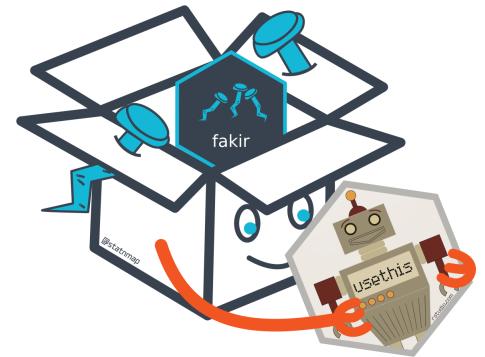
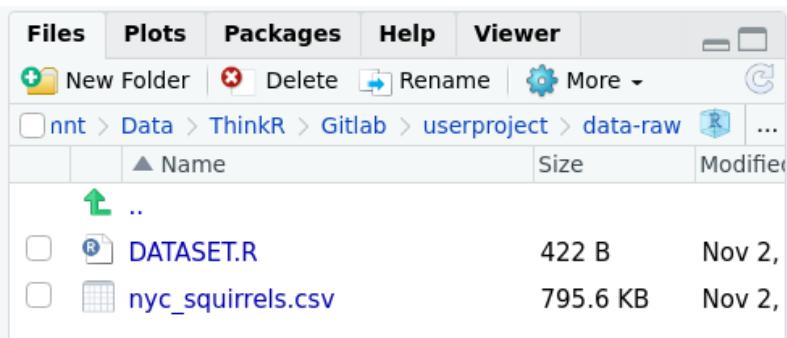


⚠️ These steps are to illustrate our approach, this is not exactly the workflow we will recommend. Stay focus until the end! **⚠️**

Add your data in "data-raw"

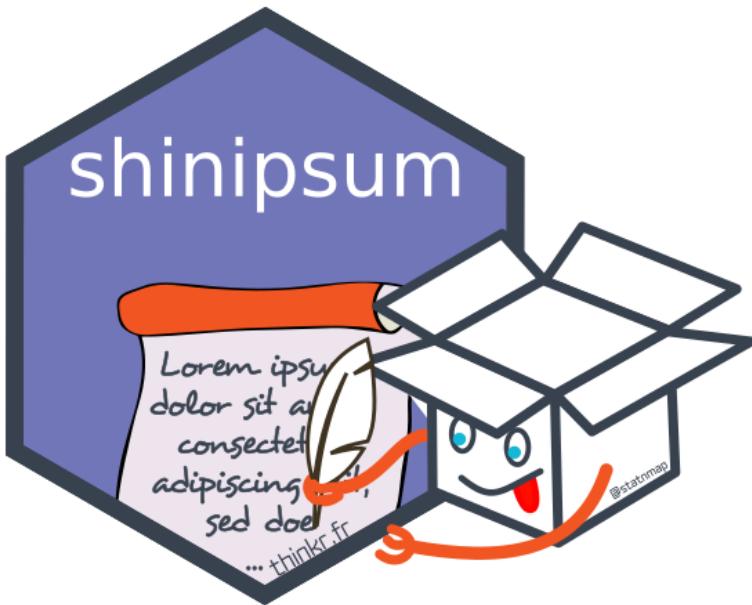
Package `{usethis}` is a useful tool

- use `usethis::use_data_raw()`
- Add your dataset to analyse in "data-raw"
- File "data-raw/DATASET.R" is for raw data preparation
 - *Copy and run "DATASET.R" of the course repository*
- This is a start. Data strategy is another question

