

# Science Reproductible avec un grand

---

Maxime Jaunatre

UMR AMURE | IFREMER

2022/03/17 (updated: 2022-03-04)

# Objectif : bonnes pratiques en R

Background : Apprentissage de R pour l'écologie, traitements de données.

R est construit par et pour des statisticiens mais possède une grande communauté qui partage de plus en plus de "bonne pratiques". Plus proche du software engineering.

On ne va pas parler de stats aujourd'hui.

**photo de moi sur le terrain**

# Reproductibilité ?

Pouvoir reproduire une analyse, une figure ou des données efficacement !

## Principe FAIR :

- **F**indable
- **A**ccessible
- **I**nteroperable
- **R**eusable

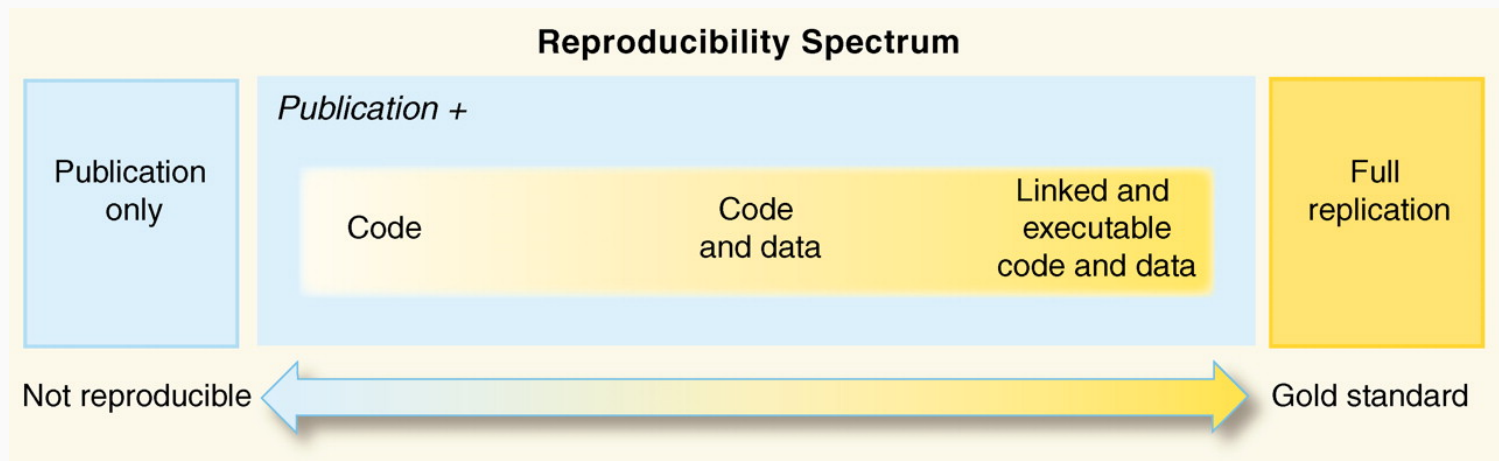


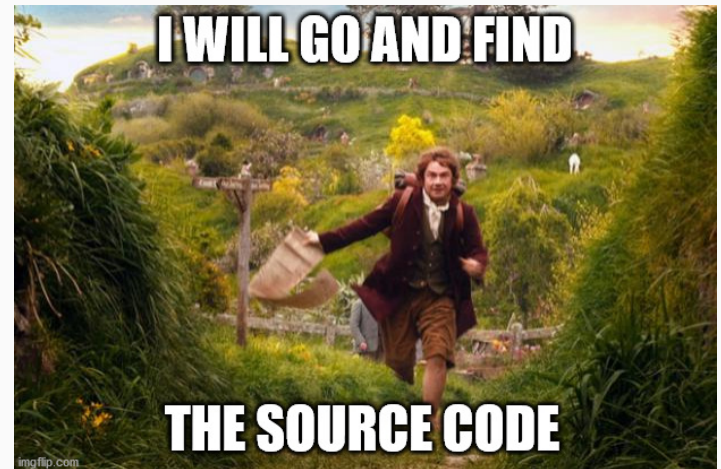
Figure : Peng. "Reproducible Research in Computational Science". In: *Science* (2011).

**Un donnée ou une figure non liées au code qui l'a produit n'est pas reproductible.**

L'emplacement du code doit être explicite. Le trouver ne doit pas être une aventure.

Cela passe par quelques conseils :

- Noms de scripts explicites.  
(on évite les noms comme **test**, **test2**)
- Structure de dossier, **Compendium**
- Disque réseau et/ou en ligne
- Dossier avec noms de **projets**  
(sans espaces et accents !)



- Logiciel de version control (**git**)

```
list.files()[1:4]
```

```
## [1] "CSU_RepRo.html" "CSU_RepRo.pdf" "CSU_RepRo.Rmd" "CSU_RepRo.Rproj"
```

# Pourquoi utiliser un CVS ?

Est-ce que vous avez déjà :

- **Fait un changement de code et voulu revenir en arrière ?**
- **Perdu du code ou une sauvegarde trop ancienne ?**
- Voulu voir la difference entre 2 versions ?
- Voulu vérifier l'historique d'un script ?
- **Voulu travailler sur un script a plusieurs ?**
- **Voulu partager votre code à quelqu'un ?**
- Voulu tester une nouveauté sans modifier du code déjà utile ?

