THINKR

The "Rmd first" method:

Documenting is taking care of your future

Meetup R Nantes

Sébastien Rochette

This presentation on Github: statnmap/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

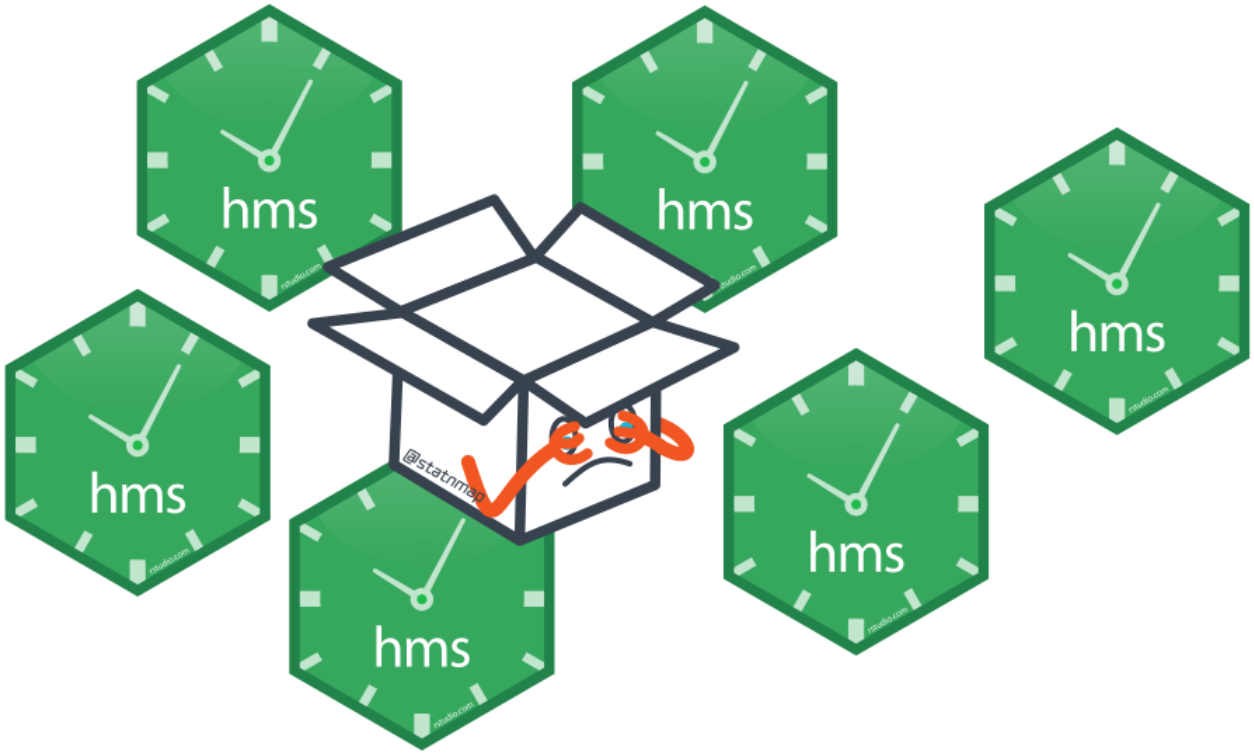
Colin Fay
Data scientist et R hacker

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 *"nices"* figures

⮊ *Present to colleagues/boss asap*

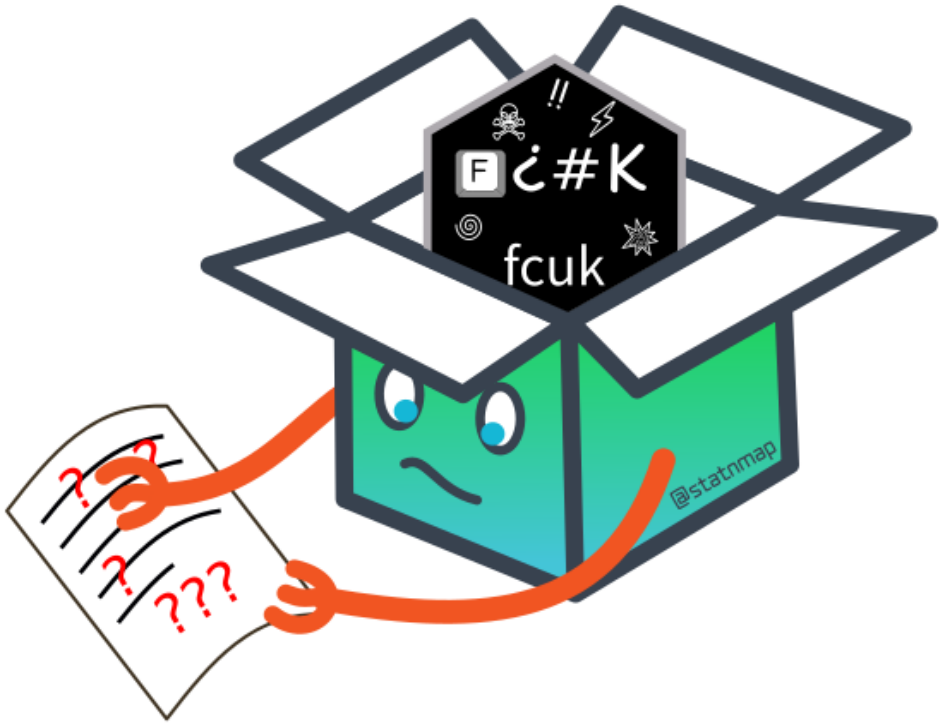
What is your real medium-term future?

- Correction of bad raw data and filters
- Add external data from new studies
