15 tips to write better analytic code

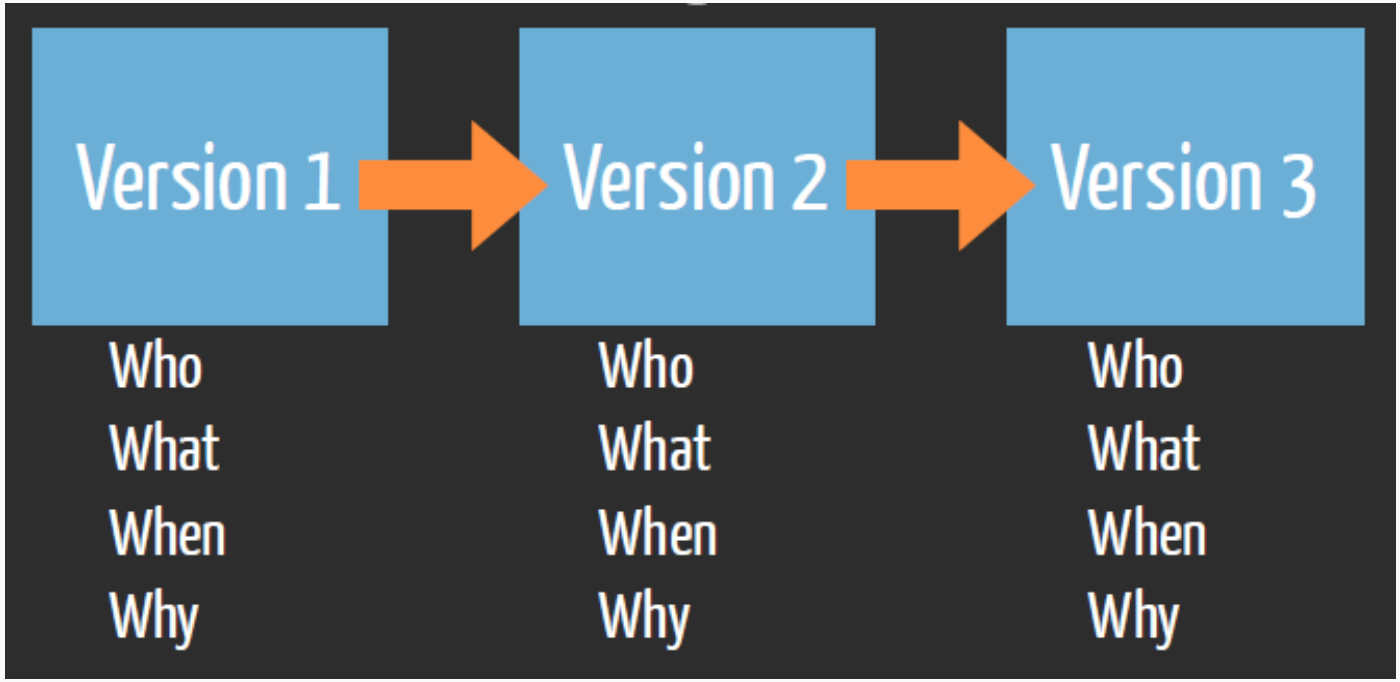
Francisco Rodríguez-Sánchez

@frod_san

https://frodriguezsanchez.net

1. Version control everything

Version control data, code, outputs (`git`)