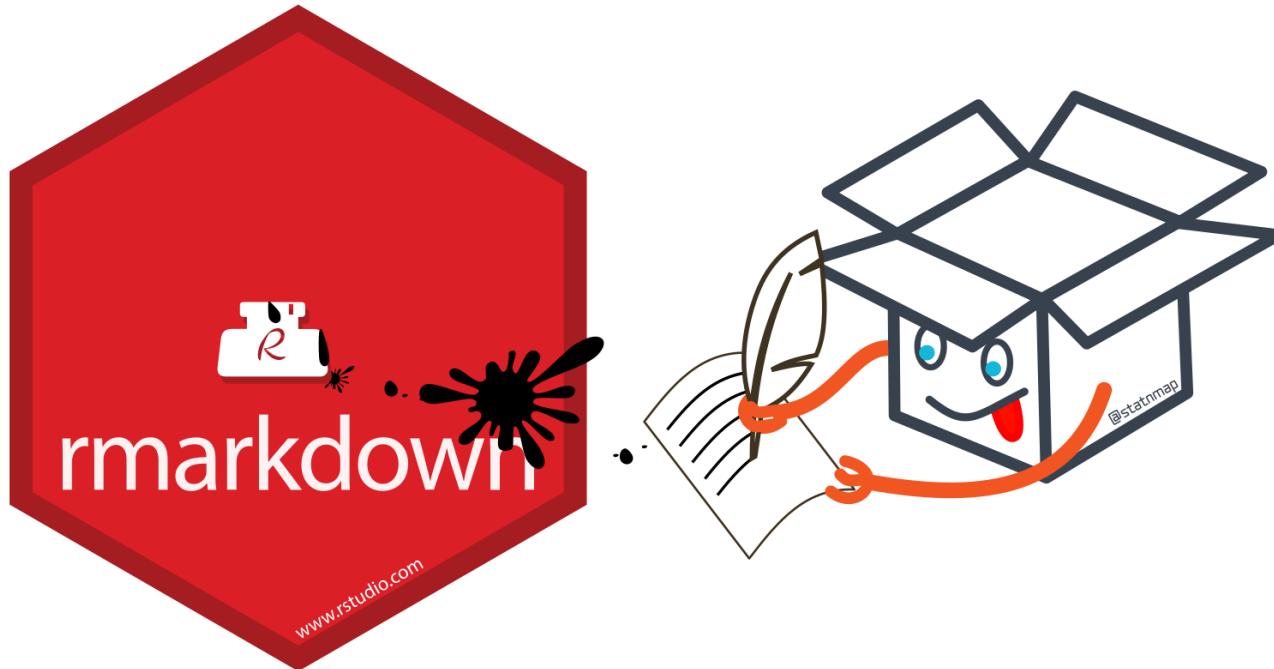


ThinkR tip: store development steps in a file

- Create a file named "dev_history.R"
- Store non-documented development stage directly applied in the console
- You will find it useful in the future...



Create a Rmd file at the root of the project



What is Rmarkdown?

Say what you will do...

...do what you said



Say what you will do...



```
DATASET.R x my-analysis.Rmd x
[File] [New] [Open] [Save] [ABC] [Search] [Knit] [Help] [Insert] [Run] [Edit] [View] [R Markdown]
24 ## Read a client database
25
26 Dataset has been built using package {fakir} available on Github with
`remotes::install_github("ThinkR-open/fakir")`. It is saved as a CSV
file in this project.
27
28 ```{r, message=FALSE}
29 dataset_path <- here("data-raw/clients.csv")
30 clients <- read_csv(dataset_path)
31 ```

32 ## Table by department
33 - Create a function to filter on a particular department
34
35 - Create a function to filter on a particular department
36
37 ```{r}
38 filter_by_dpt <- function(x, dpt) {
39   filter(x, id_dpt == dpt)
40 }
41 # Examples
42 filter_by_dpt(clients, dpt = 11)
43 ```

23:1 # Load packages
```



...do what you said



```

24 ## Read a client database
25
26 Dataset has been built using package {fakir} available on Github with
`remotes::install_github("ThinkR-open/fakir")`. It is saved as a CSV
file in this project.
27
28 ```{r, message=FALSE}
29 dataset_path <- here("data-raw/clients.csv")
30 clients <- read_csv(dataset_path)
31 ```

32
33 ## Table by department
34
35 - Create a function to filter on a particular department
36
37 ```{r}
38 filter_by_dpt <- function(x, dpt) {
39   filter(x, id_dpt == dpt)
40 }
41 # Examples
42 filter_by_dpt(clients, dpt = 11)
43 ```

23:1 | # Load packages | R Markdown

```



This is not
an hex !

Develop your analysis in the Rmd

Try Rmarkdown

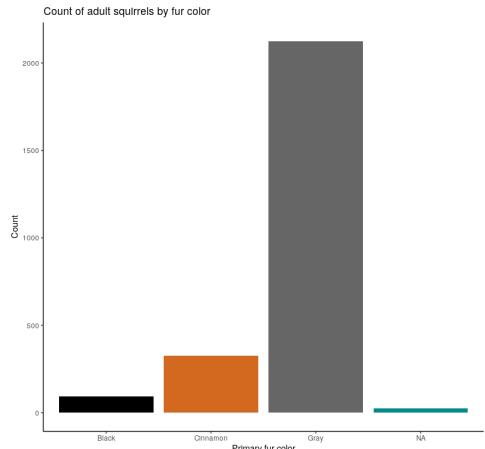
- File > New File > Rmarkdown > "OK"
- Save as "analyse-squirrels.Rmd"
- "Knit" to HTML to see what happens



Fill the Rmarkdown file

☞ Use content of file named "userproject/01-analyse-squirrels.Rmd"

- Read the dataset
 - Reproducibility: Use path relative to the project
- Clean the `date` and `age` columns
- Filter, count and plot fur color for one age group



☞ What if we want to build this same figure for both ages separately in our report?

