YOU MAY HAVE ALREADY WRITTEN A PACKAGE AND NOT EVEN KNOW IT

RMD FIRST AND {FUSEN}

SÉBASTIEN ROCHETTE, THINKR

Sébastien

Team leader, R expert, R instructor.

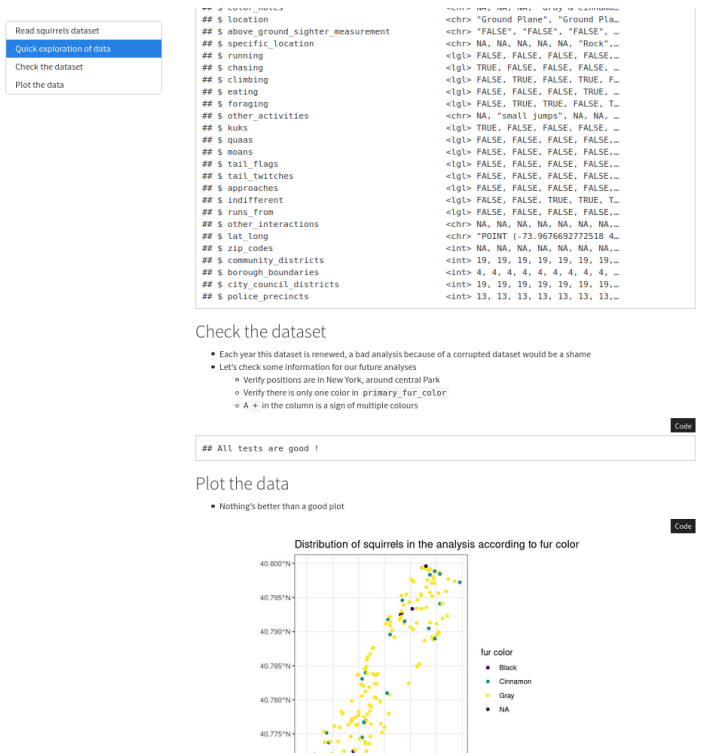
- ThinkR Website: https://rtask.thinkr.fr
- ThinkR GitHub: https://github.com/ThinkR-open
- ThinkR Twitter: @Thinkr_FR
- Personal website: https://statnmap.com
- Personal Twitter: @statnmap

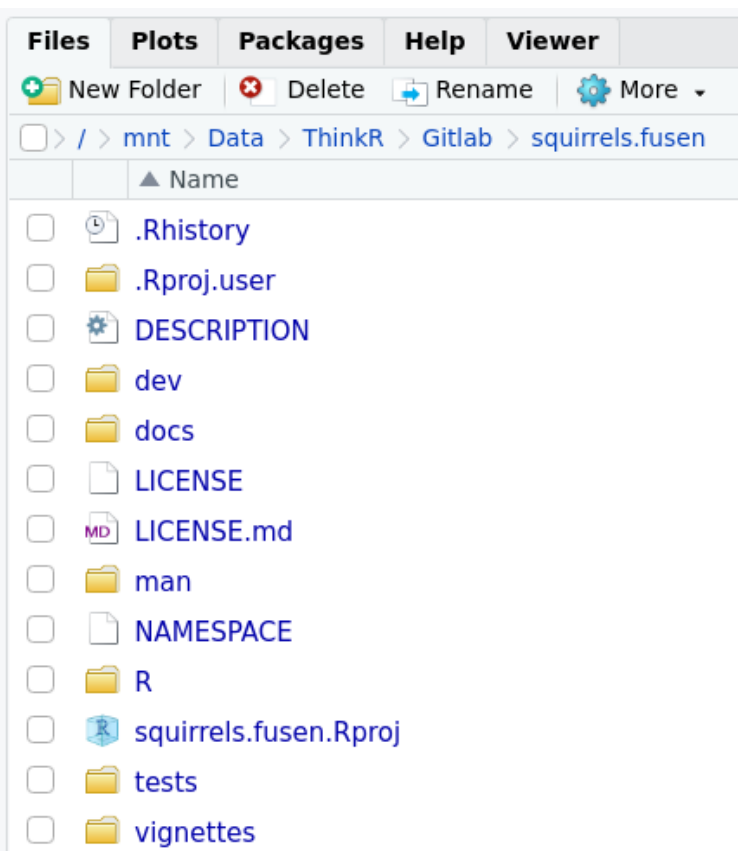
What is this presentation about?

Do you analyse regularly updated data?
Do you write “one-shot” code, but are used to re-using or sharing it?

Let’s see why you should think about packaging it…

- A Rmarkdown is almost already a package
- What is really a R-package?
- How to inflate a flat file as a package?





Q: Do you work with R Markdown / Quarto documents?

- A: Yes, everytime
- B: Yes, sometimes, for reports or other specific cases
- C: No, but I know what it is and eventually I tried once
- D: No, and I don’t really know what is is

R Markdown / Quarto file, what for?

- A file format, which allows to mix code and text
- Execute R code, in “chunks”
- Allow reproducible and literal analysis
- A (dynamic) document that is easy to share, distribute and publish

> *A good basis to document your analysis!*

Anatomy of R Markdown / Quarto

```
---
title: "Untitled"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

2:1 Untitled R Markdown

- **YAML header**: Metadata for the output
- **Text**: Text written using Markdown syntax
- **Chunks**: Place where you write classical R code

Knit the R Markdown / Render the Quarto

```
---
title: "Untitled"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

2:1 Untitled R Markdown

Untitled

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

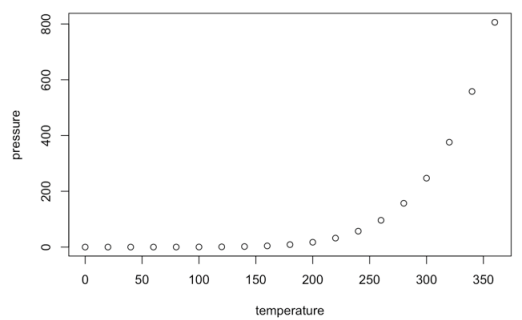
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