Si oui, et dans plein de cas, un système de version control aurait pu vous simplifier la vie.

# Pourquoi utiliser un CVS ?

"FINAL".doc



FINAL.doc!



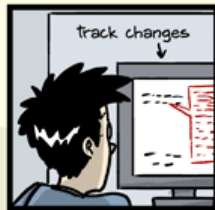
FINAL\_rev.2.doc



FINAL\_rev.6.COMMENTS.doc



FINAL\_rev.8.comments5.  
CORRECTIONS.doc



FINAL\_rev.18.comments7.  
corrections9.MORE.30.doc



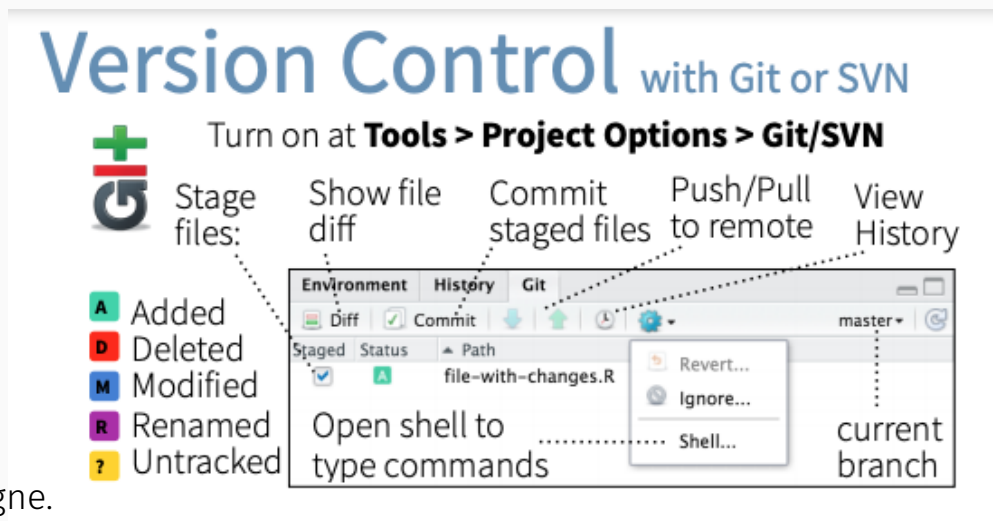
FINAL\_rev.22.comments49.  
corrections.10.#@\$%WHYDID  
ICOMETOGRADSCHOOL?????.doc

JORGE CHAN © 2012

# Git : comment ça marche ?

## 3 commandes à retenir : commit, push, and pull

- pull: met à jour le projet local avec la dernière version du projet\*
- commit: enregistre une snapshot du code à un certain point temporel.  
(permet d'associer du texte pour expliquer les modifications)
- push: met en commun les modifications locales avec le projet principal\*



[\*] Si hébergé en ligne.

# Git : comment ça marche ?

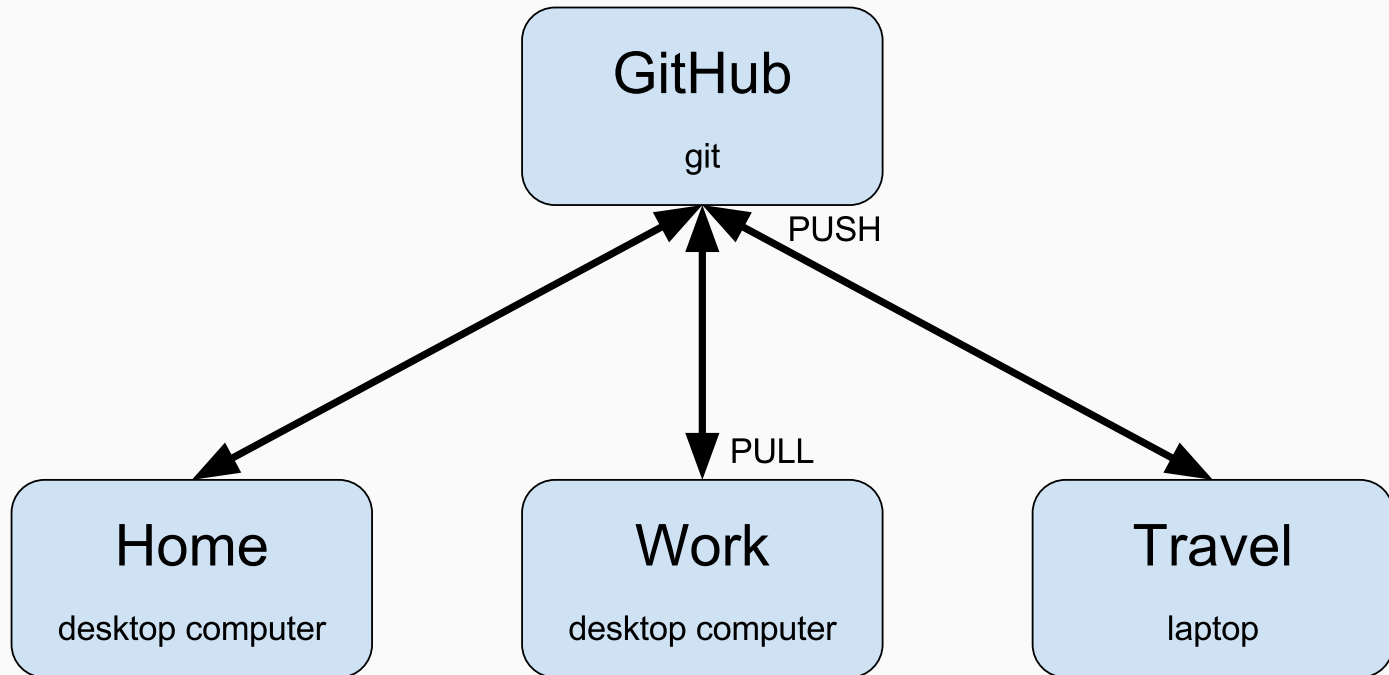
	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFT	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.