Transform appropriate code as functions

💡 Use content of file named "userproject/02-analyse-squirrels-function.Rmd"

- Transform the code into a
 - `filter_count()` function
 - `plot_count()` function

```
# Function

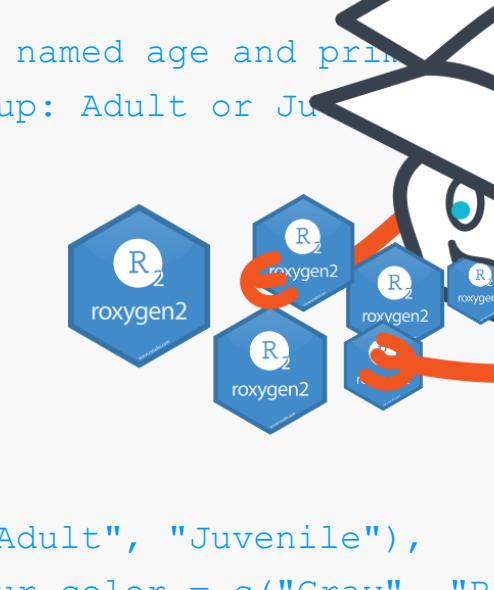
filter_count <- function(x, my_age) {
  # Test
  if (!all(c("age", "primary_fur_color") %in% names(x))) {
    stop("x should contain 'age' and 'primary_fur_color' columns")
  }
  x %>%
    filter(age %in% my_age) %>%
    count(primary_fur_color)
}
# Example
squirrels_filtered <- filter_count(x = nyc_squirrels, my_age = "Adult")
```

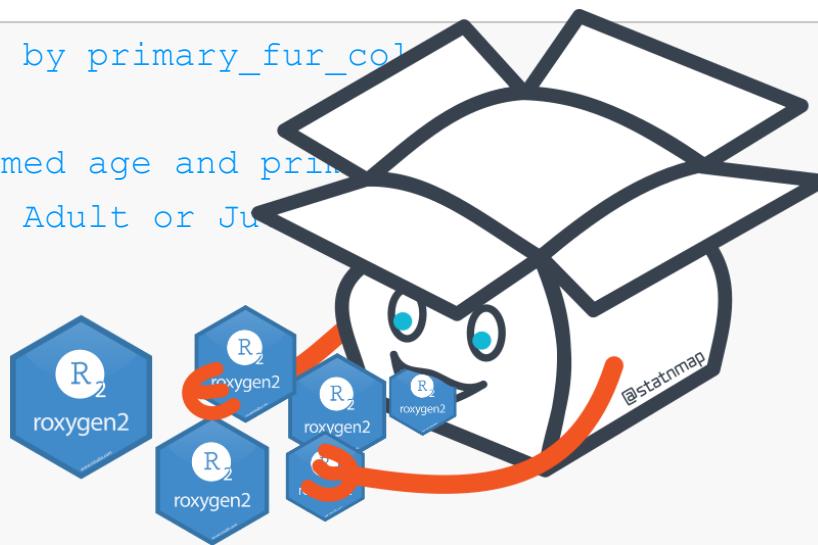
Document your function

-  Use content of file named "userproject/03-analyse-squirrels-roxygen.Rmd"

- Document the function using roxygen skeleton
 - Use your data example as a reprex

```
#' Filters data on age group and count by primary_fur_color
#'
#' @param x data.frame with columns named age and primary_fur_color
#' @param my_age Character. Age group: Adult or Juvenile
#'
#' @importFrom dplyr filter count
#' @importFrom magrittr %>%
#'
#' @export
#' @examples
#' library(dplyr)
#'
#' my_squirrels <- tibble(age = c("Adult", "Juvenile"),
#'
#'                         primary_fur_color = c("Gray", "Black"))
#'
#' filter_count(x = my_squirrels, my_age = "Adult")
filter_count <- function(x, my_age) { ... }
```

A cartoon squirrel wearing a lab coat and safety goggles is holding several blue hexagonal icons. Each icon contains a white 'R' with a subscript '2' and the text 'roxygen2'. A red ribbon is wrapped around the icons and the squirrel's arm.