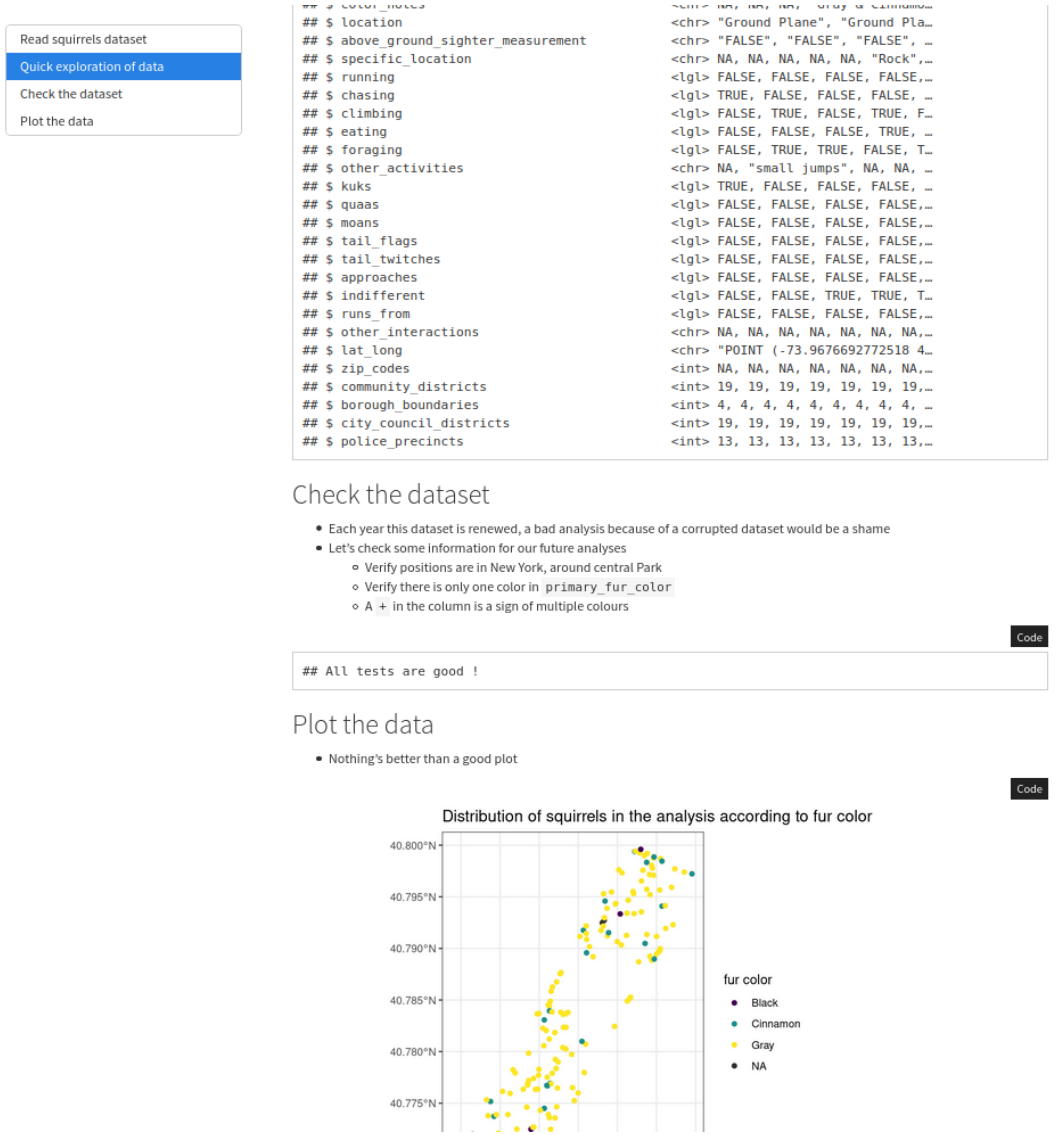
Including Plots

You can also embed plots, for example:



My data analysis in a Rmd

- TidyTuesday, NYC Squirrel Census:
  - Original study: https://www.thesquirrelcensus.com/
  - Data source: https://github.com/rfordatascience/tidytuesday/tree/master/data/2019/2019-10-29



Let's open the "nyc_squirrels_rmd_simple.Rmd" file and the associated "nyc_squirrels_rmd_simple.html"

You wrote the code of your data analysis, in a Rmarkdown or Quarto. What's next ?

Q: Which of these situations have you faced?

Maintenance

- A: Damn, I updated `{random.package}` last week, and my [old written] analysis does not work anymore
- B: I protected my project in a Docker / {renv}, and I want this last `{random.package}` functionality but that may break my code
- C: My colleagues modified part of our shared analysis for their data, but it broke mine somewhere
- D: Maybe I need to add some verification inside my scripts to protect from unfortunate modifications or inputs

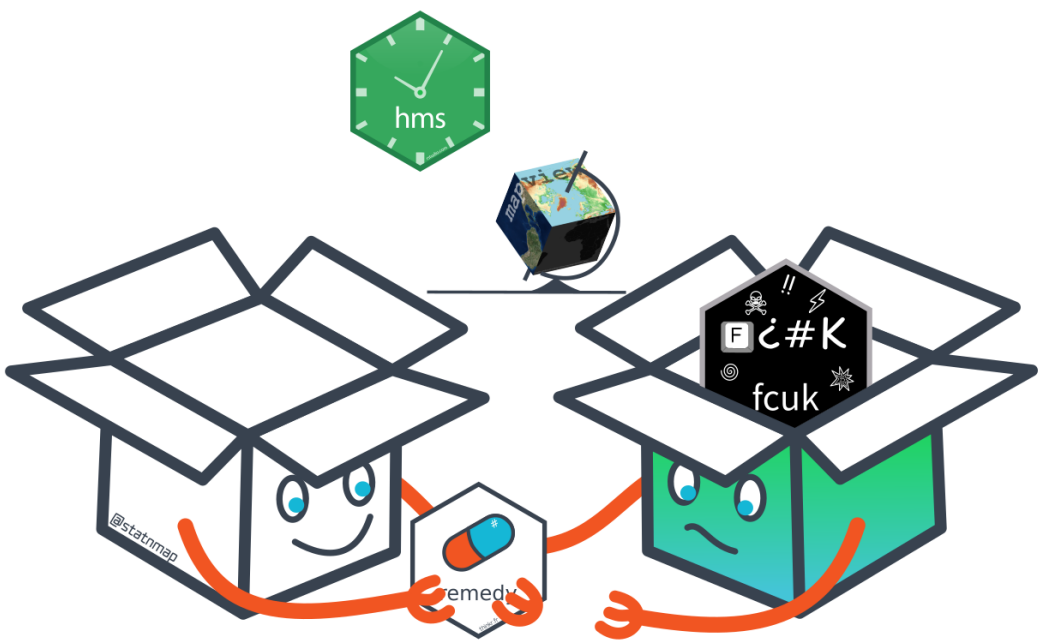
```
Warning: 'att_from_description()' est obsolète.
Utilisez plutôt 'att_amend_desc()'.
Voir help("Deprecated") et help("attachment-deprecated").

Error in old_function(): impossible de trouver la fonction "old_function"
```

Q: Which of these situations have you faced?

Collaboration

- E: My colleagues do not get how to adapt my scripts to their specific case and come ask me a new question every day
- F: I got someone else code, but each time I execute it, another package is missing…
- G: I use to copy-paste some lines, but with a small modification



Packages framework helps for these situations

Let's explore a package structure

- Package structure during development != installed on your computer
- Let's open the heart of {attachment}
- https://github.com/ThinkR-open/attachment

What to do with this package ?

My questions as a user

- What does it do?
- How to install it with its dependencies?
- What are its function?
- How to fill parameters of this function?
- Can I have an example on how to use this function?
- Can I have an example on how to use the package as a whole?
- Will it work with the last version of R and dependencies?

What to do with this package ?

My answers as a user

Questions	Answers
What does it do?	CRAN page: https://cran.r-project.org/web/packages/attachment/index.html
How to install it with its dependencies?	`install.packages('attachment')`
What are its function?	`?attachment` => Index
How to fill parameters of this function?	`?att_amend_desc`
Can I have an example on how to use this function?	`?att_amend_desc` => Examples
Can I have an example on how to use the package as a whole?	Vignettes, GitHub: https://thinkr-open.github.io/attachment/articles/fill-pkg-description.html
Will it work with the last version of R and dependencies?	Readme Check, https://github.com/ThinkR-open/attachment

The dedicated website gathers all these answers: https://thinkr-open.github.io/attachment/

What to do with this package ?

How the developers answered your questions?

- Let's open it: https://github.com/ThinkR-open/attachment

Questions	Answers
What does it do?	DESCRIPTION
How to install it with its dependencies?	DESCRIPTION
What are its function?	'R/' directory
How to fill parameters of this function?	'R/': Roxygen skeleton
Can I have an example on how to use this function?	'R/': `@examples`
Can I have an example on how to use the package as a whole?	'vignettes/' directory
Will it work with the last version of R and dependencies?	'tests/' directory and Continuous Integration

> *A minimum of 4 different places to store code and documentation*

Q: Have you already built a package with all these?

- A: Yes, everything. Functions, examples, tests, vignettes
- B: Only part of documentation. Functions, examples, maybe vignettes
- C: Only functions in a “R/” directory
- D: No. I never built a package from scratch

There are many things to set up as a developer

You probably already (almost) built a package

> *What ? Nah…*
> *The only thing I (barely?) know is the R Markdown / Quarto…*

Follow me...