R. Fitzjohn

2. Use projects

Use (Rstudio) projects

Everything in a folder

```
- data/
  |- raw_data/
  |- derived_data/

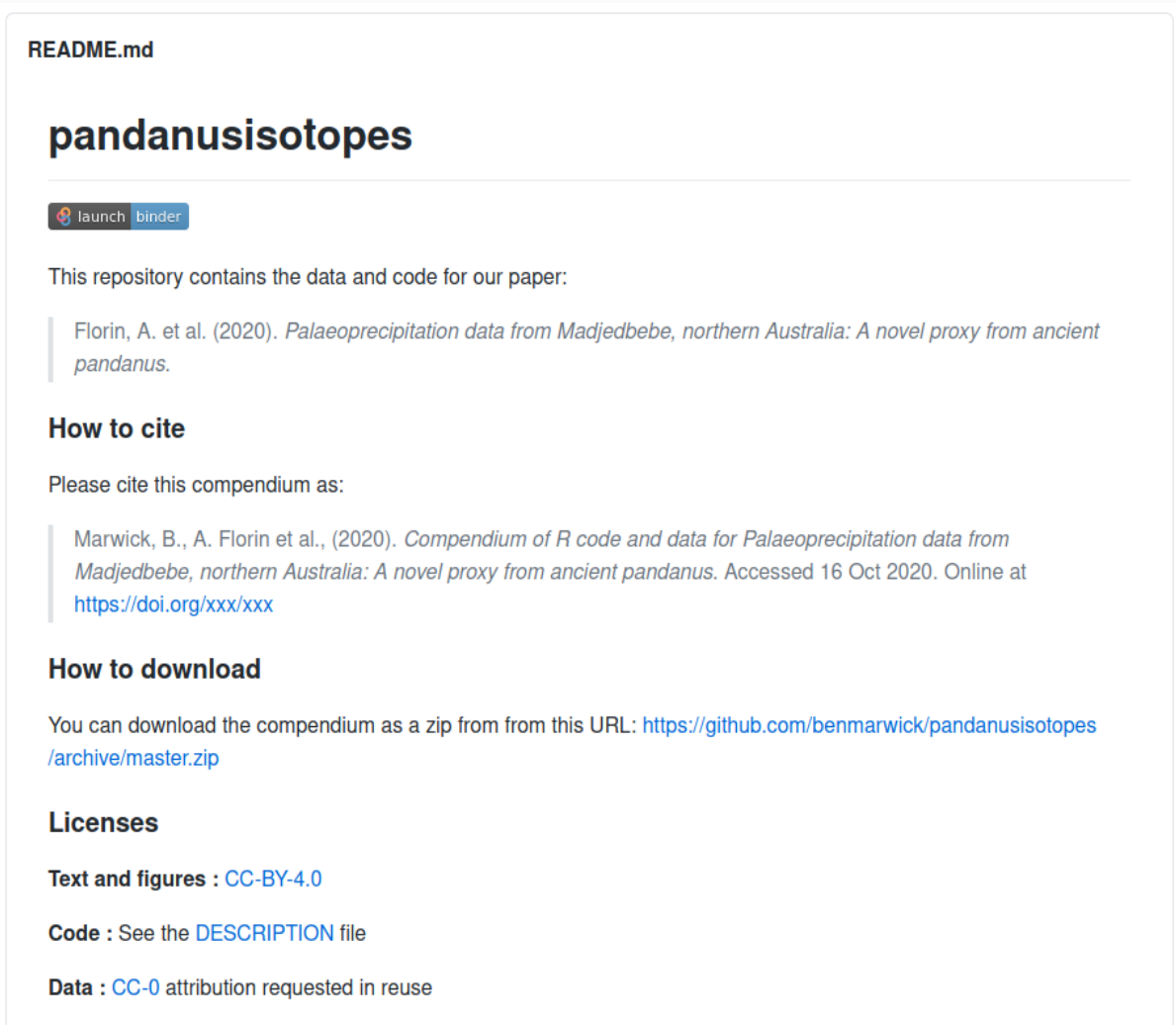
- analysis/
  |- paper/
      |- paper.Rmd
      |- references.bib
  |- figures/

- README
- LICENSE
```

3. Use README

Use README

- 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

4. Use a master script

`makefile` runs code in appropriate order

`makefile.R`

```
source("clean_data.R")

source("fit_model.R")

source("generate_report.R")
```

