MONASH University

MONASH BUSINESS SCHOOL

Efficient reproducible workflows with R

Rob J Hyndman

3 July 2025

robjhyndman.com/pku2025

Research tools

quarto®

renv

targets

github.com/ropensci/targets

Outline

Reproducibility

Not reproducible:

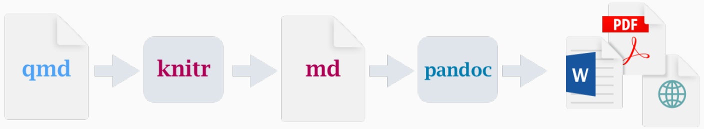
- Data edited in a spreadsheet
- Click and point analysis
- Copy and paste graphs and tables
- Tables typed by hand

Reproducible

- All data edits scripted
- All analysis scripted
- Graphs and tables automatically pulled in to the thesis
- Tables generated with scripts

Quarto

- Next generation of Rmarkdown.
- Supports R, Python, Javascript and Julia chunks.
- Separates style from content
- Format complex equations
- Automatic numbering and bibliography
- Many output formats, and many options for customizing format.
- Download and help: `quarto.org`



Code chunks

Chunk options use the hash-pipe `#|`

~~~
```{r}
#| label: fig-chunklabel
#| fig-caption: My figure
#| fig-width: 6
#| fig-height: 4
mtcars |>
  ggplot(aes(x = mpg, y = wt)) +
  geom_point()
```
~~~

Reference the figure using `@fig-chunklabel`.

Extensions and templates

- Quarto extensions modify and extend functionality.
- They are stored locally, in the `_extensions` folder alongside the qmd document.
- See `https://quarto.org/docs/extensions/` for a list.
- Templates are extensions used to define new output formats.
- Journal templates at `https://quarto.org/docs/extensions/listing-journals.html`
- Monash templates at `https://github.com/quarto-monash`

quarto on the command line

- `quarto render` to render a quarto or Rmarkdown document.
- `quarto preview` to preview a quarto or Rmarkdown document.
- `quarto add <gh-org>/<gh-repo>` to add an extension from a github repository.
- `quarto update <gh-org>/<gh-repo>` to update an extension
- `quarto remove <gh-org>/<gh-repo>` to remove an extension
- `quarto list extensions installed`
- `quarto use template <gh-org>/<gh-repo>` to use existing repo as starter template.

Add a custom format

From the CLI: `quarto add quarto-monash/memo`

Add a custom format

From the CLI: `quarto add quarto-monash/memo`

New folder/files added

```
├── _extensions
│   └── quarto-monash
│       └── memo
│           └── ...
```

Add a custom format

From the CLI: `quarto add quarto-monash/memo`

New folder/files added

```
├── _extensions
│   └── quarto-monash
│       └── memo
│           └── ...
│
```

Update YAML

```
---
title: "My new file using the Monash memo format"
format: memo-pdf
---
```

Outline

Reproducible environments

- Creates project-specific R environments.
- Uses a package cache so you are not repeatedly installing the same packages in multiple projects.
- Does not ensure R itself, system dependencies or the OS are the same.
- Not a replacement for Docker or Apptainer.

Reproducible environments

- Can use packages from CRAN, Bioconductor, GitHub, Gitlab, Bitbucket, etc.
- `renv::init()` to initialize a new project.
- `renv::snapshot()` to save state of project to `renv.lock`.
- `renv::restore()` to restore project as saved in `renv.lock`.
- `renv::update()` gets latest versions of all dependencies from wherever

