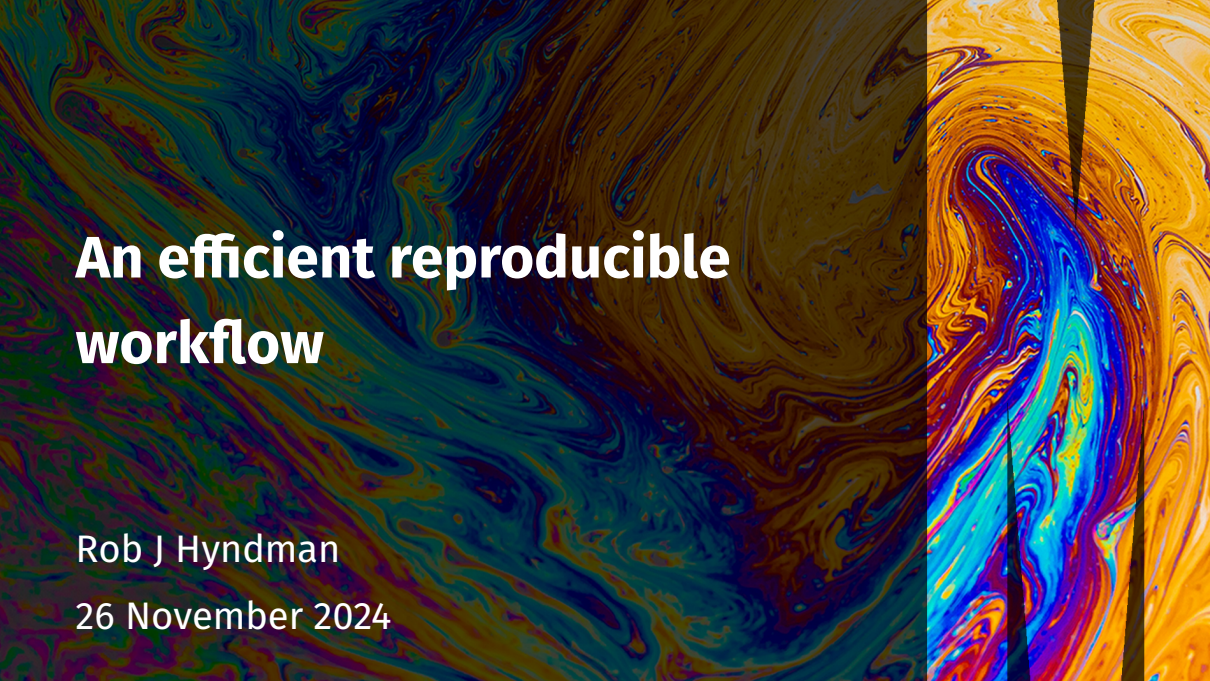


An efficient reproducible workflow

Rob J Hyndman

26 November 2024

Example paper

Hyndman RJ, Rostami-Tabar B (2024) Forecasting interrupted time series, *Journal of the Operational Research Society*, in press.
DOI: 10.1080/01605682.2024.2395315

bahmanrostamitabar/
forecasting_interrupted_time_series

Tools


GitHub


R


renv
www.rstudio.com


targets
github.com/ropensci/targets


quarto

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`.

Efficient computational pipelines

- 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.