`makefile` runs code in appropriate order

https://github.com/Pakillo/exclosures-Almoraima

```
#### READ AND PREPROCESS DATA #####

## Read site info
read_siteinfo("data-raw/sites_info_raw.csv")

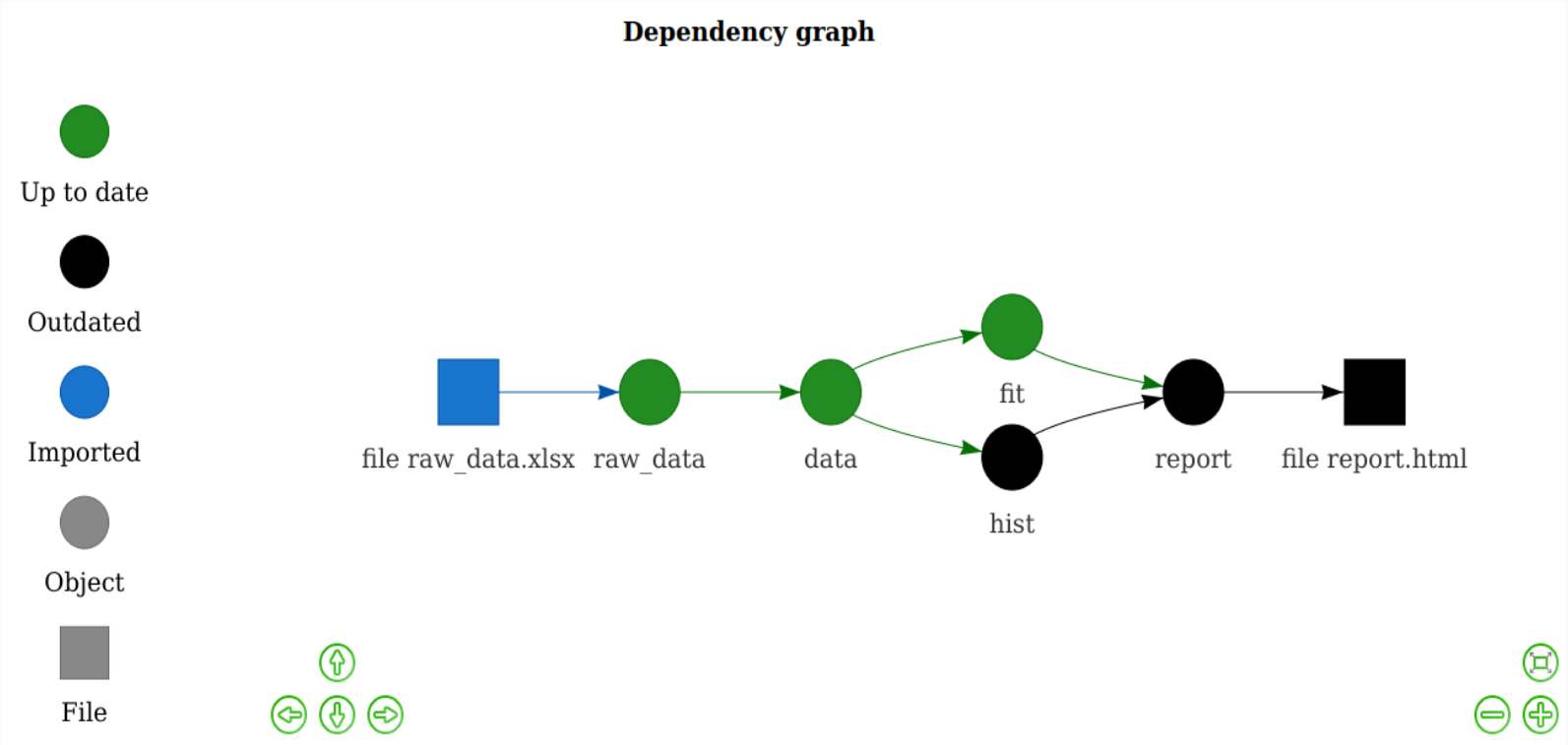
## Read and prepare species info
read_sppinfo(sppdata = "data-raw/species_info_raw.csv")

## Prepare dataset
make_dataset()


#### EXPLORATORY ANALYSIS ####
rmarkdown::render("analyses/EDA.Rmd")


#### MANUSCRIPT ####
rmarkdown::render("manuscript/cercados_Almoraima/cercados_Almoraima.Rmd")
```