Document your function (reprex)

💡 Use content of file named "userproject/03-analyse-squirrels-roxygen.Rmd"

- Use your data example as a reprex

```
library(dplyr)
#' @examples
my_squirrels <- tibble(
  age = c("Adult", "Juvenile"),
  primary_fur_color = c("Gray", "Black")
)
filter_count(x = my_squirrels, my_age = "Adult")
```

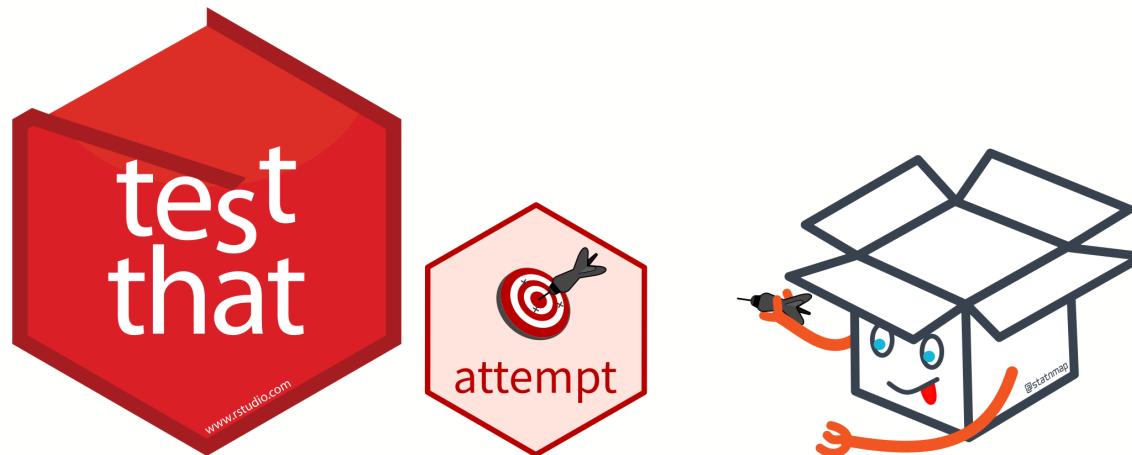


Create some unit tests on this function

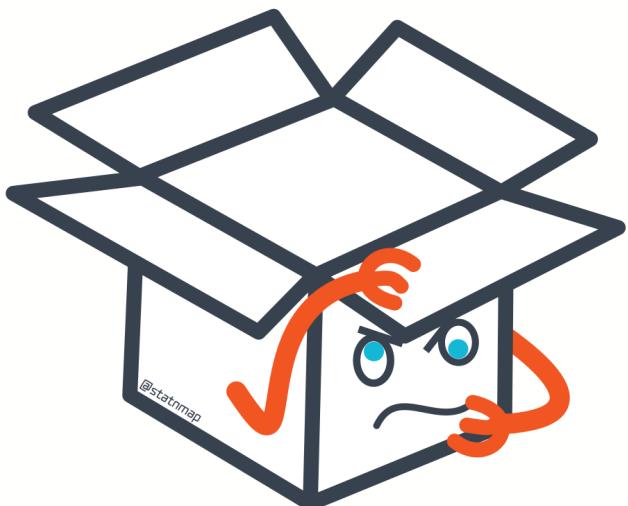
💡 Use content of file named "userproject/04-analyse-squirrels-tests.Rmd"

- Use the reprex to build some unit tests

```
my_squirrels <- tibble(age = c("Adult", "Juvenile"),
                        primary_fur_color = c("Grey", "Black"))
out_count <- filter_count(x = my_squirrels, my_age = "Adult")
# tests
expect_equal(nrow(out_count), 1)
expect_equal(ncol(out_count), 2)
```