- Add complexity to models
- Test other methods asked by colleague
- Train a student on it to *"accelerate"* delivery
- Transfer to another colleague the update of analyses when you are on holiday, sick leave, ...
- Share your code with the world in many ways

Will you be able to go back to your code in a few months?

What will your "future you" think about your "past you" wasting his time?
How long will be the headache of your colleagues taking upon your analyses?

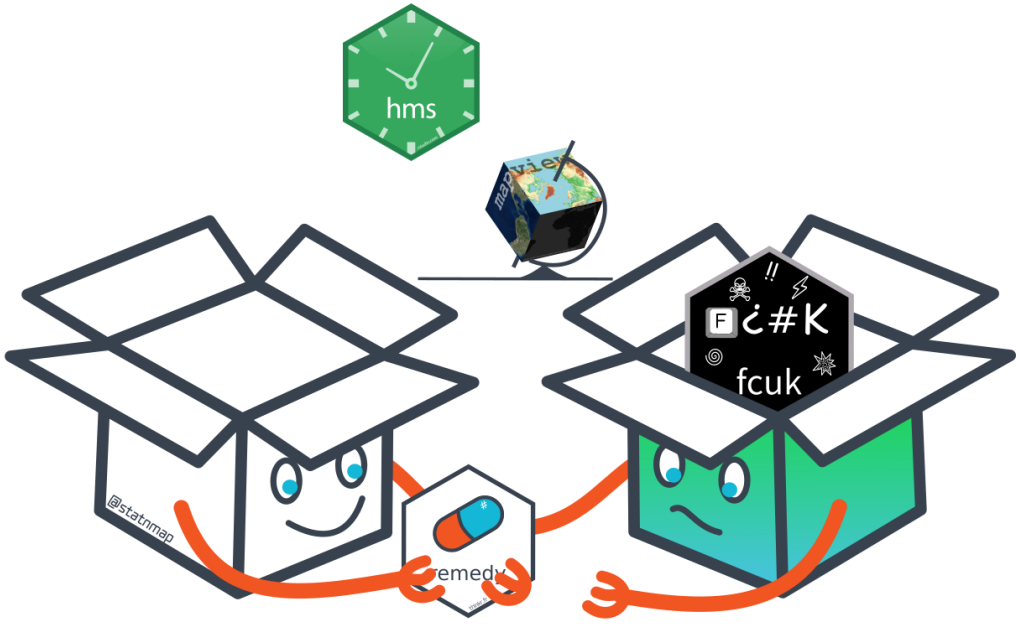
What would have been a better use of your past time?