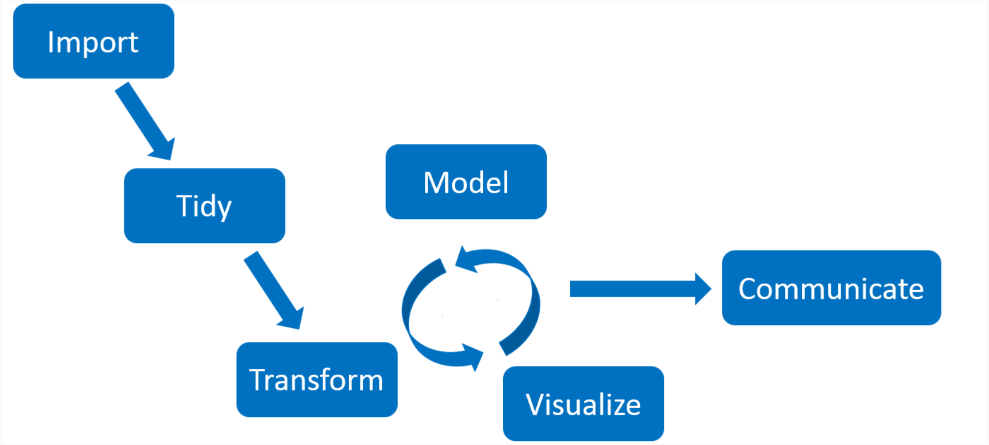
Outline

targets: reproducible computation at scale

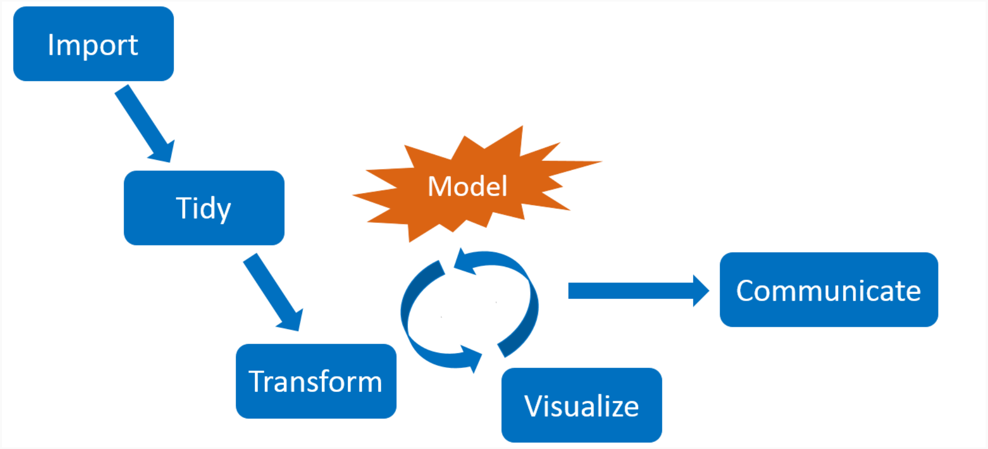
- Supports a clean, modular, function-oriented programming style.
- Learns how your pipeline fits together.
- Runs only the necessary computation.
- Abstracts files as R objects.
- Similar to Makefiles, but with R functions.