Interconnected tasks

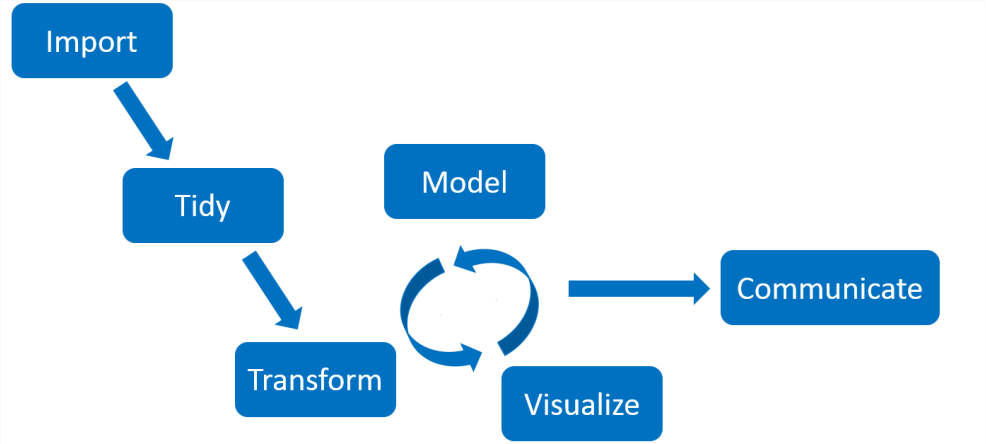


Image from https://wlandau.github.io/targets-tutorial

Interconnected tasks

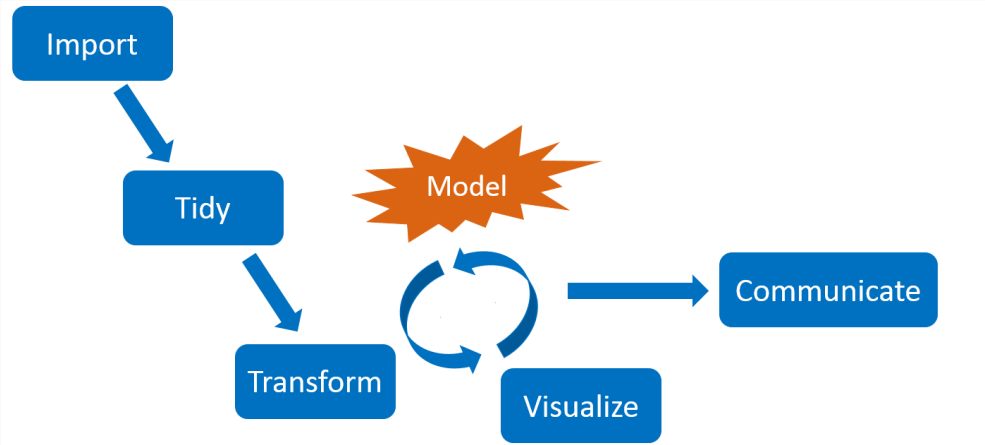


Image from https://wlandau.github.io/targets-tutorial

Interconnected tasks

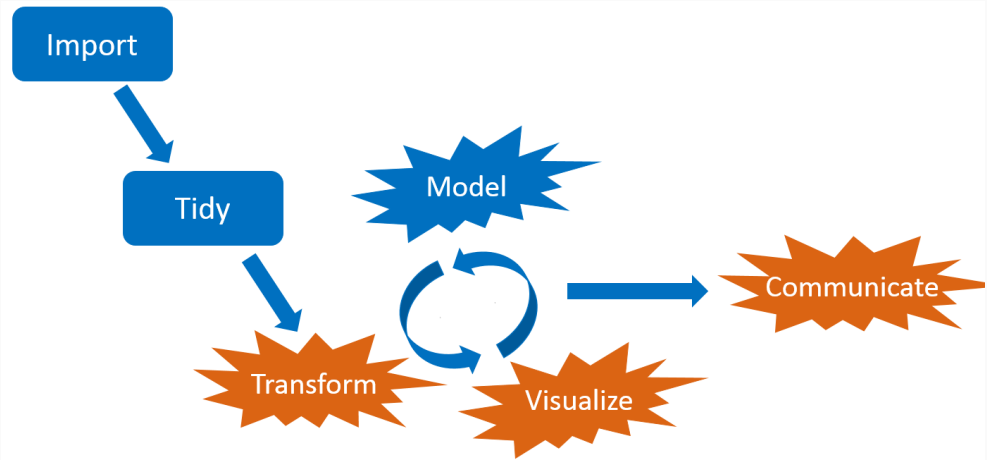


Image from https://wlandau.github.io/targets-tutorial

Interconnected tasks

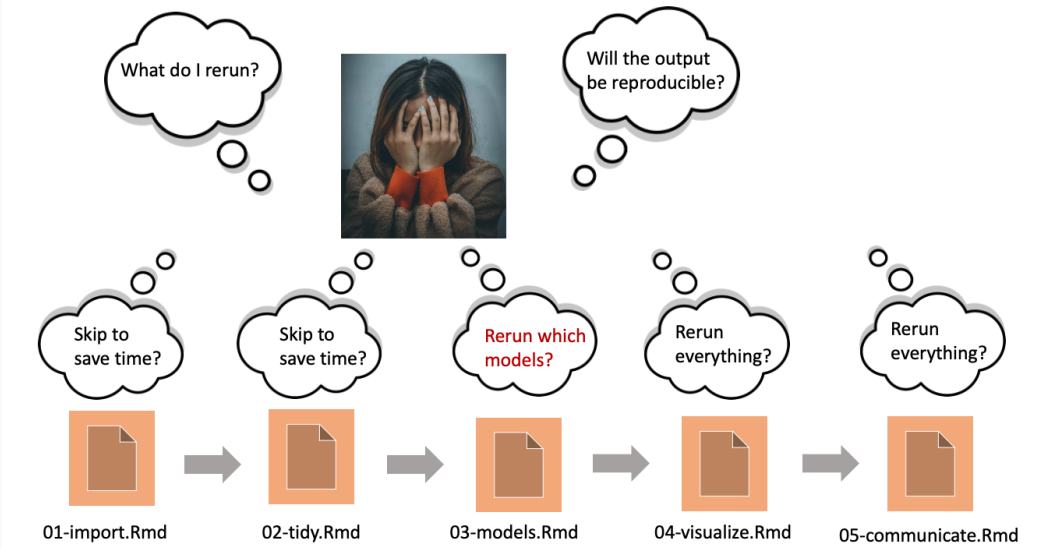
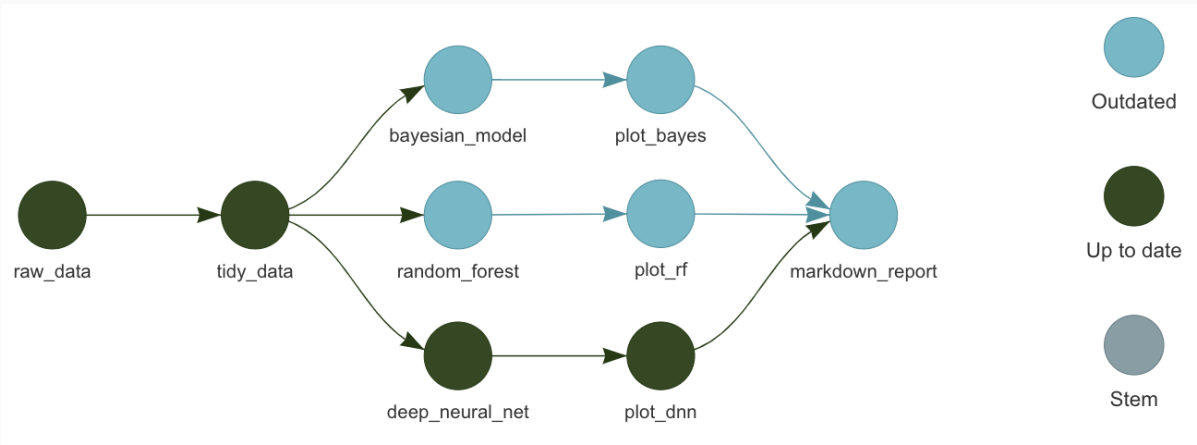


Image from https://wlandau.github.io/targets-tutorial

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

```
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

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

_targets.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))
)
```

Useful targets commands

- `tar_option_set()` to set options.
- `tar_target()` to create a target.
- `tar_source()` to source all files in a folder.
- `tar_make()` to run the pipeline.
- `tar_read(object)` to read a target.
- `tar_load(object)` to load a target.
- `tar_visnetwork()` to visualize the pipeline.

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.

Reproducible documents

- Generalization of Rmarkdown (not dependent on R)
- Supports R, Python, Javascript and Julia chunks by using either ‘knitr‘, ‘jupyter‘ or ‘ObservableJS‘ engines.
- Consistent yaml header and chunk options.
- Many output formats, and many options for customizing format.
- Uses pandoc templates for extensions

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)
```
````

Chunk options

Reference the figure using `@fig-chunklabel`.

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)
```
````

targets with quarto

```
library(targets)
library(tarchetypes)
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)),
  tar_quarto(report, "file.qmd", extra_files = "references.bib")
)
```

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

Extensions and templates

- **Quarto extensions** modify and extend functionality.
  - See `https://quarto.org/docs/extensions/` for a list.
  - They are stored locally, in the `_extensions` folder alongside the qmd document.
- **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`