The squirrels analysis - write the code

- First you write your script
  Here it checks that data updated each year are correct

```
#
# Verify points are in New York around Central Park
all_coords_ok <- all(
  c(
    min(nyc_squirrels[["lat"]]) > 40.76400,
    max(nyc_squirrels[["lat"]]) < 40.80100,
    min(nyc_squirrels[["long"]]) > -73.98300,
    max(nyc_squirrels[["long"]]) < -73.94735
  )
)
if (!all_coords_ok) {stop("Not all data are in Central Park")}

# Verify there is only one color in primary_fur_color.
# A `+` in the column is a sign of multiple colours
if (any(grepl("+", nyc_squirrels[["primary_fur_color"]], fixed = TRUE))) {
  stop("There are multiple colors in some 'primary_fur_color'")
}

message("All tests are good !")
#
```

The squirrels analysis - build the function

- Then, if this is not what you already do, transform it as a function
  - Make parameters explicitly declared
  - document properly with {roxygen2} syntax

```r
#' Check data integrity
#'
#' @param x dataframe with at least colums "lat", "long" and "primary_fur_color"
#'
#' @return Original dataframe if all tests are good. Otherwise stops.
#' @export
check_data_integrity <- function(x) {
  # Verify points are in New York around Central Park
  all_coords_ok <- all(
    c(
      min(x[["lat"]]) > 40.76400,
      max(x[["lat"]]) < 40.80100,
      min(x[["long"]]) > -73.98300,
      max(x[["long"]]) < -73.94735
    )
  )
  if (!all_coords_ok) {stop("Not all data are in Central Park")}

  # Verify there is only one color in primary_fur_color.
  # A `+` in the column is a sign of multiple colours
  if (any(grepl("+", x[["primary_fur_color"]], fixed = TRUE))) {
    stop("There are multiple colors in some 'primary_fur_color'")
  }

```


The squirrels analysis - demonstrate by example

- Usually, you verify that your code works with an example
  - Same here with the function and a reproducible example

Here, a data.frame with "lat", "long", "primary_fur_color" as input

```r
# A working example
my_data_example <- data.frame(
  lat = c(40.77, 40.78),
  long = c(-73.95, -73.96),
  primary_fur_color = c("grey", "black")
)
check_data_integrity(my_data_example)
```

```
All tests are good !
```

The squirrels analysis - test it

What do you visually verify with your examples?
Write it! You already have the code.

- Use some reproducible examples to test different possible cases

```
my_data_example <- data.frame(
  lat = c(40.77, 40.78), long = c(-73.95, -73.96),
  primary_fur_color = c("grey", "black")
)
my_data_example_error <- data.frame(
  lat = c(40.77, 40.78), long = c(-73.95, -73.96),
  primary_fur_color = c("grey+blue", "black") # not unique color
)

test_that("check_data_integrity works correctly", {
  expect_message(check_data_integrity(my_data_example), "All tests are good !")
  expect_error(check_data_integrity(my_data_example_error), "multiple colors")
})
```

```
Test passed 🌈
```

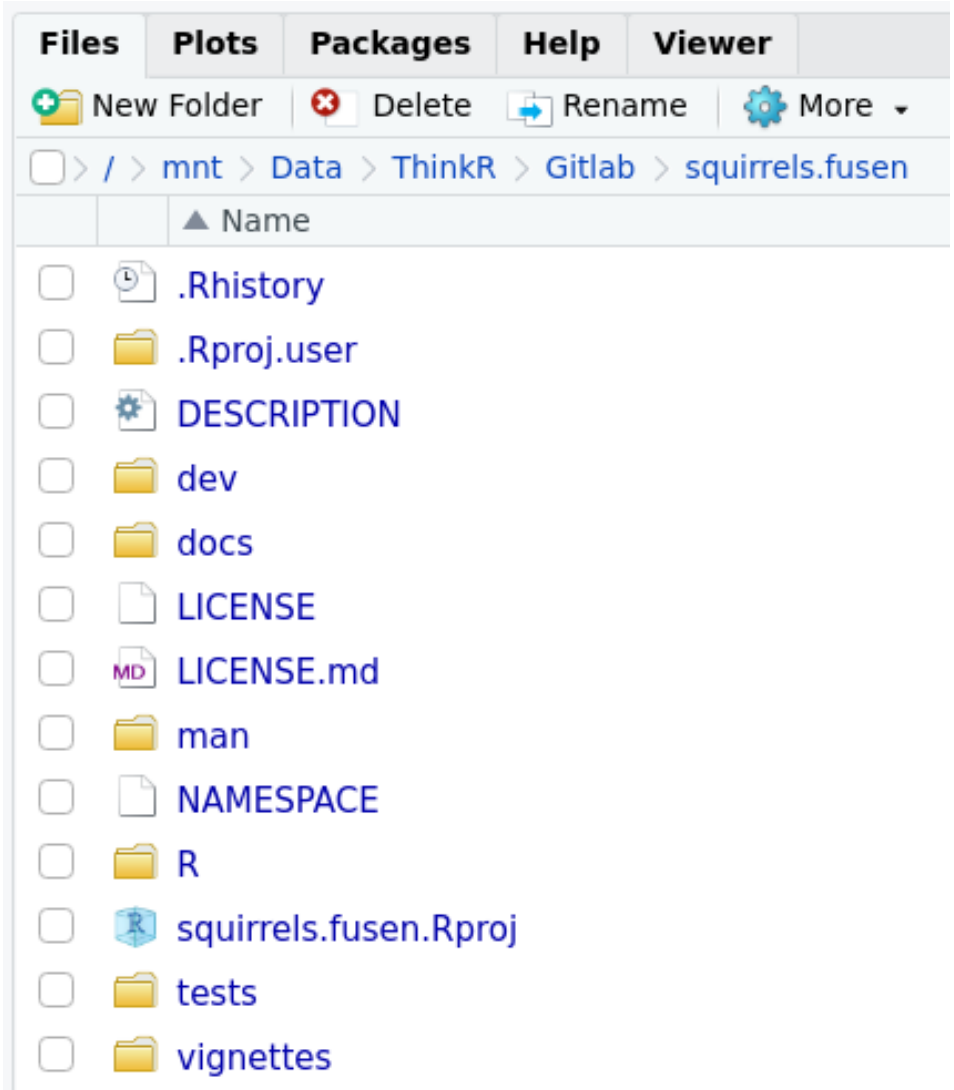
See ? You already write packages!

- This code is currently in a unique file
- Hopefully in a Rmd/Qmd, so that you can describe what it does between chunks of code
- You may also have a project structure with a specific folder for the scripts to source

What about the package structure?