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?

Start with the documentation !

"Rmd first" method

- Forces coding in a reproducible way
- Does not use too much of your present time
- Does not mean to write a book
- 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/

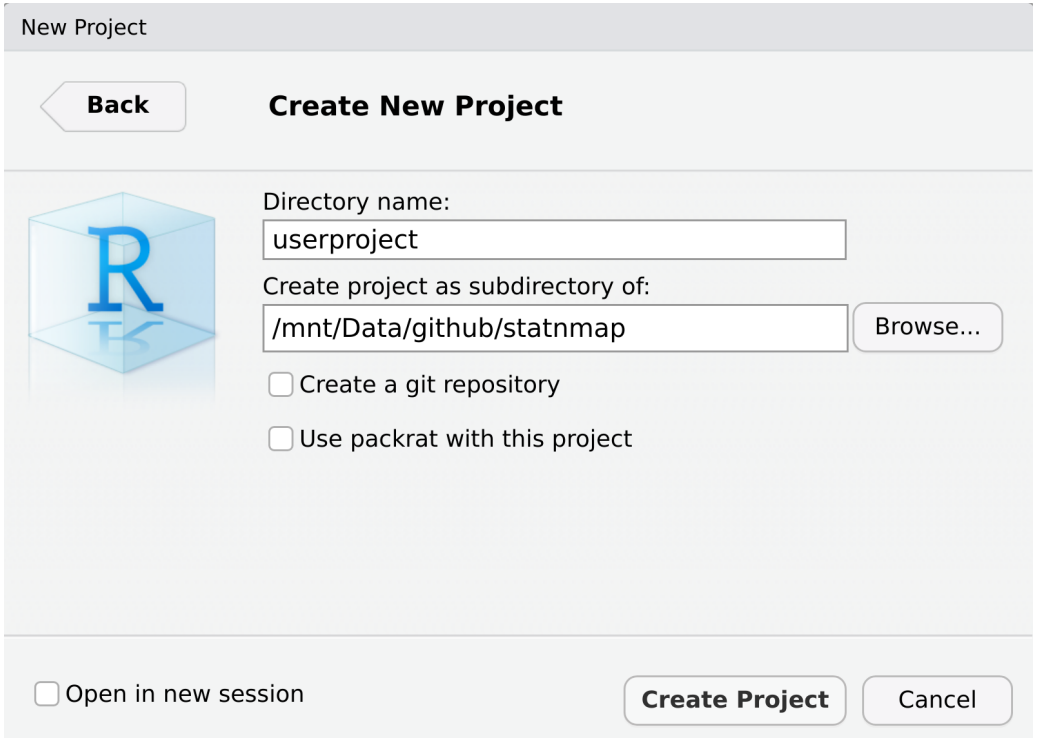
```
#>                ------
#>            Interested in my life?
#>                ------
#>                    \
#>                     \
#>                      \
#>                          . .
#>                          |\|\_
#>                          /  @ \
#>                         /  _-_/°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

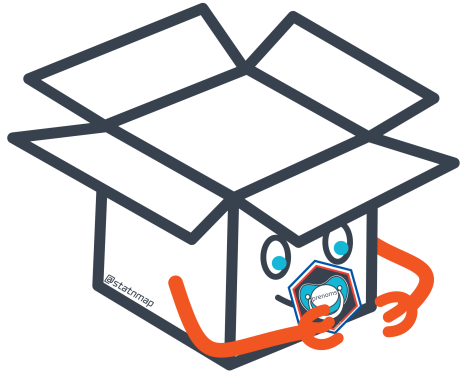
Here starts the live demo...
Download course material on
https://github.com/statnmap/course_material
Directory: "2019-11-06_rmd_first_method"

Always work in a project

`one_analysis <- one_project <- one_folder`