# Git : comment ça marche ?

Sauvegarder en ligne ?



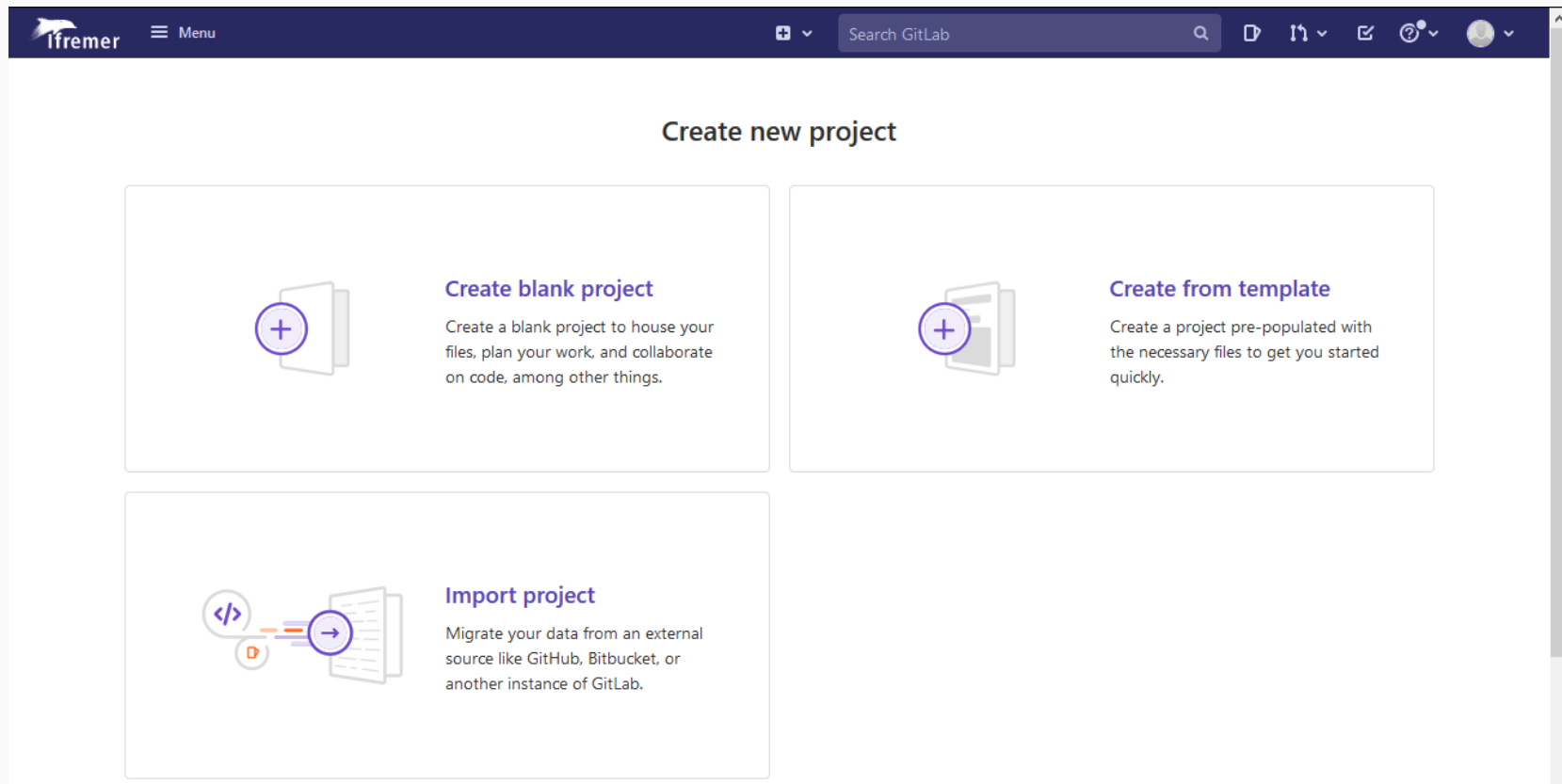
# Git : comment ça marche ?