Remember: there are different places to store code and documentation

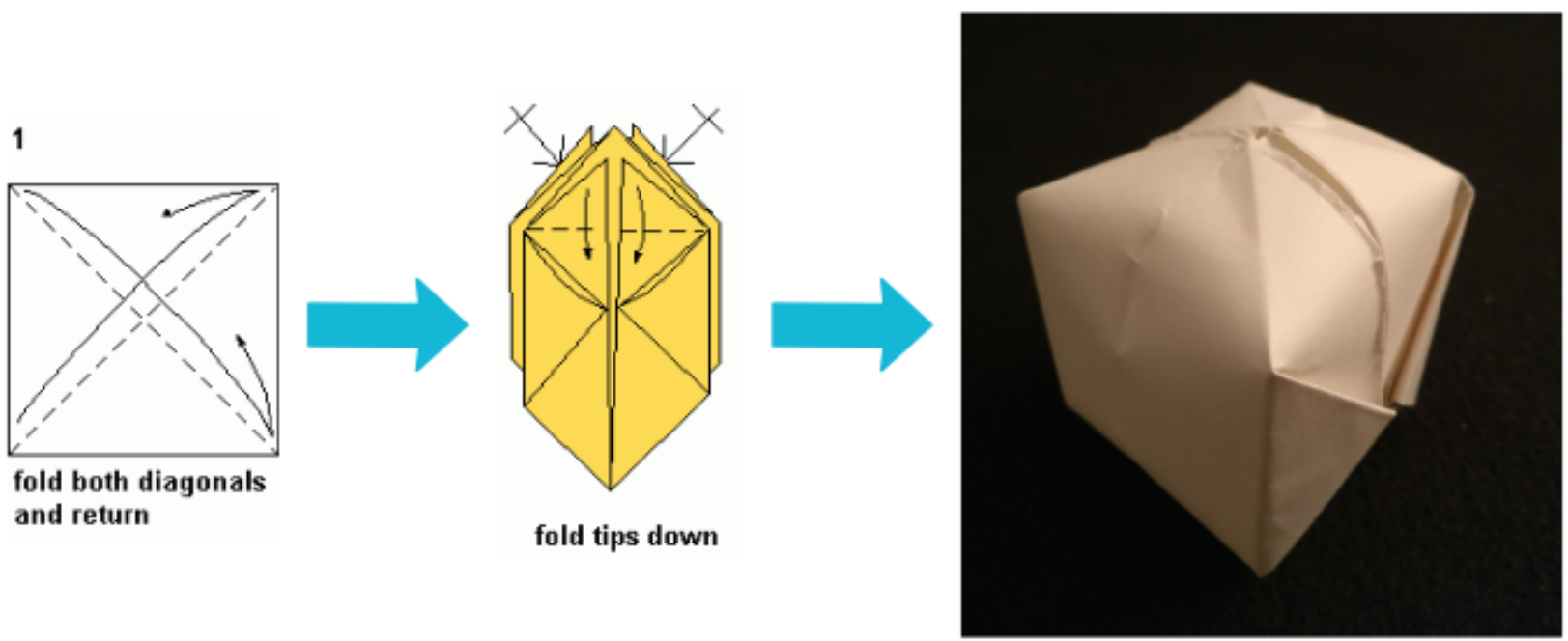
- **DESCRIPTION**: package documentation
- '**R/**' directory: functions and examples
- '**tests/**' directory: unit tests
- '**vignettes/**' directory: documentation


Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
/ mnt Data ThinkR Gitlab squirrels.fusen
Name
.Rhistory
.Rproj.user
DESCRIPTION
dev
docs
LICENSE
LICENSE.md
man
NAMESPACE
R
squirrels.fusen.Rproj
tests
vignettes

How to jump from Rmd to the package structure?

Many files and info to remember…

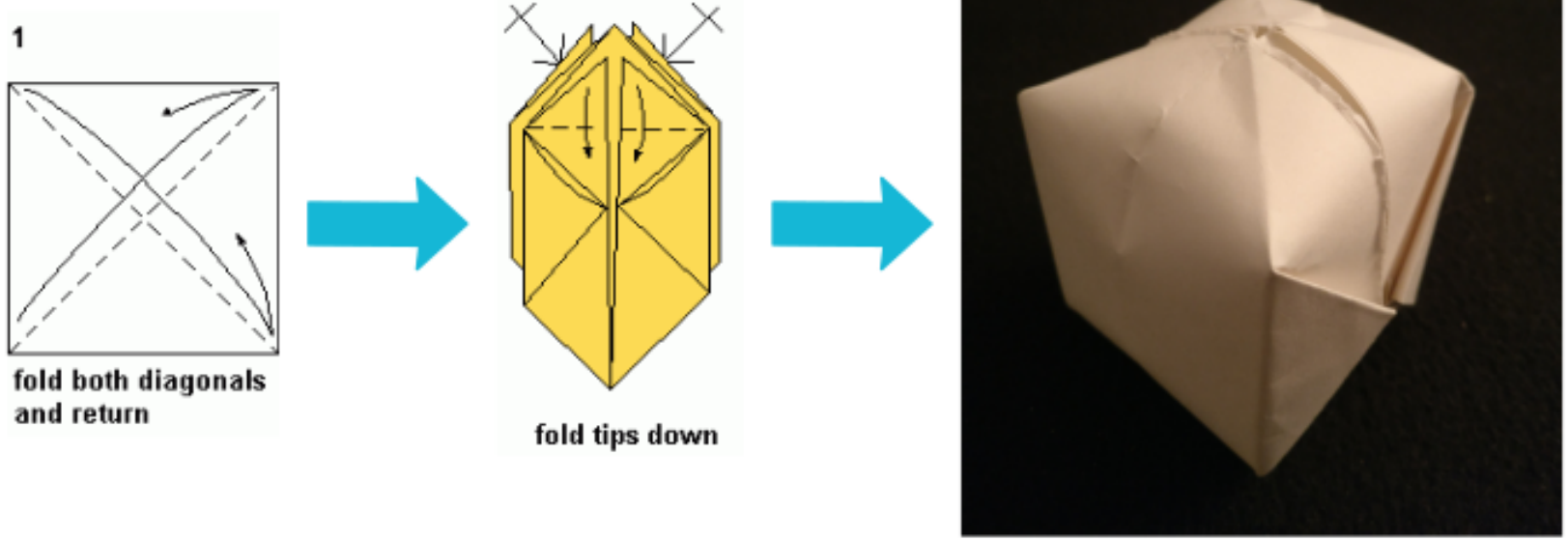
What if there was a package that could take a Rmd/Qmd file, a bit like a flat sheet of paper, and if you follow the right folding, you can inflate it as a package?



> *https://rtask.thinkr.fr/fusen-create-a-package-from-a-single-rmarkdown-file/*

Let {fusen} deal with the package structure

- Write your Rmd/Qmd
- Follow the folding lines
- Inflate



{fusen} needs to distinguish these places to be able to correctly distribute

- You'll need to separate your code according to actions : build, demonstrate, test, document

Arrange your code

BUSINESS
LOGIC
(dev/)

flat_02_explore_utils.Rmd

Filter data on multiple variables

filter_data()

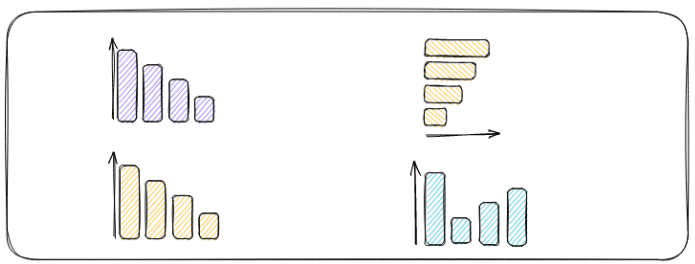
examples

tests

Build a graph with two variables

graph_two_vars()

Which one would you prefer in the application ?