drake/targets: v. powerful approach to manage workflow

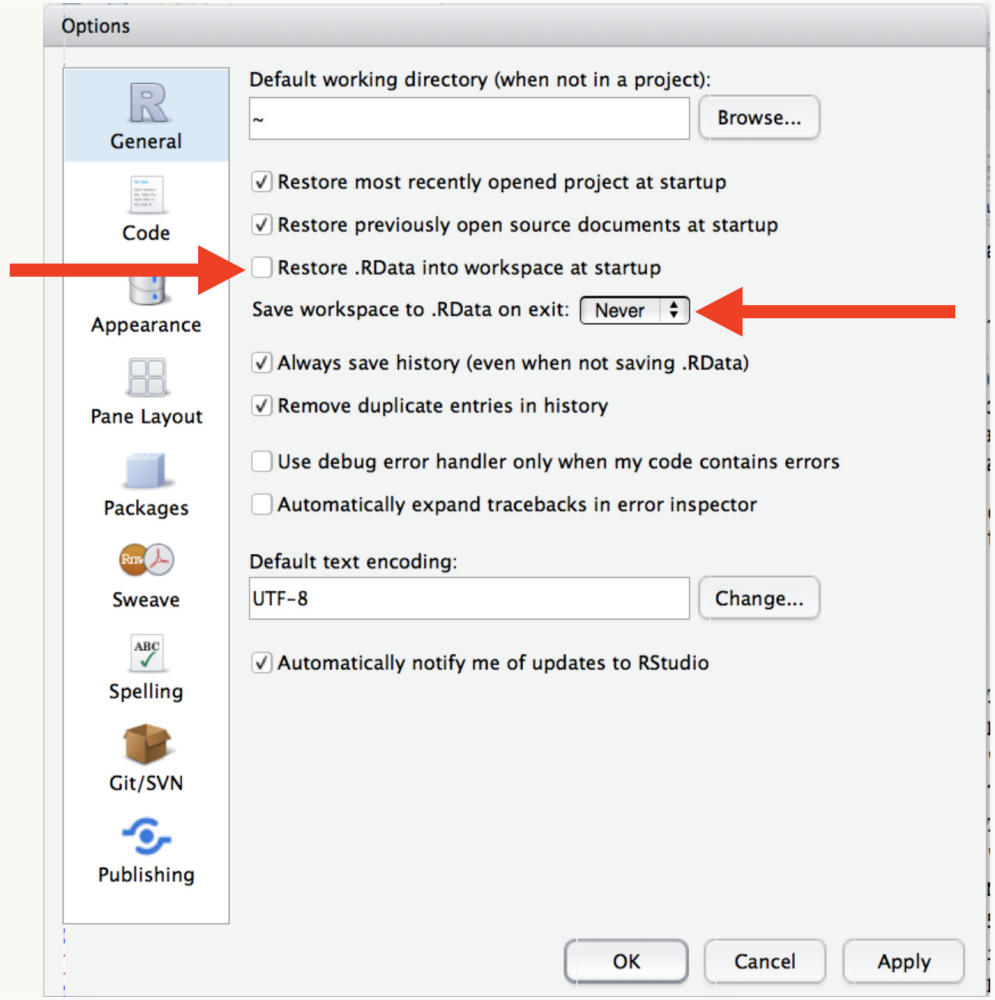


https://docs.ropensci.org/drake/

https://wlandau.github.io/targets/

5. Do not save workspace

Save source code, not workspace



https://rstats.wtf

6. Use Rmarkdown

Use Rmarkdown

a

```
---
title: "Does sunshine make people happy?"
output: pdf_document
bibliography: refs.bib
---

# Introduction

Climate influences individual well-being [@Rehdanz_2005].
However, ...

# Methods

```{r echo=FALSE}
# read data
data <- read.table("data.txt", header=T)
data[10,1] <- 11   # correct error

# fit linear model
model <- lm(happiness ~ sunshine, data=data)
```

We collected data on `r nrow(data)` individuals and fitted a
linear model.

Results

We found that...

```{r echo=FALSE, results='asis'}
# make table with model output
print(xtable::xtable(model), comment = FALSE)
```

```{r echo=FALSE, fig.height=3, fig.width=3, fig.align='center'}
visreg::visreg(model) # plot
```

Discussion

Our results confirm that happiness is related to
sunshine (slope = `r coef(model)[2]`).

References
```

b

Does sunshine make people happy?