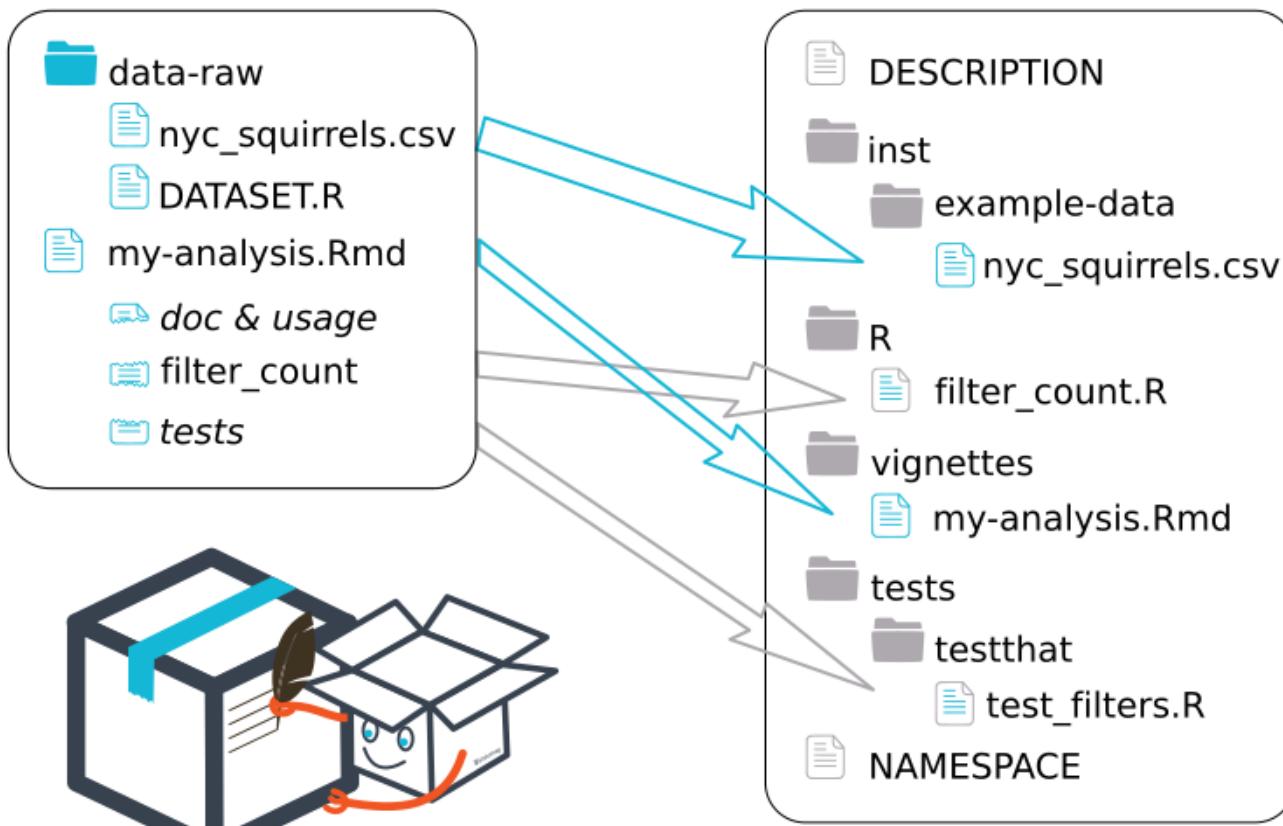


Where are we ?



We are (almost) in a package !

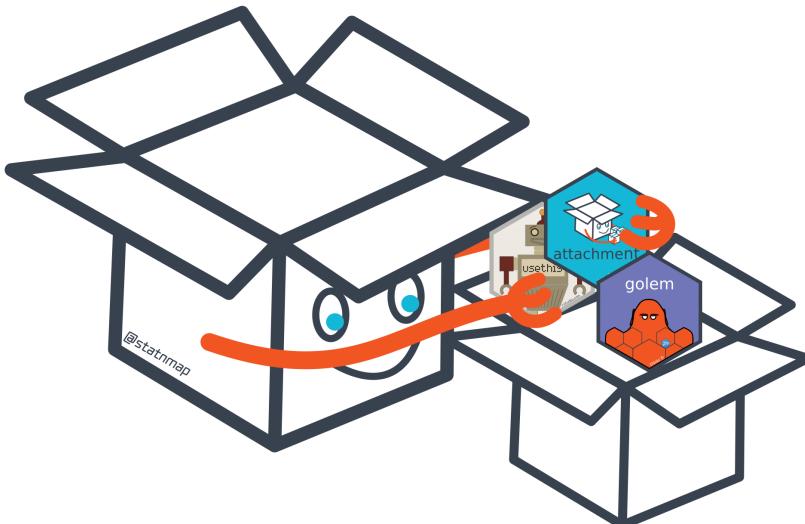
Clean your Rmd to see it



Let's build this package

💡 Use content of file "userpackage/dev_history.R" at the root

- File > New project >
 - New directory > "**R package using devtools**"
 - Name: "userpackage"
 - No special character, no space, dots `.` allowed
- Document your package in DESCRIPTION file
- Choose a License with `usethis::use_xxx_license()`