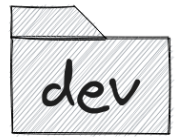


tests

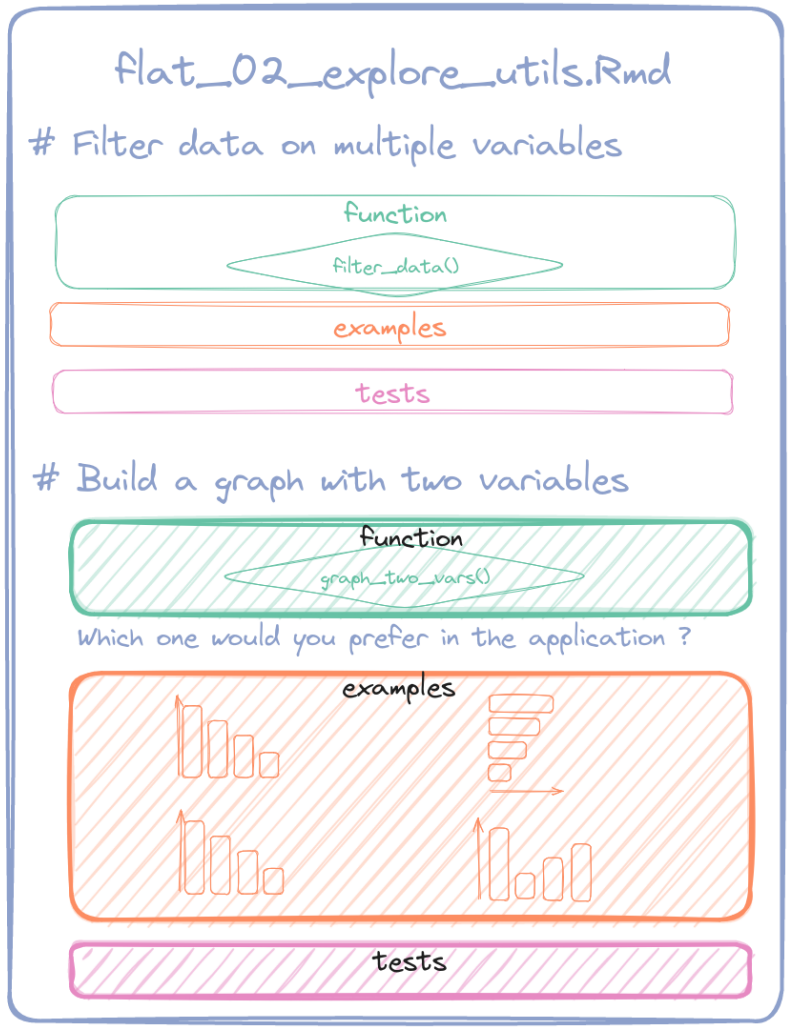
{fusen} needs to distinguish these places to be able to correctly distribute

- You'll need to separate your code according to actions : build, demonstrate, test, document

