Structuring projects

Francisco Rodriguez-Sanchez

https://frodriguezsanchez.net

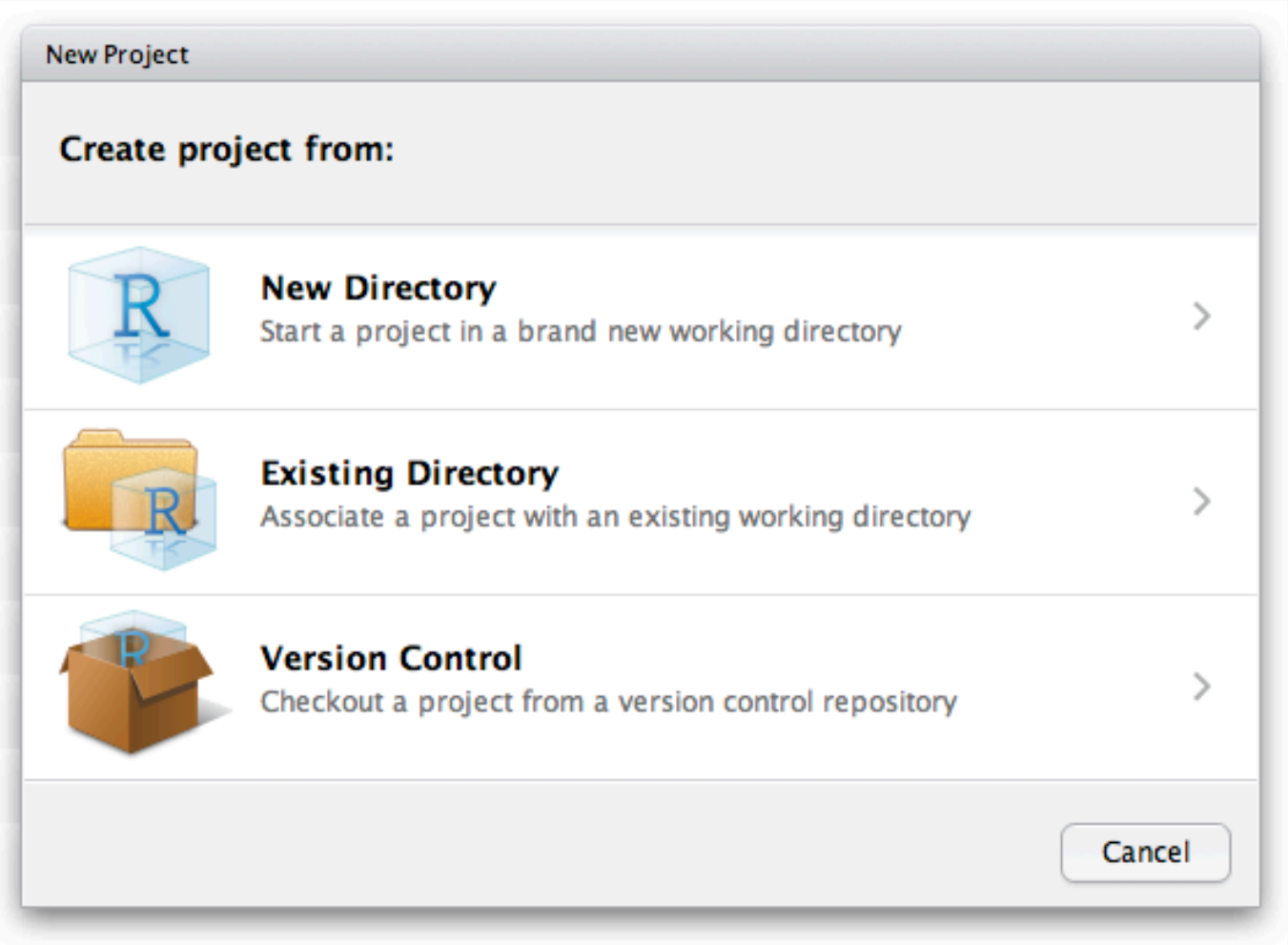
One Project = One Folder

```
myproject
  |
  |- data
  |
  |- code
  |
  |- output (figures etc)
  |
  |- manuscript
```