Introduction

Climate influences individual well-being (Rehdanz and Maddison 2005). However, ...

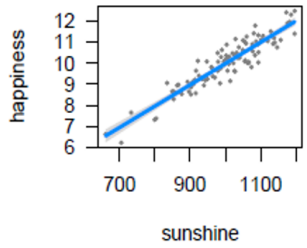
Methods

We collected data on 100 individuals and fitted a linear model.

Results

We found that...

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.0986	0.4271	-0.23	0.8180
sunshine	0.0101	0.0004	23.75	0.0000



Discussion

Our results confirm that happiness is related to sunshine (slope = 0.0100652).

References

Rehdanz, Katrin, and David Maddison. 2005. "Climate and Happiness." *Ecological Economics* 52 (1). Elsevier BV: 111–25. doi:10.1016/j.ecolecon.2004.06.015.

7. Use tools that help you write better code

Use `fertile` to get instant feedback on code reproducibility

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

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

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

8. Use comments

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)
```

9. Use meaningful names

Use meaningful names for objects

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

Use meaningful names for objects

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

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

10. Document your data

Document your data

```
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
```

11. Check data before analysis

Check data before analysis

```
library("assertr")

dataset %>%
  assert(within_bounds(0, 0.20), fruit.weight) %>%
  assert(in_set("black", "red"), colour)
```

Check out also `pointblank`

12. Check outputs

Check outputs

```
output %>%
  verify("result" %in% names(output)) %>%
  assert(within_bounds(1, 100), result)
```

13. Write modular code

Write modular code

Break up scripts

```
prepare_data.R
explore_data.R
run_analysis.R
make_figures.R
```

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

Write modular code

Write functions (documented + tested)

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

14. Don’t Repeat Yourself (DRY)

Don’t Repeat Yourself

```
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

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

```
library("purrr")

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

15. Document dependencies

`sessionInfo` records OS, package versions etc.

```
sessionInfo()
```

```
R version 4.0.3 (2020-10-10)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.1 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0

locale:
 [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=es_ES.UTF-8        LC_COLLATE=en_GB.UTF-8
 [5] LC_MONETARY=es_ES.UTF-8    LC_MESSAGES=en_GB.UTF-8
 [7] LC_PAPER=es_ES.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=es_ES.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] knitr_1.30

loaded via a namespace (and not attached):
 [1] compiler_4.0.3  magrittr_2.0.1  htmltools_0.5.0 tools_4.0.3
 [5] yaml_2.2.1      stringi_1.5.3   rmarkdown_2.5   binb_0.0.6
 [9] stringr_1.4.0   xfun_0.19       digest_0.6.27   rlang_0.4.9
[13] evaluate 0.14
```

`automagic` records used packages (CRAN + GitHub)

```
automagic::make_deps_file()
```

Creates `deps.yaml` specifying dependencies:

```
- Package: equatiomatic
  Repository: CRAN
  Version: 0.1.0

- Package: report
  GithubUsername: easystats
  GithubRepo: report
  GithubRef: HEAD
  GithubSHA1: c48a4bb0a40df7116bc502aa3ce2cbbc9d70b7e2
```

To install all those dependencies:

```
automagic()
```

renv also records dependencies

```
renv::init()

renv::snapshot()
# Captures dependencies in lockfile

renv::restore()
# Restore dependencies from lockfile
```

https://environments.rstudio.com/

15 tips to write better analytic code

1. **Version control** everything
2. Use **projects**
3. Use **README**
4. Use **makefile**
5. Do not save **workspace**
6. Use **Rmarkdown**
7. Use **tools** that help you write better code
8. **Comment** your code
9. Use **meaningful names**
10. **Document** your data
11. **Check data** before analysis
12. **Check outputs**
13. Write **modular code**
14. **Don't repeat** yourself
15. Document **dependencies**

To learn more

- BES guide to reproducible code
- Turing Way
- Good enough practices in scientific computing
- What they forgot to teach you about R