Name chunks



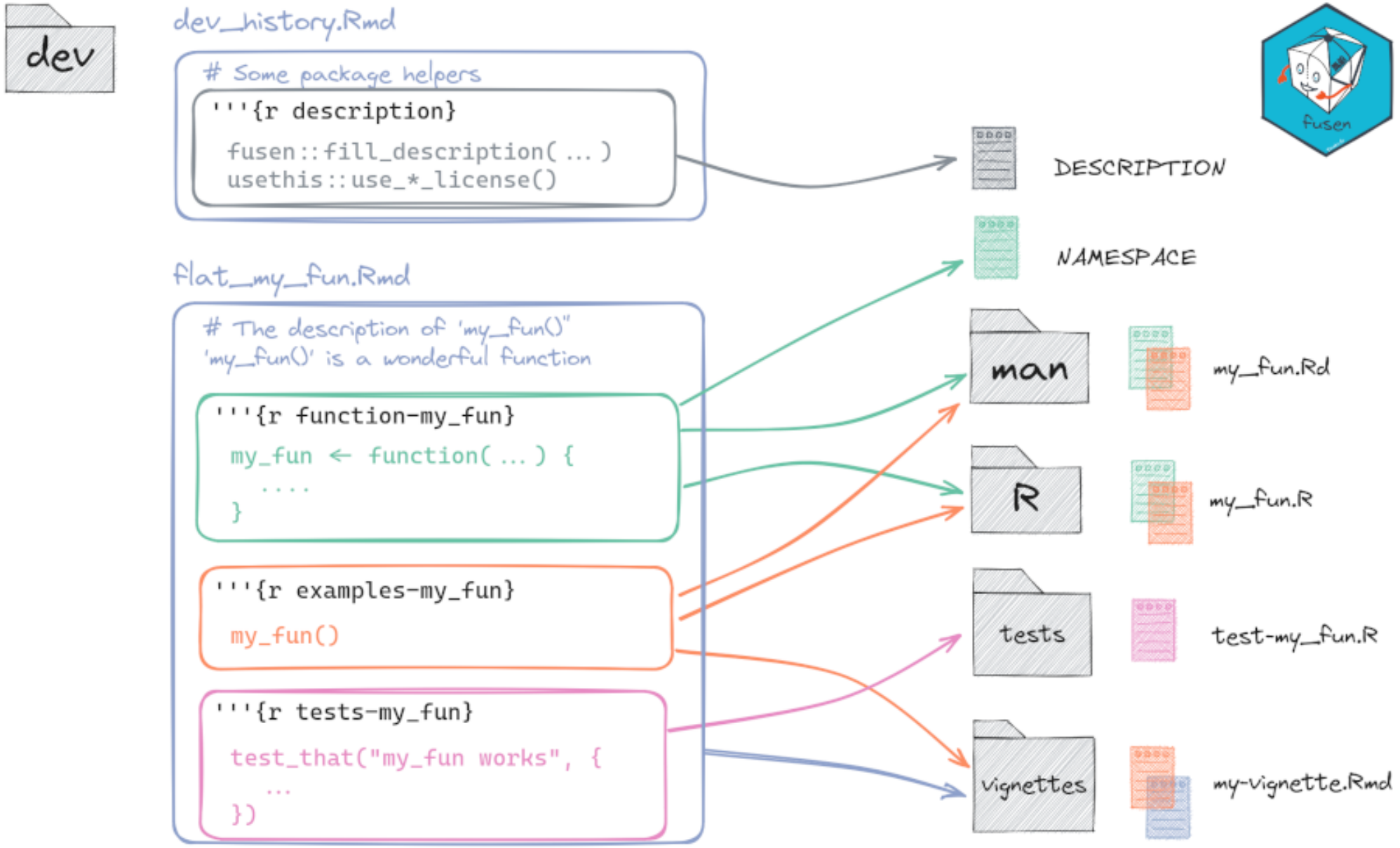
BUSINESS LOGIC



Inflate the package

- Inflate !

```
fusen::inflate(flat_file = "dev/flat_minimal.Rmd")
```

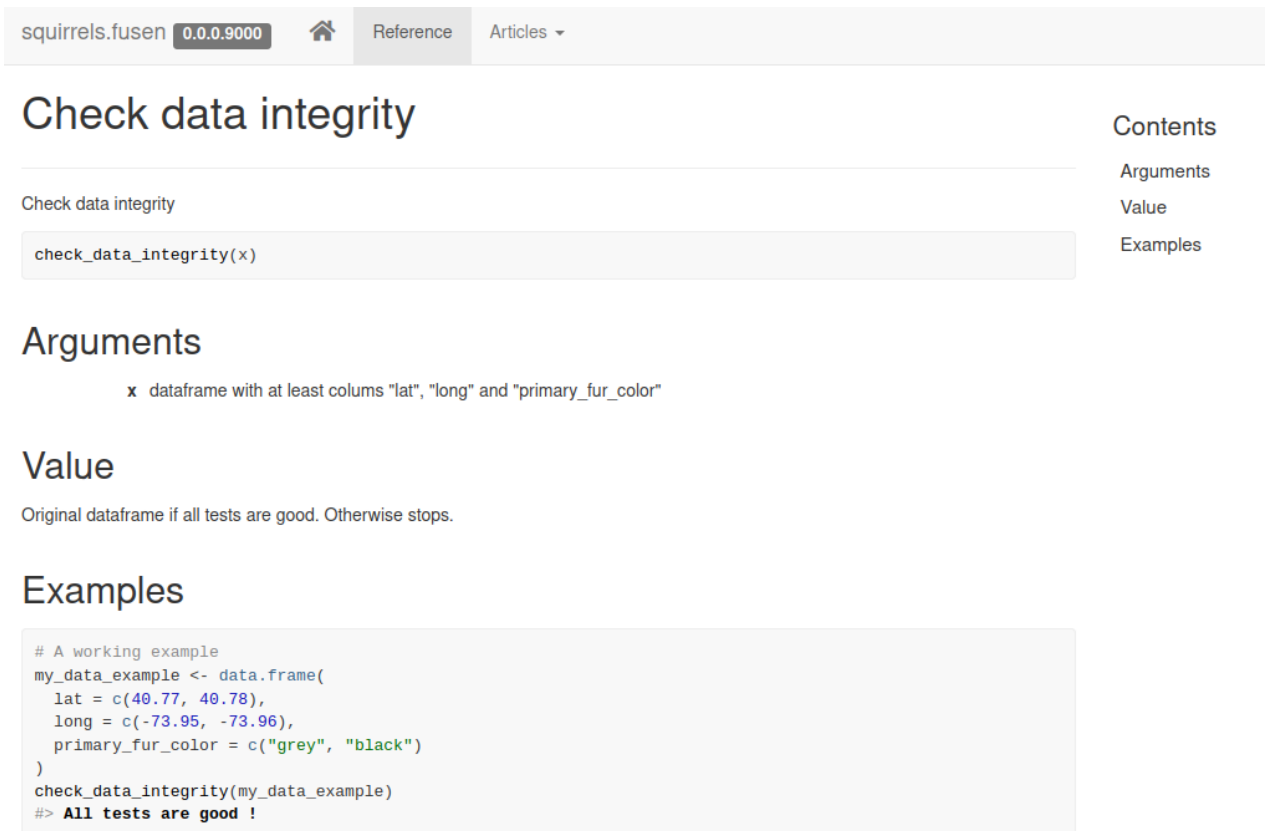


dev
dev_history.Rmd
Some package helpers
```{r description}
fusen::fill_description( ... )
usethis::use_*_license()
```
flat_my_fun.Rmd
The description of 'my_fun()"
'my_fun()' is a wonderful function
```{r function-my_fun}
my_fun <- function( ... ) {
....
}
```
```{r examples-my_fun}
my_fun()
```
```{r tests-my_fun}
test_that("my_fun works", {
...
})
```
fusen
DESCRIPTION
NAMESPACE
man
my_fun.Rd
R
my_fun.R
tests
test-my_fun.R
vignettes
my-vignette.Rmd

Share with your community on GitHub

Share your package along with its documentation

This increases the adoption of your work, and collaboration around it.

```
1  fusen::init_share_on_github()
```



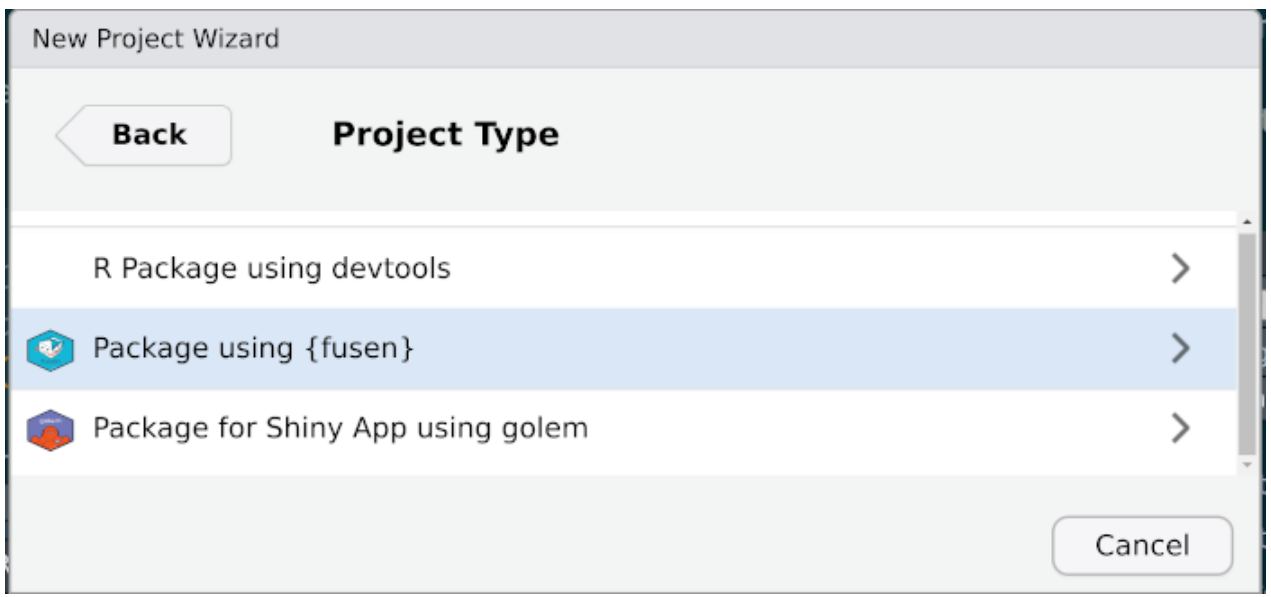
> *e.g. https://github.com/statnmap/fusen.github.0.5.2*
> *https://statnmap.github.io/fusen.github.0.5.2*

Initialisation - the real first step

Use the {fusen} Rmd templates to write your code

- To begin from an existing project:
  - Use the {fusen} template: `fusen::add_flat_template(template = "minimal_package")`
- To begin from scratch:
  - Use the {fusen} package template: `fusen::create_fusen(path = "my.package", template = "minimal")`

Follow the “dev/0-dev_history.Rmd” file to init a first package.



How this answers original problems?

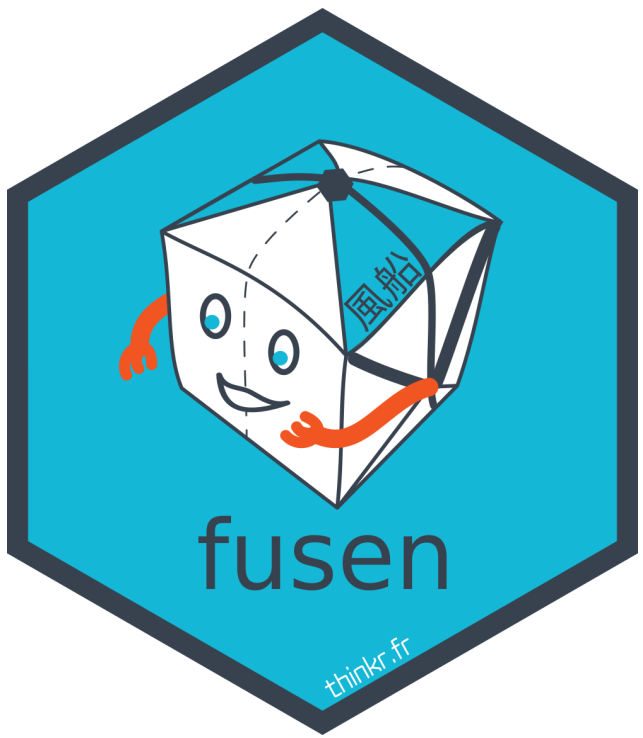
- *A: Damn, I updated `{random.package}` last week, and my [old written] analysis does not work anymore*
  - **Unit tests**
- *B: I protected my project in a Docker / {renv}, and I want this last `{random.package}` functionality but that may break my code*
  - **Unit tests**
- *C: My colleagues modified part of our shared analysis for their data, but it broke mine somewhere*
  - **Unit tests**
- *D: Maybe I need to add some verifications inside my scripts to protect from unfortunate modifications or inputs*
  - **Unit tests**
- *E: My colleagues do not get how to adapt my scripts to their specific case and come ask me a new question every day*
  - **Vignette + examples** + pkgdown
- *F: I got someone else code, but each time I execute it, another package is missing…*
  - **DESCRIPTION**
- *G: I use to copy-paste some lines, but with a small modification*
  - **functions**

What's next?