- Create *baby* project in Rstudio



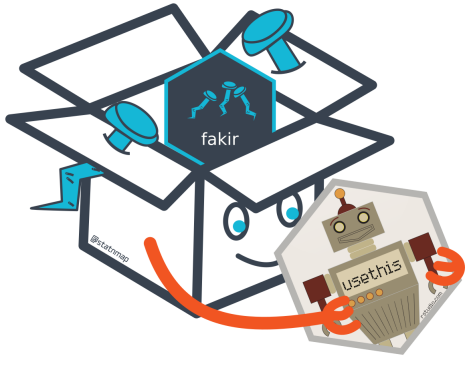


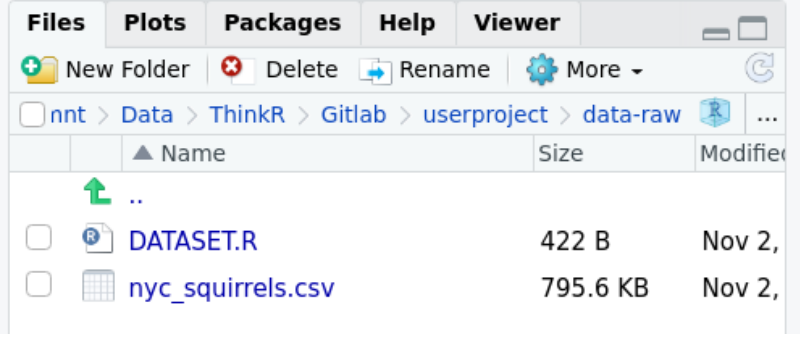
⚠ These steps are to illustrate our approach, this is not exactly the workflow we will recommend. Stay focus until the end! ⚠

Add your data in "data-raw"

Package {usethis} is a useful tool

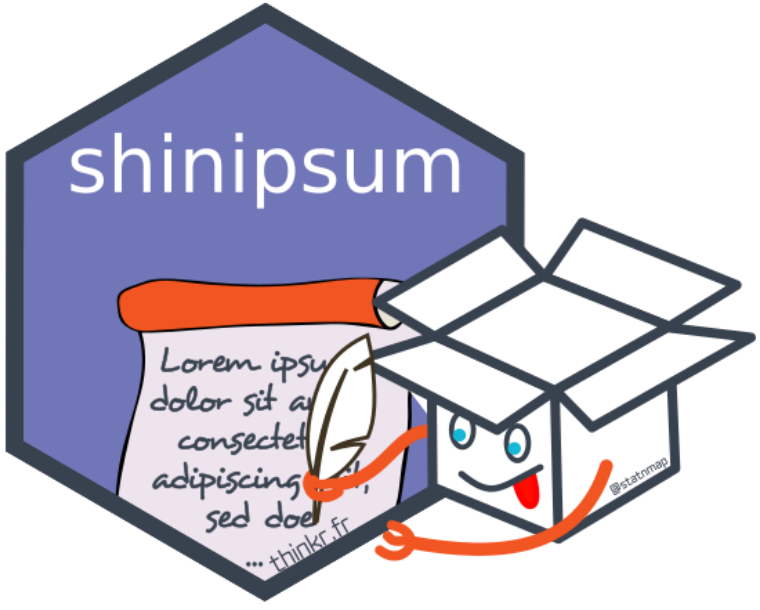
- use `usethis::use_data_raw()`
- Add your dataset to analyse in "data-raw"
- File "data-raw/DATASET.R" is for raw data preparation
  - *Copy and run "DATASET.R" of the course repository*
- This is a start. Data strategy is another question



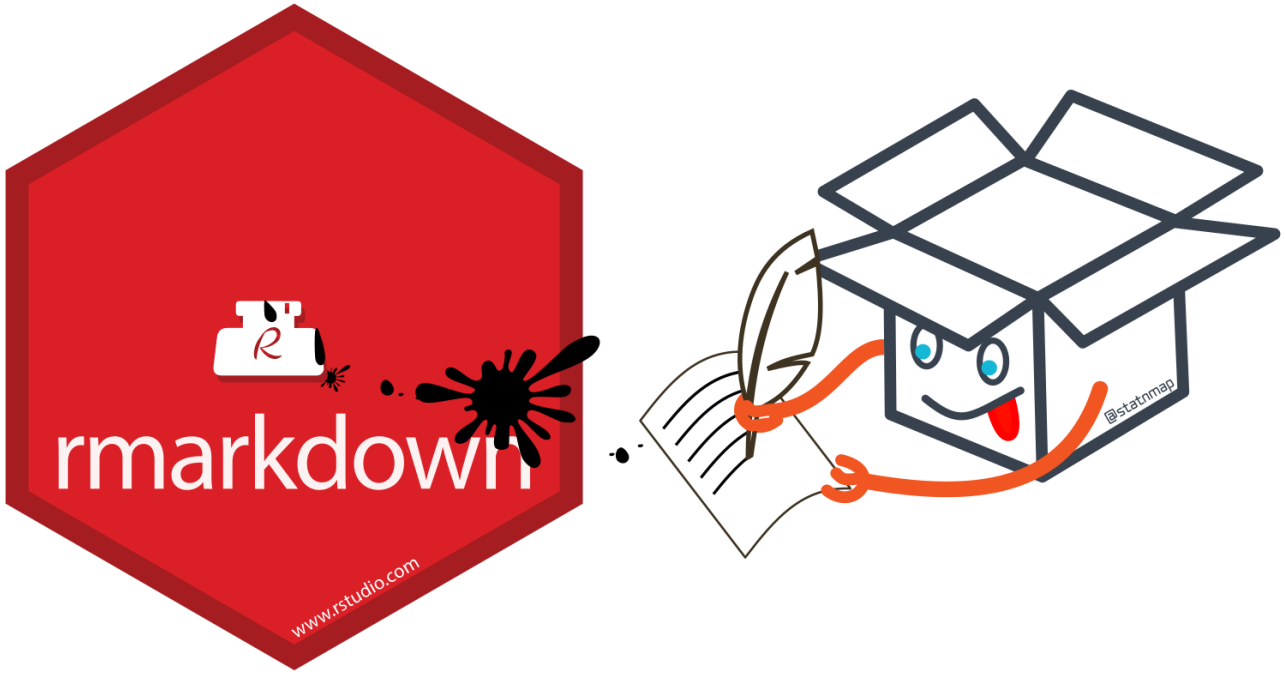


ThinkR tip: store development steps in a file

- 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...



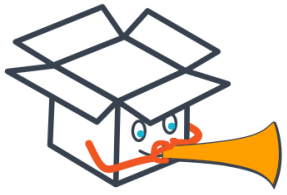
Create a Rmd file at the root of the project



Say what you do...

...do what you said

Say what you do...