The screenshot shows the GitLab interface for a user named 'lfremer'. The top navigation bar includes the 'lfremer' profile, a 'Menu' icon, a 'Search GitLab' bar, and various utility icons. The main section is titled 'Projects' and features a 'New project' button. Below this, there are tabs for 'Your projects' (3), 'Starred projects' (0), and 'Explore projects'. A search bar 'Filter by name...' and a dropdown 'Most stars' are also present. The 'All' tab is selected, showing a list of three projects:


Project Name	Owner/Maintainer	Stars	Forks	Issues	Discussions	Updated
IAM / IAM clean CDD Maxime Jaunatre in AMURE UMR Cleaning, Documentation and Compilation of the Impact Assessment bio...	Owner	0	0	0	0	Updated 15 hours ago
IAM / IAM Impact Assessment bio-economic Model for fisheries management (IAM) package	Owner	0	0	0	0	Updated 1 day ago
JAUNATRE / IslaNublar Experimental repository for C++ in R package	Maintainer	0	0	0	0	Updated 1 week ago



**Connection avec login extranet**, possibilité de partager à des membres externes.






# Git : comment ça marche ?




# Git : comment ça marche ?

Menu

 Search GitLab 



## Create blank project


Create a blank project to house your files, plan your work, and collaborate on code, among other things.

New project > Create blank project

---

**Project name**

**Project URL**





**Project slug**


Want to house several dependent projects under the same namespace? [Create a group.](#)


**Project description (optional)**

```
# Project
This is a short project in R language to show how Git can work.
This is about using Gitlab.ifremer.fr
```

**Visibility Level** 

☒  **Private**  
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

☐  **Internal**

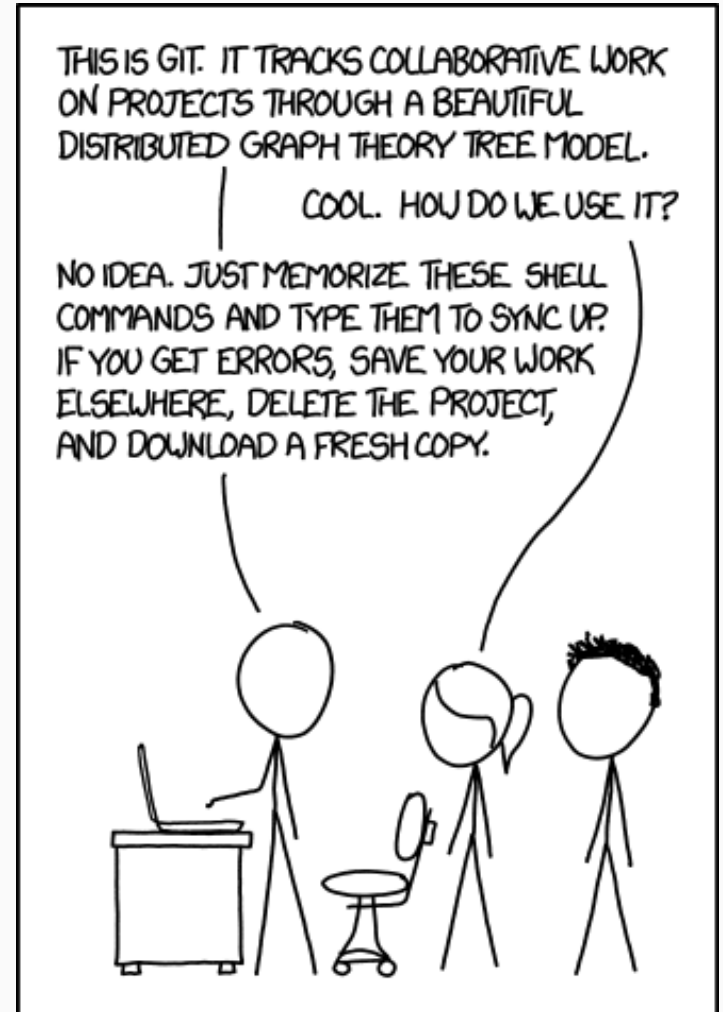
☐  **Public**

☒ **Initialize repository with a README**  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

# Ressources Git

- Tutoriel ThinkR : R and Git
- Advance R 📱, H. Wickham
- Happy Git and GitHub for the userR
- Git cheatsheet
- Quand ça part en vrille
- Réparer une erreur
- Créer une nouvelle branche avec git et merge des branches