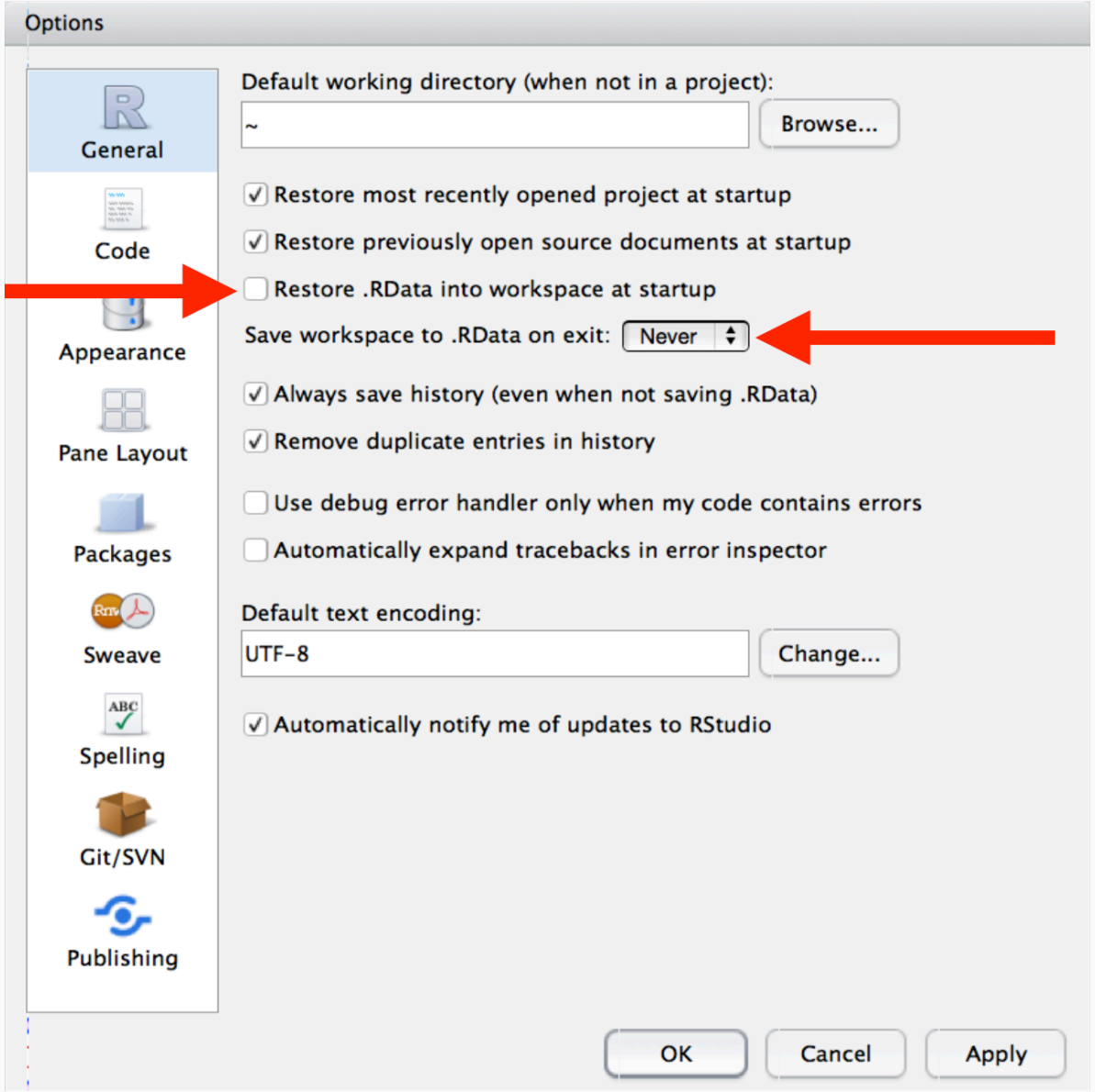
Project-Oriented Workflow: advantages

- Self-contained
- Easy to navigate (file paths)
- Easy to share

Rstudio projects

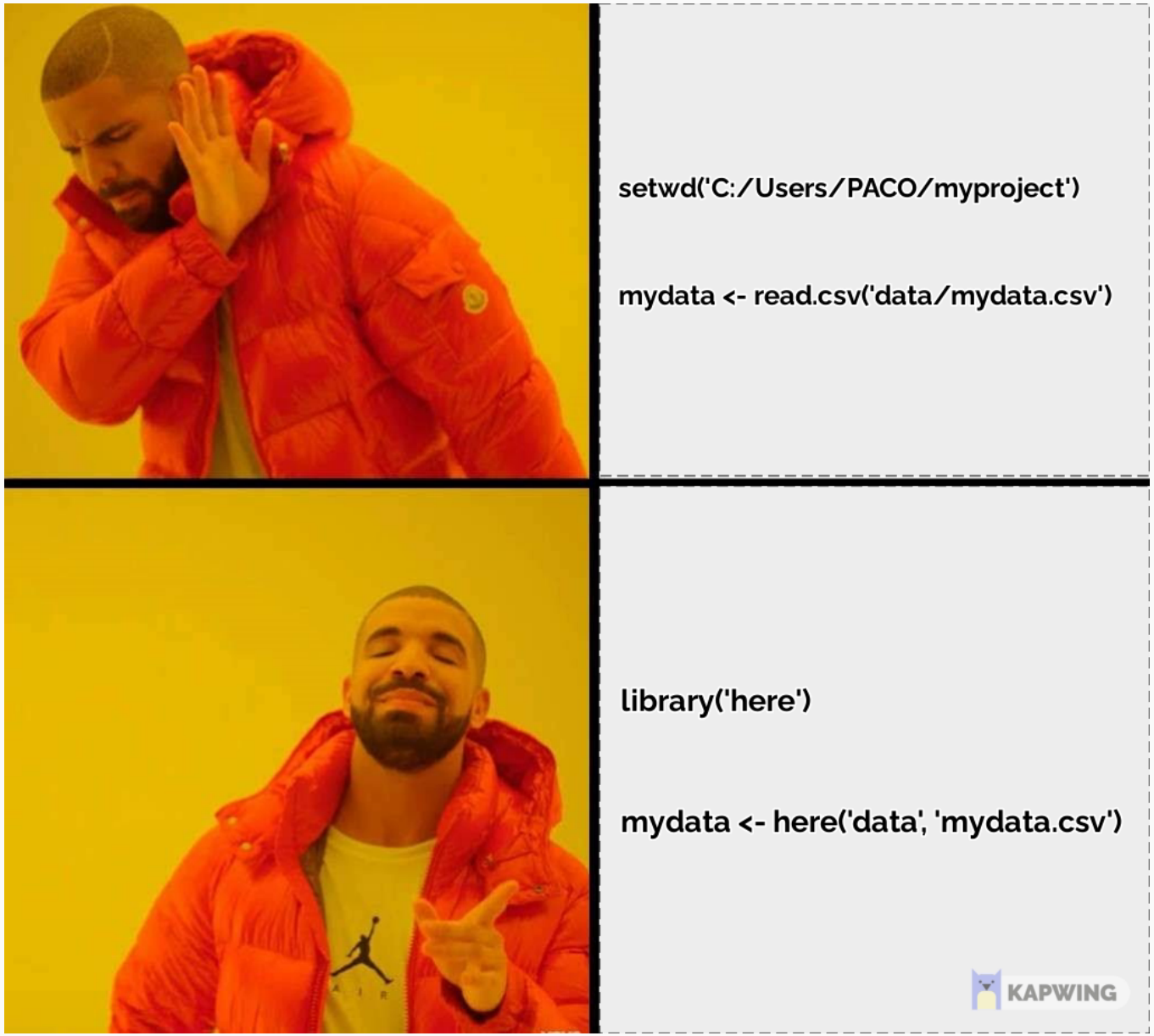


Avoid saving workspace



https://rstats.wtf

Use `here` for file paths



`fertile` package: real-time feedback on reproducibility

```r
library('fertile')

setwd("C:/Users/FRS")
```