```
## Read a client database

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.
```

```
## Table by department

- Create a function to filter on a particular department
```

...do what you said

DATASET.R | my-analysis.Rmd

Knit | Insert | Run

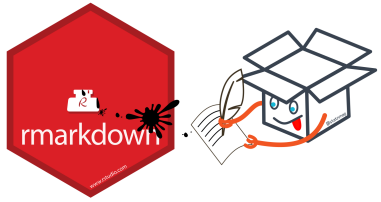
```
24 ## Read a client database
25
26 Dataset has been built using package {fakir} available on Github with
   `remotes::install_github("ThinkR-open/fakir")`. It is saved as a CSV
   file in this project.
27
28 ```{r, message=FALSE}
29 dataset_path <- here("data-raw/clients.csv")
30 clients <- read_csv(dataset_path)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

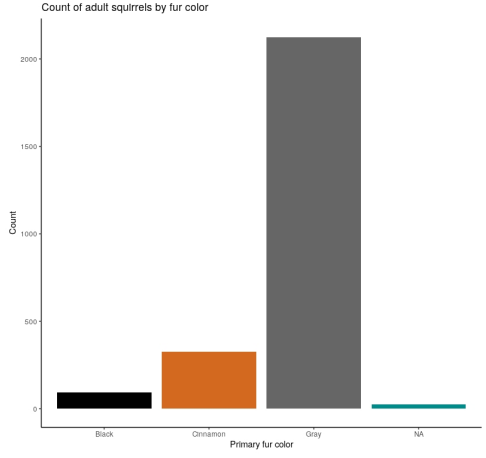


- File > New File > Rmarkdown > "OK"
- Save as "analyse-squirrels.Rmd"
- "Knit" to HTML to see what happens

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

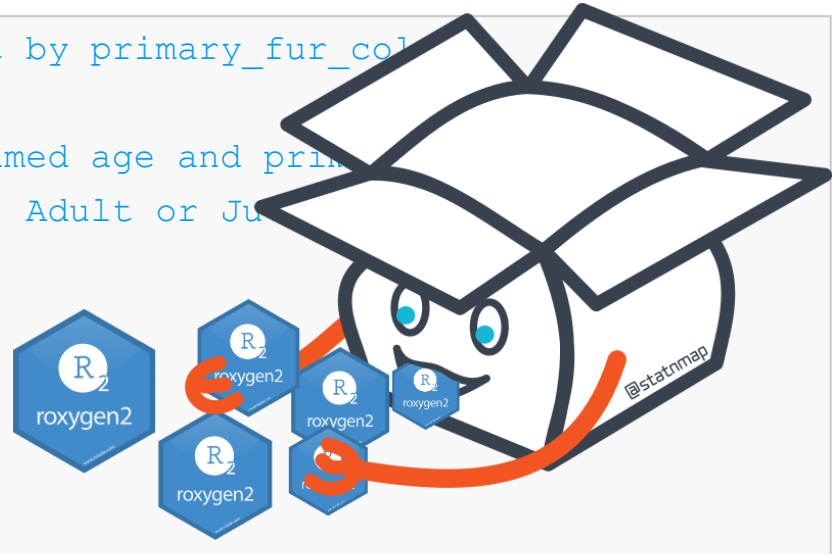
```
# Function
filter_count <- function(x, my_age) {
  # Test
  if (!all(c("age", "primary_fur_color") %in% names(x))) {
    stop("x should contain 'age' and 'primary_fur_color' columns")
  }
  x %>%
    filter(age %in% my_age) %>%
    count(primary_fur_color)
}
# Example
squirrels_filtered <- filter_count(x = 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_col
#'
#' @param x data.frame with columns named age and prim
#' @param my_age Character. Age group: Adult or Ju
#'
#' @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) {...}
```



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"