Some images from https://wlandau.github.io/targets-tutorial

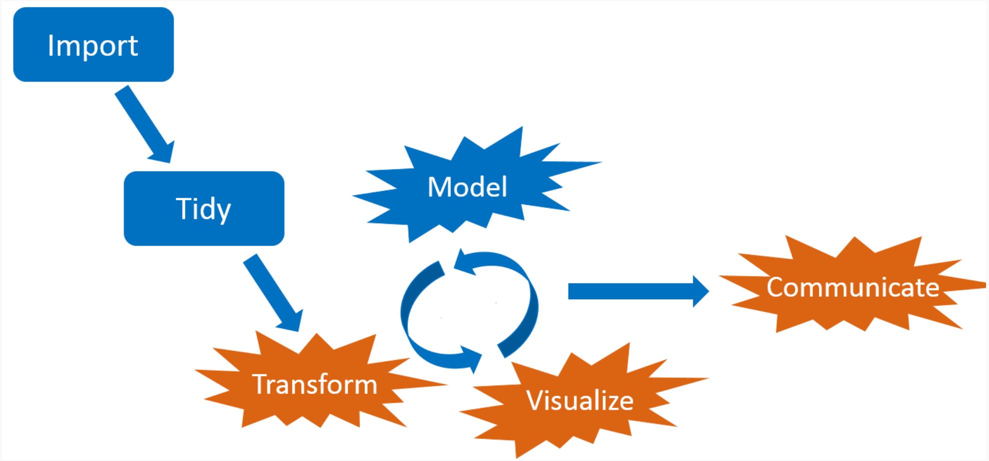
Interconnected tasks



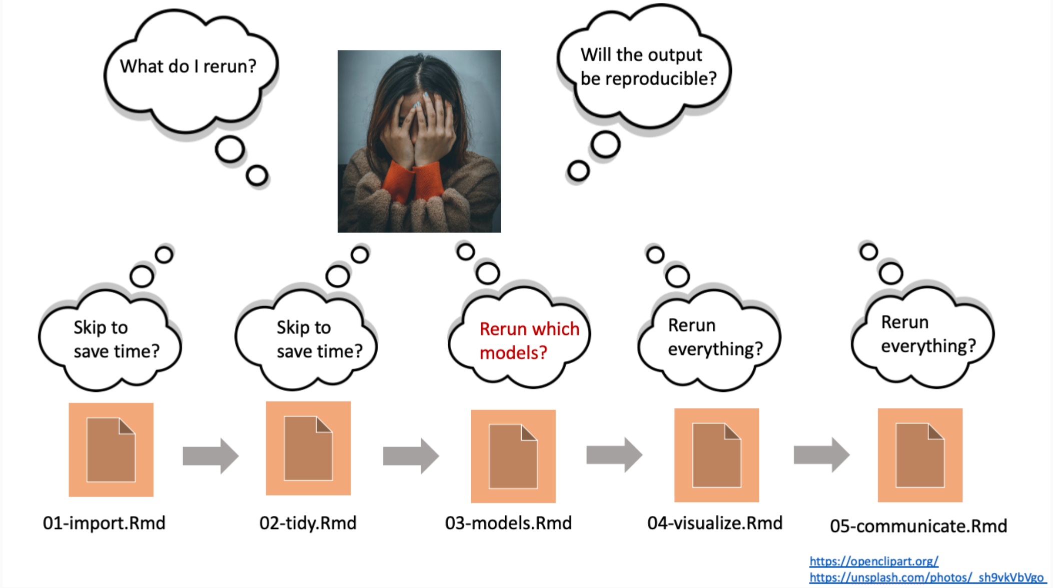
Interconnected tasks



Interconnected tasks

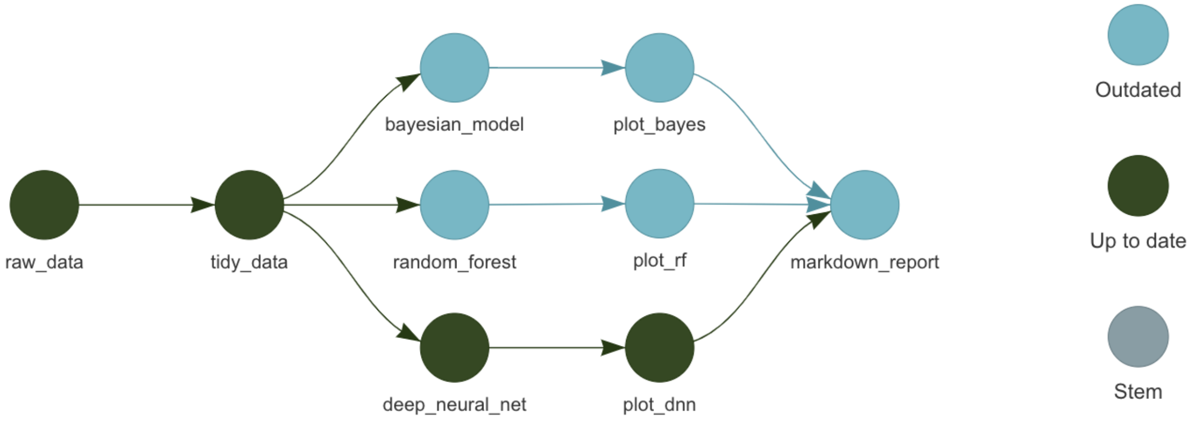


Dilemma: short runtimes or reproducible results?



https://openclipart.org/
https://unsplash.com/photos/_sh9vkVbVgo

Let a pipeline tool do the work



- Save time while ensuring computational reproducibility.
- Automatically skip tasks that are already up to date.