Error: setwd() is likely to break reproducibility. Use here::here() instead.

https://github.com/baumer-lab/fertile

Structuring projects: guidelines

Guidelines for structuring projects

- All files in **same directory**

Noble 2009, Rodriguez-Sanchez et al 2016, Wilson et al 2017

Guidelines for structuring projects

- All files in **same directory**
- **Raw data** separate from **clean data**

Noble 2009, Rodriguez-Sanchez et al 2016, Wilson et al 2017

Guidelines for structuring projects

- All files in **same directory**
- **Raw data** separate from **clean data**
- **Modular code** (functions)

Noble 2009, Rodriguez-Sanchez et al 2016, Wilson et al 2017

Guidelines for structuring projects

- All files in **same directory**
- **Raw data** separate from **clean data**
- **Modular code** (functions)
- **Output disposable & separate** from code

Noble 2009, Rodriguez-Sanchez et al 2016, Wilson et al 2017

Guidelines for structuring projects

- All files in **same directory**
- **Raw data** separate from **clean data**
- **Modular code** (functions)
- **Output disposable & separate** from code
- `makefile` runs analyses in **appropriate order**

Noble 2009, Rodriguez-Sanchez et al 2016, Wilson et al 2017

Guidelines for structuring projects

- All files in **same directory**
- **Raw data** separate from **clean data**
- **Modular code** (functions)
- **Output disposable & separate** from code
- `makefile` runs analyses in **appropriate order**
- **Software dependencies** under control

Noble 2009, Rodriguez-Sanchez et al 2016, Wilson et al 2017

Guidelines for structuring projects

- All files in **same directory**
- **Raw data** separate from **clean data**
- **Modular code** (functions)
- **Output disposable & separate** from code
- `makefile` runs analyses in **appropriate order**
- **Software dependencies** under control
- README

Noble 2009, Rodriguez-Sanchez et al 2016, Wilson et al 2017

Guidelines for structuring projects

- All files in **same directory**
- **Raw data** separate from **clean data**
- **Modular code** (functions)
- **Output disposable & separate** from code
- `makefile` runs analyses in **appropriate order**
- **Software dependencies** under control
- README
- License

Noble 2009, Rodriguez-Sanchez et al 2016, Wilson et al 2017

Project organisation example

```
- data
  - data-raw
  - data-clean

- code

- output (figures etc)

- manuscript

- README
- License
- Makefile
```

README file

- What
- Who
- How
- Licence
- Citation
- etc

README.md

pandanusisotopes

launch binder

This repository contains the data and code for our paper:

> Florin, A. et al. (2020). *Palaeoprecipitation data from Madjedbebe, northern Australia: A novel proxy from ancient pandanus.*

How to cite

Please cite this compendium as:

> Marwick, B., A. Florin et al., (2020). *Compendium of R code and data for Palaeoprecipitation data from Madjedbebe, northern Australia: A novel proxy from ancient pandanus.* Accessed 16 Oct 2020. Online at https://doi.org/xxx/xxx

How to download

You can download the compendium as a zip from from this URL: https://github.com/benmarwick/pandanusisotopes/archive/master.zip

Licenses

Text and figures : CC-BY-4.0

Code : See the DESCRIPTION file

Data : CC-0 attribution requested in reuse

Document your data

```r
library("dataspice")
create_spice()   # create CSV templates for metadata

edit_creators()  # open Shiny apps to edit the CSVs
prep_access()
edit_access()
prep_attributes()
edit_attributes()
edit_biblio()

write_spice()  # write machine-readable metadata

build_site()  # build human-readable metadata report
```

Write modular code

Break up scripts

```
prepare_data.R

run_analysis.R

make_figures.R
```

(and `makefile` will run them in the right order)

`makefile` runs code in appropriate order

`makefile.R`

```r
source("prepare_data.R")

source("run_analysis.R")

source("make_figures.R")
```

Don't Repeat Yourself (DRY)

```r
dataset %>%
  filter(species == "Laurus nobilis") %>%
  ggplot() +
  geom_point(aes(x, y))

dataset %>%
  filter(species == "Laurus azorica") %>%
  ggplot() +
  geom_point(aes(x, y))
```