- Use classic development tools as listed in "dev/01_dev_history.Rmd"
- Use git to track your modifications
  - Insert and run `usethis::use_git()` in "dev/01_dev_history.Rmd"
- Create new "flat_*.Rmd" files for new vignettes and families of functions
  - `fusen::add_flat_template(flat_name = "additional")`
- Use {fusen} with an already existing package
  - https://thinkr-open.github.io/fusen/articles/Maintain-packages-with-fusen.html
- Share your work with your community !
- Share on CRAN ? See https://github.com/ThinkR-open/prepare-for-cran

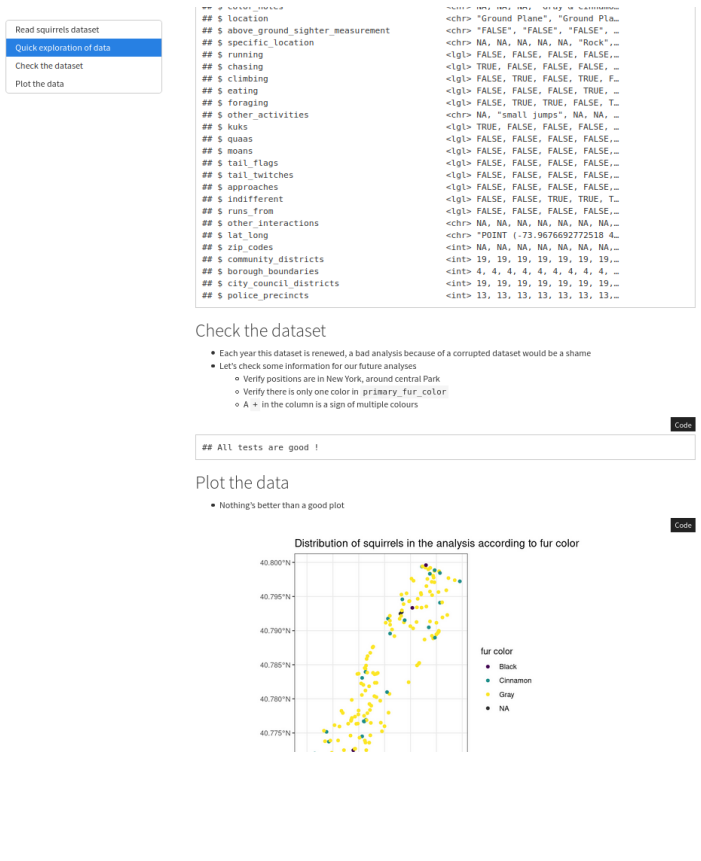
Note: a {fusen} package is a real package. If you remove the "dev/" directory, nobody will see the difference.

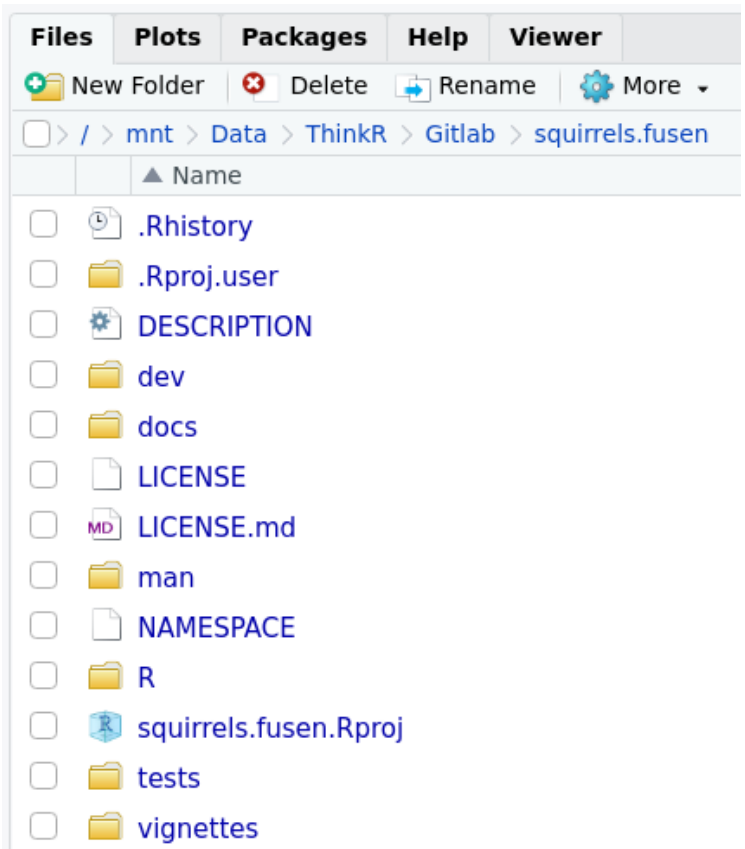
Thank you



DETAILS OF THE STEPS

Let's inflate the squirrels analysis





The squirrels analysis - DESCRIPTION

- To begin from an existing project:
  - Use the {fusen} template: `fusen::add_flat_template(template = "minimal_package")`
- Open the “dev/01-dev_history.Rmd” file
- Fill functions `fusen::fill_description()` and `usethis::use_mit_license()`
- Execute the content of the chunk

````
```{r description}
# Describe your package
fusen::fill_description(
  pkg = here::here(),
  fields = list(
    Title = "Tools to Build the Annual Study of NYC Squirrel data",
    Description = "The NYC Squirrel data includes the locations, fur coloration,
activities, and bizarro behavior of over 2,000 City squirrels. The present package
gives different tools to explore this dataset.",
    `Authors@R` = c(
      person("Sebastien", "Rochette", email = "sebastien@thinkr.fr", role =
c("aut", "cre"), comment = c(ORCID = "0000-0002-1565-9313")),
      person(given = "ThinkR", role = "cph")
    )
  )
)
# Define License with use_*_license()
usethis::use_mit_license("Sébastien Rochette")
```
````

dev_history.Rmd | DESCRIPTION

```
1  Package: squirrels.fusen
2  Version: 0.0.0.9000
3  Title: Tools to Build the Annual Study of NYC Squ
4  Description: The NYC Squirrel data includes the
5  Authors@R: c(person(given = "Sebastien",
6           family = "Rochette",
7           role = c("aut", "cre"),
8           email = "sebastien@thinkr.fr",
9           comment = c(ORCID = "0000-0002-1565-931
10   person(given = "ThinkR",
11          role = "cph"))
12 License: MIT + file LICENSE
13 Encoding: UTF-8
14 LazyData: true
15 Roxygen: list(markdown = TRUE)
16 RoxygenNote: 7.1.1
17
```

> *Here, move “nyc_squirrels_rmd_simple.Rmd” to “dev/”*

The squirrels analysis - functions

- Your script

```
#
# Verify points are in New York around Central Park
all_coords_ok <- all(
  c(
    min(nyc_squirrels[["lat"]]) > 40.76400,
    max(nyc_squirrels[["lat"]]) < 40.80100,
    min(nyc_squirrels[["long"]]) > -73.98300,
    max(nyc_squirrels[["long"]]) < -73.94735
  )
)
if (!all_coords_ok) {stop("Not all data are in Central Park")}

# Verify there is only one color in primary_fur_color.
# A `+` in the column is a sign of multiple colours
if (any(grepl("+", nyc_squirrels[["primary_fur_color"]], fixed = TRUE))) {
  stop("There are multiple colors in some 'primary_fur_color'")
}

message("All tests are good !")
#
```


The squirrels analysis - functions

- Transform as a function and parameterize

```
check_data_integrity <- function(x) {
  # Verify points are in New York around Central Park
  all_coords_ok <- all(
    c(
      min(x[["lat"]]) > 40.76400,
      max(x[["lat"]]) < 40.80100,
      min(x[["long"]]) > -73.98300,
      max(x[["long"]]) < -73.94735
    )
  )
  if (!all_coords_ok) {stop("Not all data are in Central Park")}

  # Verify there is only one color in primary_fur_color.
  # A `+` in the column is a sign of multiple colours
  if (any(grepl("+", x[["primary_fur_color"]], fixed = TRUE))) {
    stop("There are multiple colors in some 'primary_fur_color'")
  }

  message("All tests are good !")
}
```