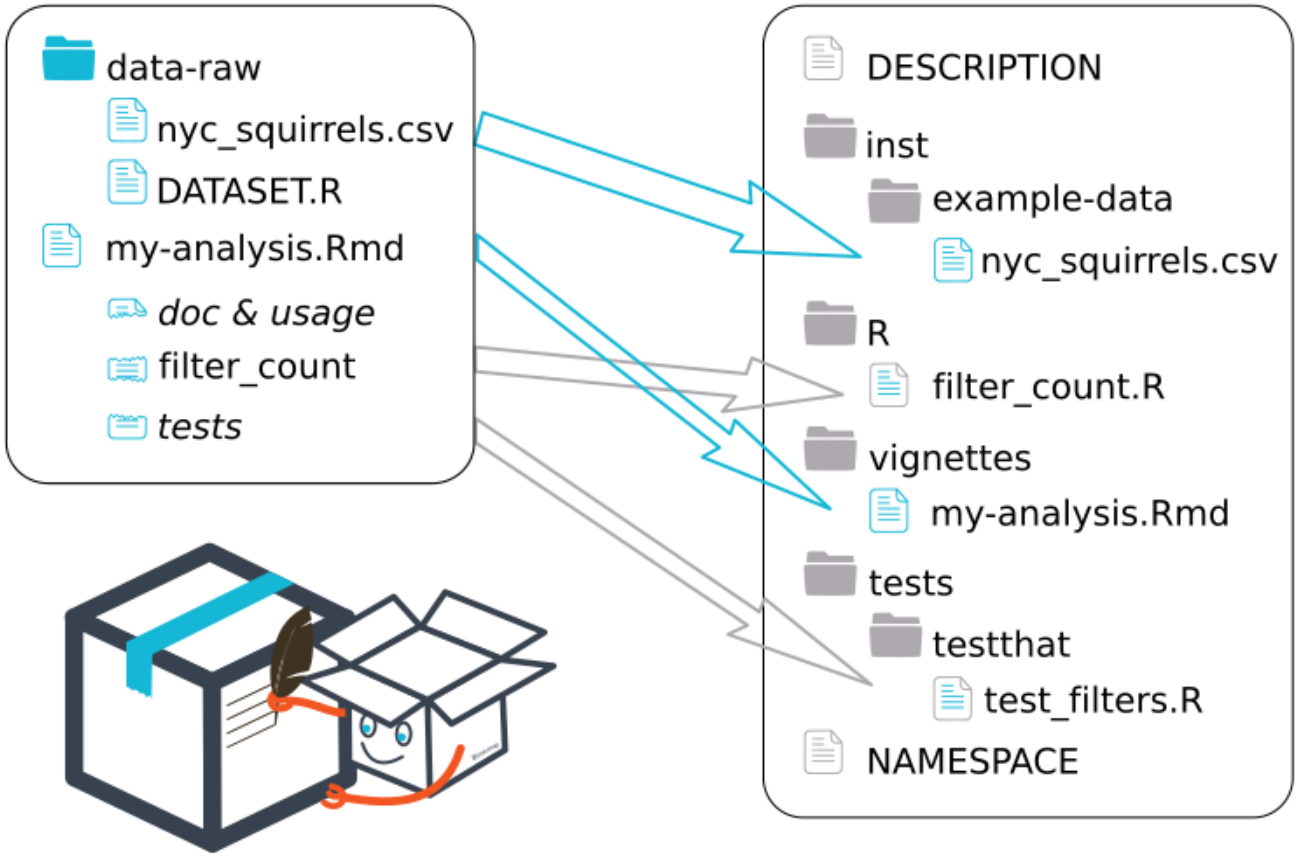
- Verify if function still work correctly after dependencies update
- Use the reprex to build some unit tests

```
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 !

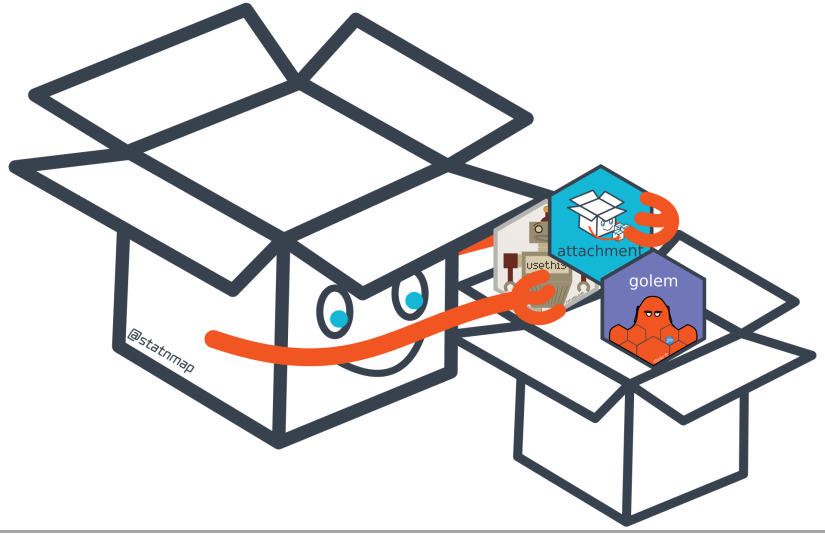
Clean your Rmd to see it