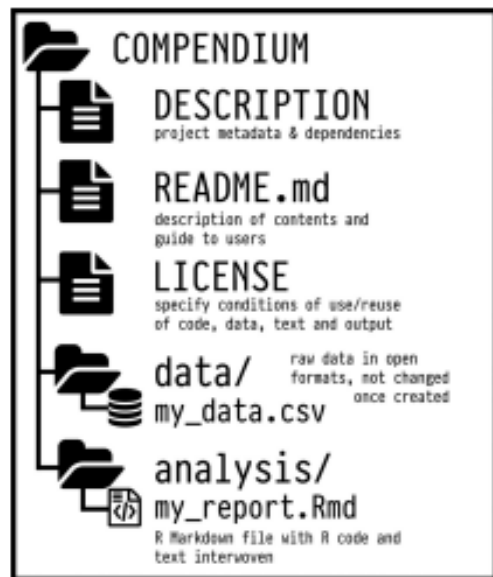
xkcd comics, CC BY-NC 2.5 license



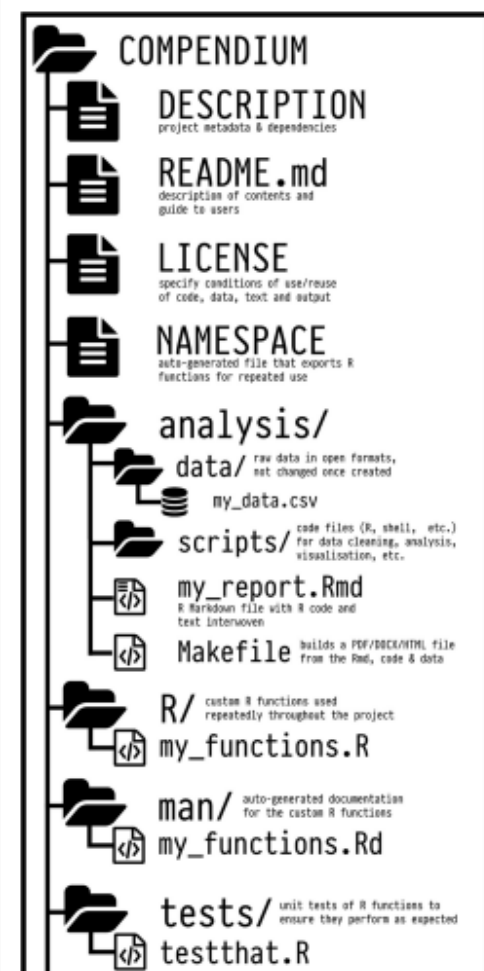
# Accessible

## COMPENDIUM

Structurer les dossiers en packages R.

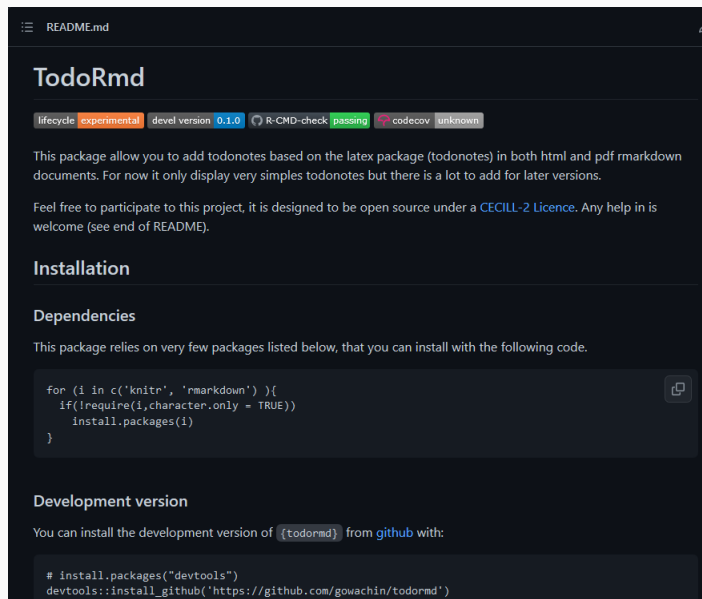


Marwick et al. "Packaging Data Analytical Work Reproducibly Using R (and Friends)". In: *The American Statistician* (2017).



# Accessible

- **README** : entrée classique de documentation.
- Aide à l'installation et l'utilisation
- Pour ajouter à un projet :  
`usethis::use_readme_md()`



- **LICENCE** : sans licence, un code est théoriquement inutilisable.  
(**MIT, GNU GPL, CC**)

Dicte les droits d'utilisation, de copie, de modification d'un code.

Loi Numérique n°2016-1321 impose une licence libre pour tout logiciel produit avec des fonds publics.



A nuancer avec les questions de publications, de tutelles etc.

# Accessible

## Projet = Package

- **DESCRIPTION** :

```
Package: todormd
Type: Package
Title: Using todonotes in rmarkdown package
Version: 0.1.0
Authors@R: c(
  person('Maxime', 'Jaunatre',
    email = "maxime.jaunatre@yahoo.fr",
    role = c('aut', 'cre'))
)
Description: Personnel project number x
License: CeCILL-2
Encoding: UTF-8
Imports:
  knitr,
  rmarkdown
RoxygenNote: 7.1.1
Suggests:
  testthat (≥ 3.0.0)
```

- **NEWS.md** : Fichier de suivis des mises à jour d'un projet.

Permet d'informer sur un changement majeur ou des nouveautés. Important d'avoir un cycle de version avec :

`0.1.1 > 0.1.0`

Why and how maintain a NEWS file for your R package?

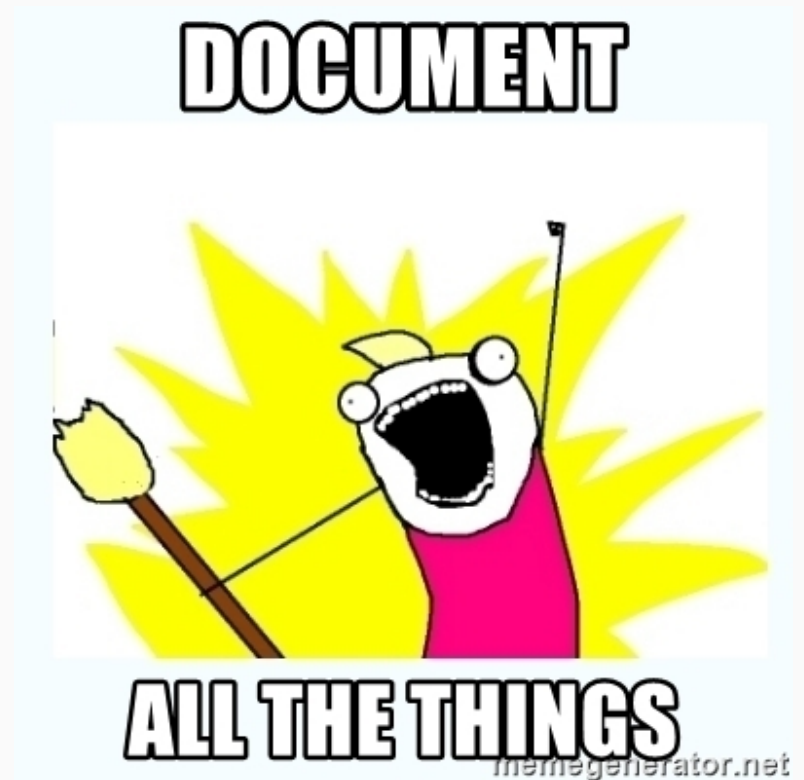
- Pour ajouter une dépendances au projet :  
`devtools::use_package("tidyr")`