Move files and script in the correct place

💡 Use content of file "userpackage/dev_history.R" at the root

- Move data in "inst/example-data"
- Create a vignette with `usethis::use_vignette("my-analysis")`
 - Add the content of your Rmd file
 - Modify access to data using `system.file()`
- Create a R script with `usethis::use_r("filter_count")` to move functions
- Create test scripts with `usethis::use_test("filter_count")` to move code
- Use {attachment} to help with dependencies with
`attachment::att_amend_desc()`
- Check your package with `devtools::check()`

— R CMD check results —

—— userpackage 0.0.0.9000 —

Duration: 23s

0 errors ✓ | 0 warnings ✓ | 0 notes ✓

R CMD check succeeded



You built a package !



What's next?

Share the userguide

- Use `{pkgdown}` to publish
 - `pkgdown::build_site()`
- Customize your Rmd templates
 - [experimental] `chameleon::build_pkgdown()`



The screenshot shows a user interface for a package named "userproject". The top navigation bar includes a logo, the name "userproject", a version number "0.0.0.9000", a home icon, a "Reference" link, and a "Articles" dropdown. There is also a "Show/Hide all code" button.

userproject

The goal of userproject is to propose an example package with "Rmd-first" method for UseR 2019 conference.

- Directly create new "R package using devtools"**
 - `usethis::create_package("myproject")` if forgotten
 - 2. Create "dev_history.R" at the root
 - Use it to store all your usethis:: and co...
 - 3. Fill in DESCRIPTION file
 - 4. Create a small dataset
 - Store in "inst/example-data" or in "data"
 - 5. "Rmd first": Create a vignette
 - `usethis::use_vignette("aa-exploration")`
 - Build your functions along with vignette**
 - Document parameters
 - Add example of use
 - Store in R/
 - Run `attachment::att_to_description()`
 - Add unit tests
 - 7. Check your package regularly
 - 8. Use git and continuous integration
 - 9. Show your work to your colleagues and customers

userproject is a **THINKR** package. Learn more at task.thinkr.fr.

Developed by Sébastien Rochette. Site built by `pkgdown`.

'Rmd first' method for every project

Documentation matters

Document for you, document for developpers
Document for colleagues, document for your boss

Start with Rmd

- Start with a Rmd as a sandbox
- Document your functions with reproducible examples
- Create your tests while you code



This presentation on Github: [statnmap/prez](https://statnmap.com/prez)

'Rmd first' method for every project

Documentation matters

- Document for you and for others

Start with Rmd

- Create reprex, document and tests

Use this method for Shiny applications

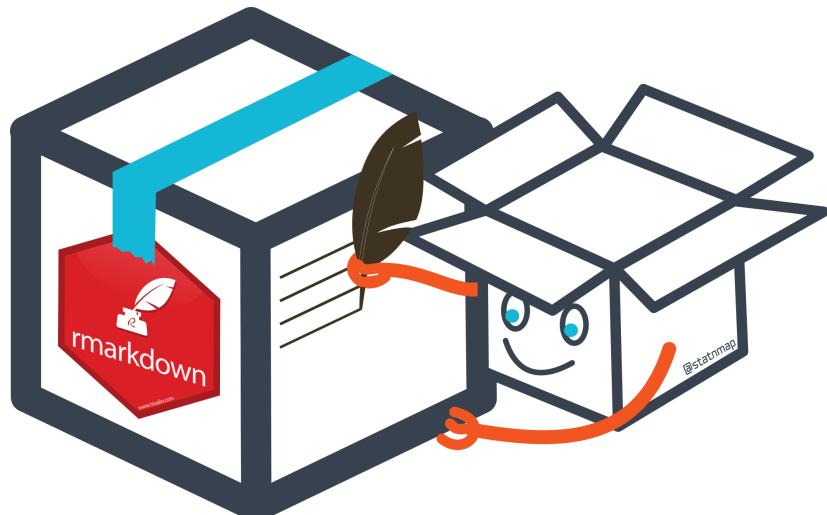
- Use with `{golem}`

THINK PACKAGE !

- Publish your analysis

THANK YOU for your attention

See more: rtask.thinkr.fr



This presentation on Github: statnmap/course_material