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 step by step

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

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

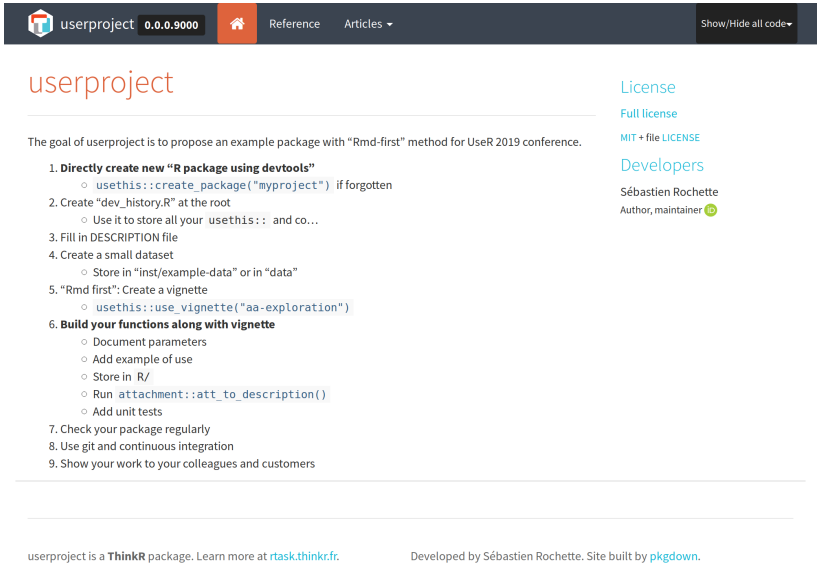
```
── R CMD check results ─────────────────────
──── userpackage 0.0.0.9000 ────
Duration: 23s