## Documentation i

- Rédiger en amont ( {Roxygen2} )

```
#' half_life
#'
#' compute half_life percentage.
#'
#' @param period period in minute
#' @param time time elapsed (min)
#'
#' @return percentage.
#' @author M. Jaunatre <mail>
#'
#' @examples
#' half_life(20, 20)
#' half_life(20, 40)
#'
half_life ← function(period, time){
  return((1 - 2^(-time/period)) * 100)
}
```



## Documentation i

### Forme longue de documentation

- {Rmarkdown} ! Permet de mélanger texte (**md**, *L<sup>A</sup>T<sub>E</sub>X*) et code **R**.
- Vignettes pour les packages
- Rédaction d'articles -> {rticles}
- Diapos -> {xaringan}
- Livre -> {bookdown}
- Site web -> {pkgdown}



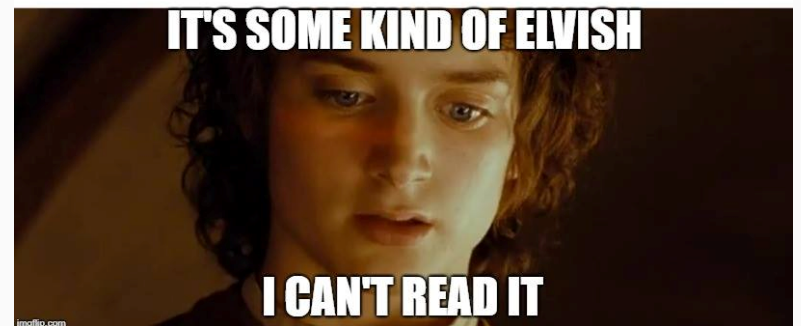
```
CSU_RepRo - main - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
CSU_RepRo.Rmd rmd2pdf.R
Find Next Prev All Replace Replace All
In selection Match case Whole word Regex Wrap
336 ]
337
338 ---
339
340 .pull-left[
341 ## Documentation &#x2139;
342
343 - Rédiger en amont ('{Roxygen2}')
344
345 {r, echo = TRUE}
346 #' half_life
347 #'
348 #' compute half_life percentage
349 #'
350 #' @param period period in minute
351 #' @param time time elapsed (min)
352 #'
353 #' @return percentage
354 #' @author M. Lapointe <mail>
355 #'
356 #' @examples
357 #' half_life(20, 20)
358 #' half_life(20, 40)
359 #'
360 half_life <- function(period, time){
361   return((1 - 2^(-time/period)) * 100)
362 }
363
364 ]
365
366 .pull-right[
367 {r, out.width="100%"}
368 .knitr::include_graphics("CSU_RepRo_files/figure-html/document-all-the-things.jpg")
369
370 ]
371
372 ---
```

## Lisibilité

```
# Title #####  
# Author, date, contact etc  
# Description  
  
## Depends #####  
library(IAM)  
# devtools::install_github(  
#   "github.com/gowachin/todormd"  
# )  
  
# sources  
sources("R/hello_world.R")  
# functions  
foo ← function() {cat("Don't panic !")}  
  
## Datasets #####  
read.csv(file = "raw_data/sole.csv")  
  
## Edit Dataset #####  
  
## Plots #####  
## Export #####
```

- **DRY** (Don't Repeat Yourself)
- **KISS** (Keep it Simple, Stupid)  
plein de petits fichiers > script infini
- `{cleanR}`: nettoyer les lignes inutiles.
- `{styleR}`: reformate le code.

When you trying to look at  
the code you wrote a month ago



## Lisibilité

```
#                               Bad                               #
if(T){print(10)}
T ← FALSE
mean ← function(x) sum(x)

if (y < 0 && debug)
  message("y is negative")
  stop("error for in y test")

function_with_many_argument("that", many
function_with_many_argument(x = "that",
  many)

x = 5
i = 0 ; y=12
```

```
#                               Nice                               #
if(TRUE){print(10)} # Full logical
testing ← FALSE # name conflict
use_sum ← function(x) sum(x)

if (y < 0 && debug) {
  message("y is negative")
  stop("error for in y test")
} # Braces

function_with_many_argument(
  x = "that",
  y = many # name args
) # line limit = 80

x ← 5 # arrow
i ← 0
y ← 12
```