Don’t Repeat Yourself

Write functions (documented + tested)

```
plot_species <- function(sp, data) {
  data %>%
    filter(species == sp) %>%
    ggplot() +
    geom_point(aes(x, y))
}
```

Don’t Repeat Yourself

Use functions

```
plot_species(sp = "Laurus nobilis", dataset)

plot_species(sp = "Laurus azorica", dataset)
```

Don’t Repeat Yourself

Use for loops

```
for (i in species) {
  plot_species(sp = i, dataset)
}
```

Don’t Repeat Yourself

Good ol’ `lapply`

```
lapply(species, plot_species, data = dataset)
```

Don’t Repeat Yourself

```r
library("purrr")

map(species, plot_species, data = dataset)
```

Comment your code

Why rather than **What**

```
## Response is not linear, so fit gam rather than lm

model.height <- gam(height ~ s(diameter), data = trees)
```

Use meaningful names for objects

```r
m1 <- lm(height ~ diameter, data = trees)
m2 <- gam(height ~ s(diameter), data = trees)
```

Use meaningful names for objects

```r
m1 <- lm(height ~ diameter, data = trees)
m2 <- gam(height ~ s(diameter), data = trees)
```

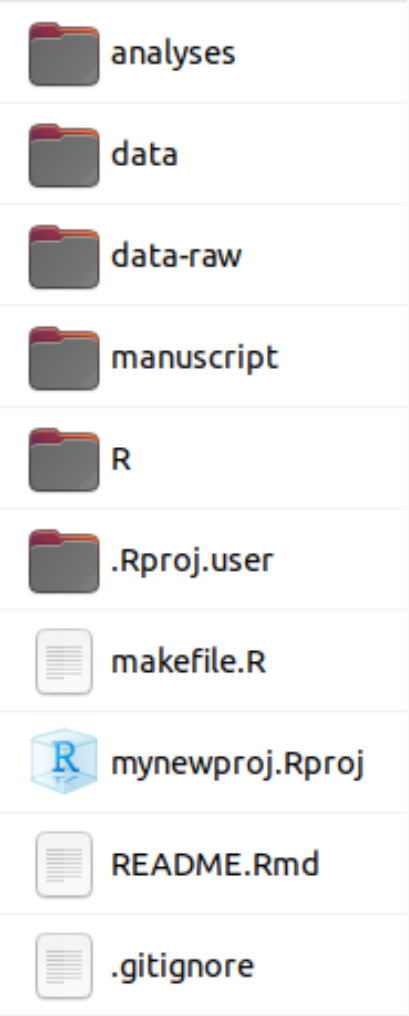
```r
model.linear <- lm(height ~ diameter, data = trees)
model.gam <- gam(height ~ s(diameter), data = trees)
```