The squirrels analysis - functions

- Document function, parameters in a chunk named `function`

````
```{r function-1}
```
````

```r
#' Check data integrity
#'
#' @param x dataframe with at least colums "lat", "long" and "primary_fur_color"
#'
#' @return Original dataframe if all tests are good. Otherwise stops.
#' @export
check_data_integrity <- function(x) {
  # Verify points are in New York around Central Park
  all_coords_ok <- all(
    c(
      min(x[["lat"]]) > 40.76400,
      max(x[["lat"]]) < 40.80100,
      min(x[["long"]]) > -73.98300,
      max(x[["long"]]) < -73.94735
    )
  )
  if (!all_coords_ok) {stop("Not all data are in Central Park")}

  # Verify there is only one color in primary_fur_color.
  # A `+` in the column is a sign of multiple colours
  if (any(grepl("+", x[["primary_fur_color"]], fixed = TRUE))) {
    stop("There are multiple colors in some 'primary_fur_color'")
  }

```

The squirrels analysis - examples

- Verify with a reproducible example in a new chunk named `examples`
  - A data.frame with “lat”, “long”, “primary_fur_color”

````
```{r examples-1}
```
````

```r
# A working example
my_data_example <- data.frame(
  lat = c(40.77, 40.78),
  long = c(-73.95, -73.96),
  primary_fur_color = c("grey", "black")
)
check_data_integrity(my_data_example)
```

````
```
````

```
All tests are good !
```
```

The squirrels analysis - tests

- Test on your reproducible examples in a chunk named `tests`

````
```{r tests-1}
```
````

```r
my_data_example <- data.frame(
  lat = c(40.77, 40.78), long = c(-73.95, -73.96),
  primary_fur_color = c("grey", "black")
)
my_data_example_error <- data.frame(
  lat = c(40.77, 40.78), long = c(-73.95, -73.96),
  primary_fur_color = c("grey+blue", "black") # not unique color
)

test_that("check_data_integrity works correctly", {
  expect_message(check_data_integrity(my_data_example), "All tests are good !")
  expect_error(check_data_integrity(my_data_example_error), "multiple colors")
})
```

````
```
````

```
Test passed 🥳
```
```

The squirrels analysis - vignette

- What about the vignette?

The Rmd is the core of the vignette, fill it with information for the users

```
# Check the validity of the entry dataset

Because my dataset may be updated regularly, I need to be sure nothing as changed in its structure. I will build a funct

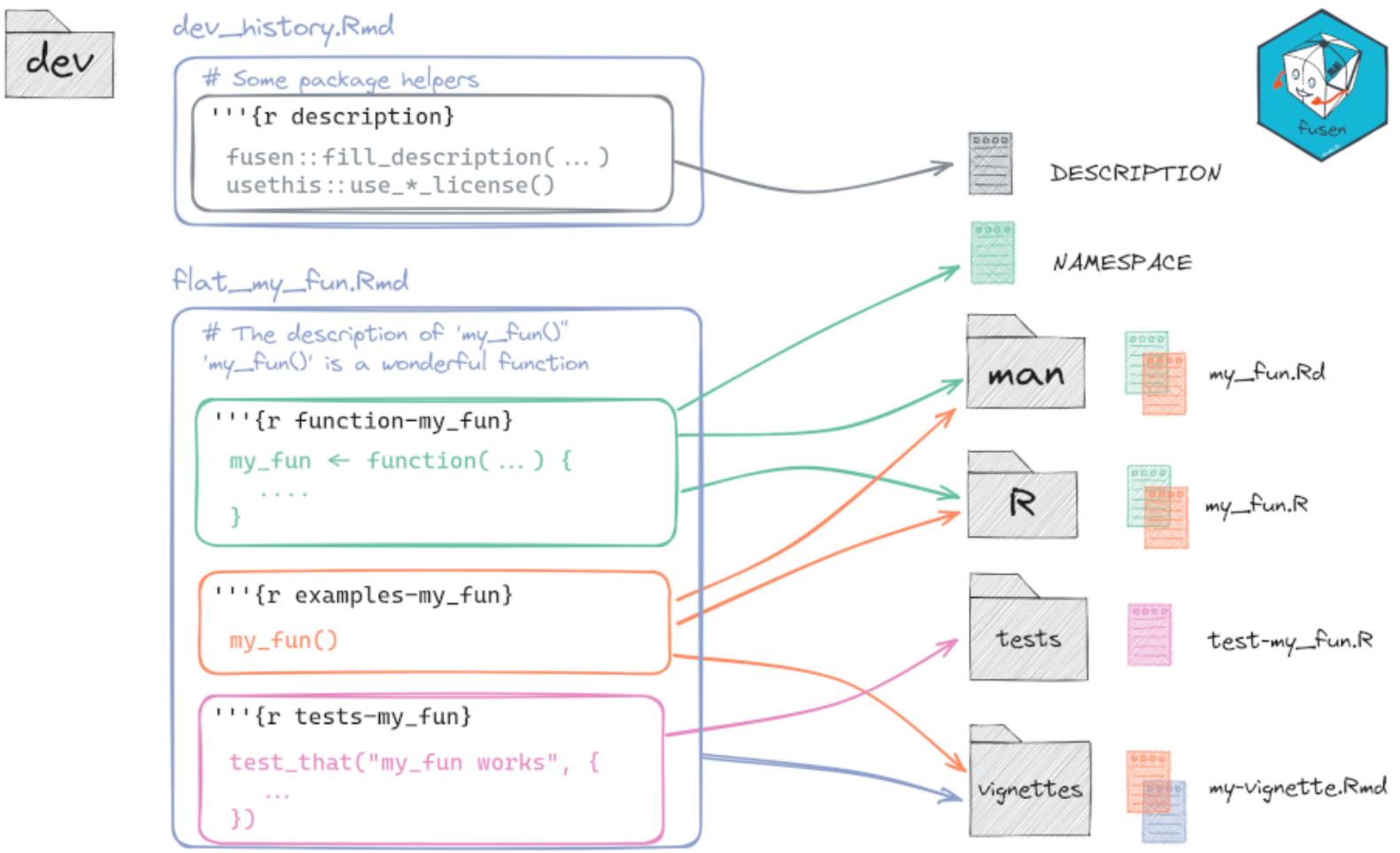
+ Verify positions are in New York, around central Park
+ Verify there is only one color in `primary_fur_color`
  + A `+` in the column is a sign of multiple colours
```

- {fusen} will remove chunks named `description`, `function`, `tests`, `development`
- Chunk named `examples` will stay as they are part of your documentation
  - They are also recycled in `@examples` in the function documentation

The squirrels analysis - the package

- Inflate !

```
1  fusen::inflate(flat_file = "dev/flat_minimal.Rmd")
```

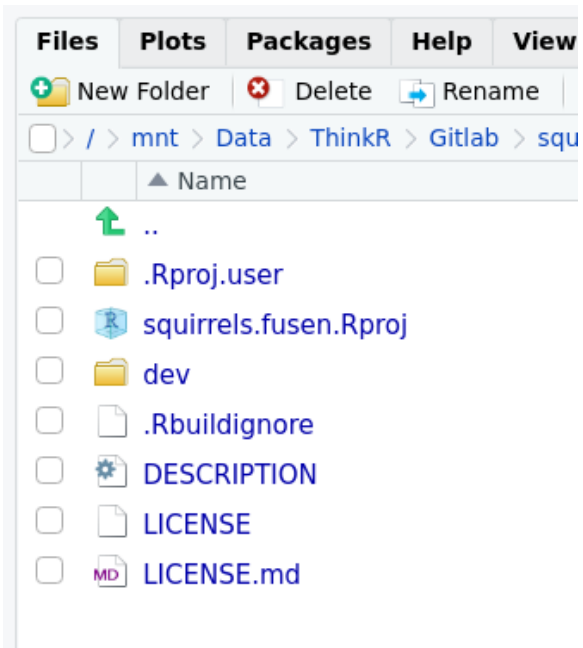


The squirrels analysis - the package

- Inflate !

```
1  fusen::inflate(flat_file = "dev/flat_minimal.Rmd")
```

Before



After