**"Code needs a lot of whitespace. That is how it breathes."** Roger Peng

# Interopérable

**Un projet ne doit pas dépendre de l'ordinateur qu'il utilise** -> Portabilité

Cela signifie l'oubli de ces commandes maudites :

```
setwd("C:/home/maxime/Documents/Projet magnifique/mouette/analyse/bob/modèle/test/")
```

```
rm(list = ls()) # → Supprime uniquement les objets "utilisateurs"  
# → Ne supprime pas des dépendances chargées.  
# → Indique une session R ouverte depuis 30 ans  
# → Vide l'environnement des copains qui veulent aider  
# → Participe à la disparition des bébés phoques
```

Si votre script possède ces lignes, **Jenny Bryan viendra bruler** votre ordinateur. 🔥

Solution ? Utiliser les **Projets Rstudio**

*A noter l'espace, et l'accent dans ce chemin !*


# Interopérable

File → New Project → New Directory

New Project

Back

Create New Project



Directory name:

my\_awesome\_project

Create project as subdirectory of:

~/Documents/workflows

☐ Create a git repository

☐ Use packrat with this project

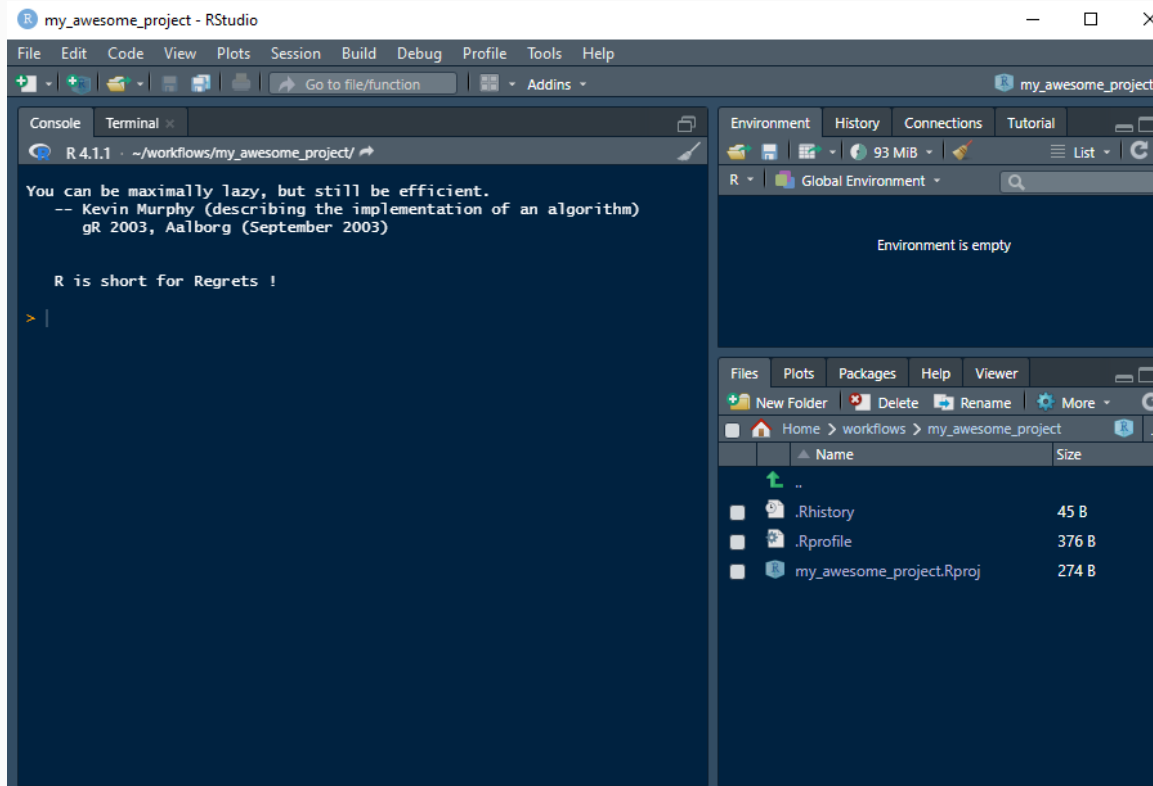
☒ Open in new session

Create Project

Cancel

# Interopérable

Dans un project, tout les chemins sont basés sur la racine du projet (où se situe `.Rproj`)



Désactiver la sauvegarde automatique !

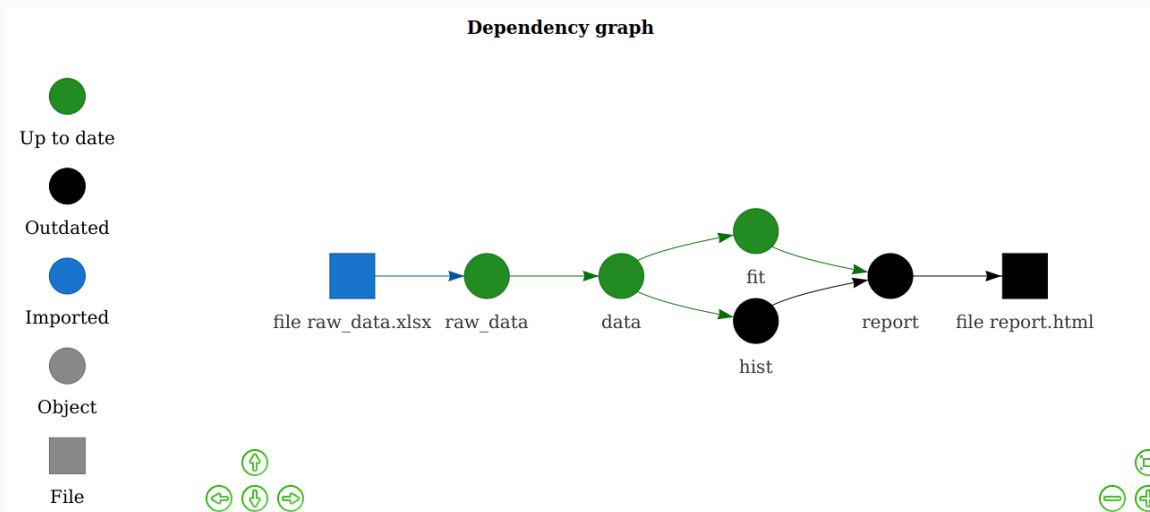
Tools → Global options → General → un-tick "always save history"