Project templates

Automatic project creation with `template`

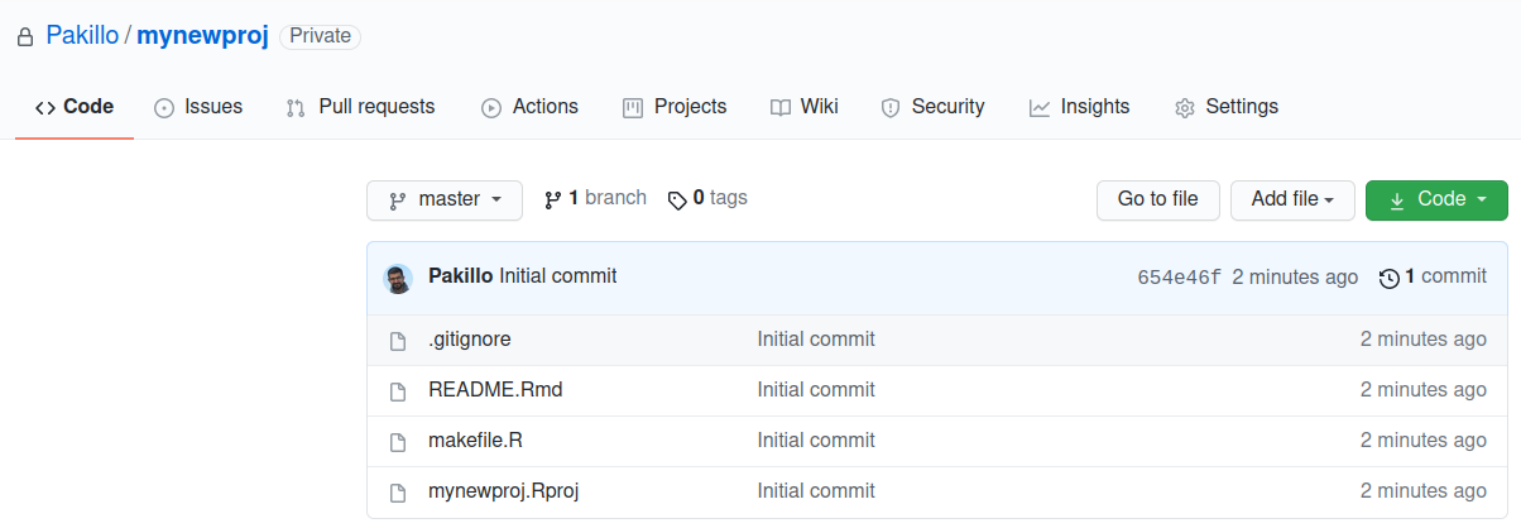
```r
library('template')
```

```r
new_project("mynewproj",
            package = FALSE)
```



`template`: New projects also on GitHub

```r
new_project("mynewproj",
            package = FALSE,
            github = TRUE)
```

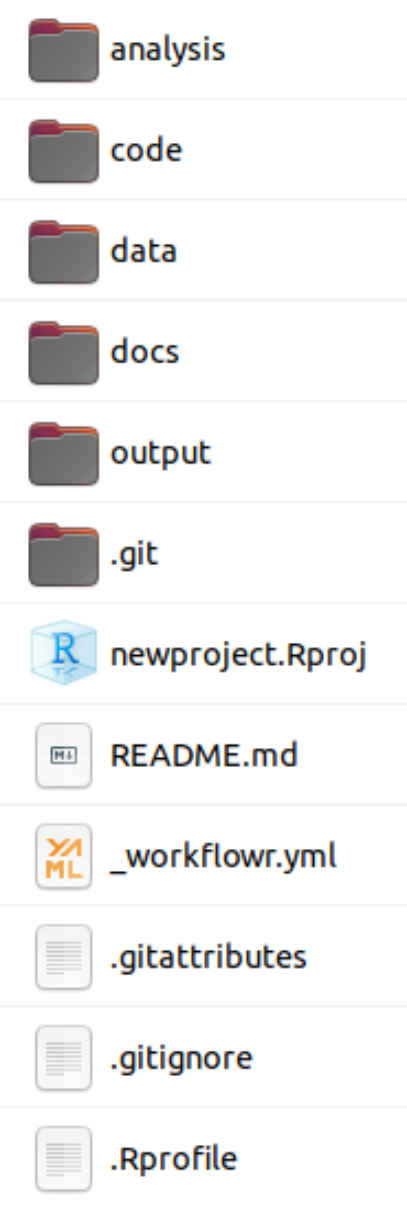


`workflowr`: reproducible projects with website

`wflow_start` creates project scaffolding

```r
library('workflowr')

wflow_start("newproject")
```



wflow_open starts new analysis

```r
wflow_open("analysis/first-analysis.Rmd")
```

```yaml
---
title: "first-analysis"
author: "Pakillo"
date: "2021-06-15"
output: workflowr::wflow_html
editor_options:
  chunk_output_type: console
---
```

Introduction

```{r}
data(iris)
plot(iris)
```

`wflow_build` runs analyses and generates website

```
wflow_build()
```

`wflow_publish` commits changes & updates everything

```r
wflow_publish(c("analysis/first-analysis.Rmd",
                "analysis/index.Rmd",
                "analysis/about.Rmd",
                "analysis/license.Rmd"),
              message = "Publish initial analyses")
```

Connect with GitHub/GitLab and deploy website

```r
wflow_use_github("Pakillo")

wflow_git_push()
```