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()`



'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

'Rmd first' method for every project

Documentation matters

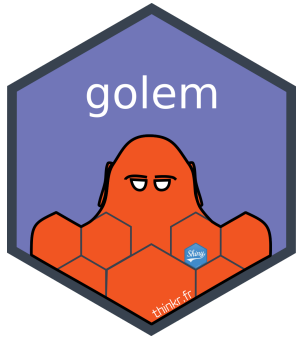
- Document for you and for others

Start with Rmd

- Create reprex, document and tests

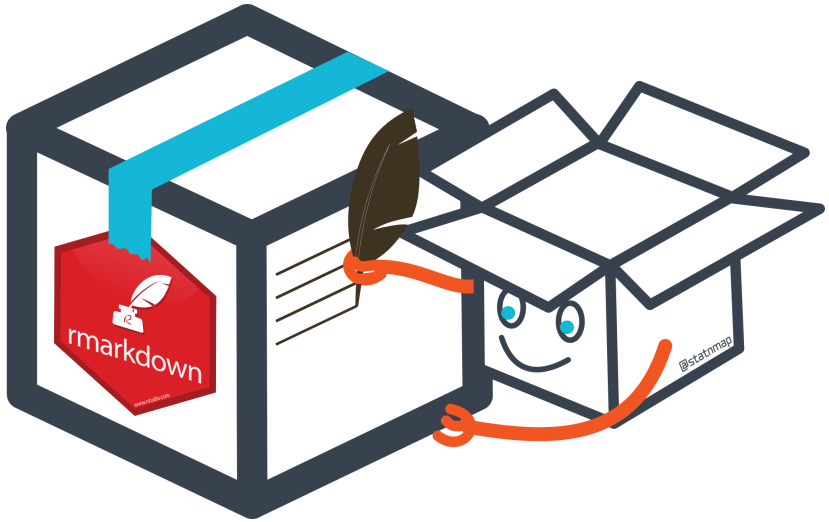
Use this method for Shiny applications

- 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