## Données

**Un script ne doit dépendre que des données présentes dans son projet.**

- Eviter les `.RData` ! Ce format de fichier montre que vos données sont trop complexes et que le script qui les produit n'est pas clair.

Seule exception : les données de type cache : permet de faire tourner les scripts plus facilement. Mais encore une fois des outils existent pour éviter cela comme `{drake}`, qui permet de refaire tourner que les parties modifiées ou dépendantes d'un script.







Parfois les données sont complexes et non distribuables. Dans ce cas, prévoir le minimum pour montrer que le code marche.

- Avoir un **exemple** simplifié pour chaque script.

Exemple : ici la fonction prend un object en entrée...mais quel format doit-il avoir ?

```
rm_dups <- function(data){  
  data <- data[! duplicated(data$x), ]  
  data  
}
```

```
df <- data.frame(x = c(0,1,1,0),  
                 y = c(2,1,0,0))  
rm_dups(df)
```

```
##    x y  
## 1 0 2  
## 2 1 1
```

Fournir un exemple est un très bon début de documentation.

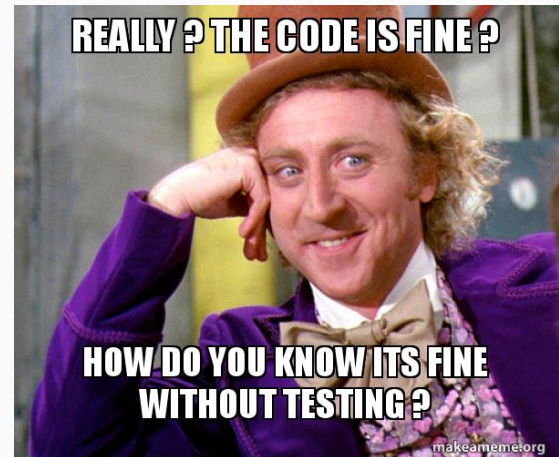
Exemple = niveau 0 du test !



## Test !

Nous testons tous nos codes à la main, et ça marche sur le coup...mais 2 semaines après ?

- Ecrire des tests unitaires qui font tourner le code tout seul.  
-> Dossier `test/`
- `usethis::use_testthat()` charge automatiquement `{testthat}`



**Il vaut mieux prévenir que guérir !**



## Test !

Utiliser `{usethis}` pour simplifier la vie !

- `usethis::use_test("add")`

```
tests
├── testthat
│   └── test-add.R
└── testthat.R
```

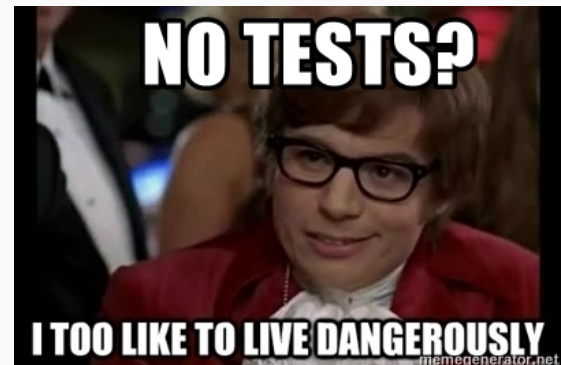
Rédiger des tests n'est jamais agréable  
mais sauve du temps plus tard.

**Retour rapide** lors de la modification du  
code + assurance de rien casser.

```
library(testthat)
add ← function(y,x){
  x+y
}

test_that("add works", {
  expect_equal(add(39, 3), 42)
})
```

```
## Test passed
```



# Trouver de l'aide pour R 🙏

## RTFM\*

- utiliser les aides `help(sum)` ou `?sum`
- lire les **manuels** de packages `ggplot2`
- avoir les **cheatsheet** quelque part !

## Stack overflow 📖

## Issue Github 🐙

## Communauté francophone de R :

- [frrrenchies](#) : list de doc fr
- [slack grrr](#) : question, news, jobs...

## Livres et blogs

- [Advance R](#), Hadley Wickham
- [Git et Rstudio](#), ThinkR
- [Project-oriented workflow](#), Jenny Bryan


## Conférences

- [Code smells and feels](#), Jenny Bryan
- Toutes les conférences UseRs 🎥

[\*] Read The Fucking Manual.

# Merci! Des questions ?

ajouter le lien vers les slides

 [Gowachin](#)    [Gowachin](#)

Slides created via the R package [xaringan](#).