Typical project structure

no_targets.R

```r
library(tidyverse)
library(fable)
source("R/functions.R")
my_data <- read_csv("data/my_data.csv")
my_model <- model_function(my_data)
```

Typical project structure

no_targets.R

```r
library(tidyverse)
library(fable)
source("R/functions.R")
my_data <- read_csv("data/my_data.csv")
my_model <- model_function(my_data)
```

_targets.R

```r
library(targets)
tar_option_set(packages = c("tidyverse", "fable"))
tar_source() # source all files in R folder
list(
  tar_target(my_file, "data/my_data.csv", format = "file"),
  tar_target(my_data, read_csv(my_file)),
  tar_target(my_model, model_function(my_data))
)
```

Generate `_targets.R` in working directory

```r
library(targets)
tar_script()
```

Useful targets commands

- `tar_make()` to run the pipeline.
- `tar_make(starts_with("fig"))` to run only targets starting with “fig”.
- `tar_read(object)` to read a target.
- `tar_load(object)` to load a target.
- `tar_load_everything()` to load all targets.
- `tar_manifest()` to list all targets
- `tar_visnetwork()` to visualize the pipeline.
- `tar_destroy()` to remove all targets.
- `tar_outdated()` to list outdated targets.

Random numbers

- Each target runs with its own seed based on its name and the global seed from `tar_option_set(seed = ???)`
- So running only some targets, or running them in a different order, will not change the results.

Folder structure

```
├── .git/
├── .Rprofile
├── .Renviron
├── renv/
├── index.Rmd
├── _targets/
├── _targets.R
├── _targets.yaml
├── R/
├──── functions_data.R
├──── functions_analysis.R
├──── functions_visualization.R
├── data/
└──── input_data.csv
```

`_targets.R` with quarto

```r
library(targets)
library(tarchetypes)                                                          (1)
tar_source() # source all files in R folder
tar_option_set(packages = c("tidyverse", "fable"))
list(
  tar_target(my_file, "data/my_data.csv", format = "file"),
  tar_target(my_data, read_csv(my_file)),
  tar_target(my_model, model_function(my_data)),
  tar_quarto(report, "file.qmd", extra_files = "references.bib")              (2)
  )
```

1. Load `tarchetypes` package for quarto support.
2. Add a quarto target.

Replace quarto chunks with `tar_read()` or `tar_load()`.

Chunk options

Chunk with regular R code

~~~
```{r}
#| label: fig-chunklabel
#| fig-caption: My figure
mtcars |>
  ggplot(aes(x = mpg, y = wt)) +
  geom_point()
```
~~~

Chunk options

Chunk with regular R code

~~~
```{r}
#| label: fig-chunklabel
#| fig-caption: My figure
mtcars |>
  ggplot(aes(x = mpg, y = wt)) +
  geom_point()
```
~~~

Chunk with targets

~~~
```{r}
#| label: fig-chunklabel
#| fig-caption: My figure
tar_read(my_plot)
```
~~~

Example paper

ISSN: 0160-5682

JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY

JORS

THE OPERATIONAL RESEARCH SOCIETY

Taylor & Francis

Hyndman RJ, Rostami-Tabar B (2024) Forecasting interrupted time series, *Journal of the Operational Research Society*, in press.

bahmanrostamitabar/forecasting_interrupted_time_series

Slides and links

robjhyndman.com/pku2025

robjhyndman.com/postdoc