

Modern research tools and workflow

robjhyndman.com/research_tools

Rob J Hyndman

30 December 2024



zotero



Semantic **Scholar**



Outline

- 1 Citing
- 2 Searching
- 3 Scripting
- 4 Saving
- 5 Caching
- 6 Writing

Managing references

Zotero

- ➔ Free and on all operating systems
- ➔ Web-version and local version synced
- ➔ Browser extension for adding papers/books
- ➔ Attach notes and annotations to papers.
- ➔ Works with Word, LibreOffice or LaTeX.
- ➔ Generate bibliography automatically
- ➔ Handles all formatting for you.

zotero

To install:

- Set up account at www.zotero.org
- Download from www.zotero.org

Managing references

Mendeley

- ➔ Free and on all operating systems
- ➔ Web-version and local version synced
- ➔ Browser extension for adding papers/books
- ➔ Attach notes and annotations to papers.
- ➔ Works with Word, LibreOffice or LaTeX.
- ➔ Generate bibliography automatically
- ➔ Handles all formatting for you.



To install:

- Set up account at mendeley.com
- Download from mendeley.com

Managing references

Paperpile

- ➔ \$3 per month and runs on Google Chrome
- ➔ Papers stored on Google Drive
- ➔ Browser extension for adding papers/books
- ➔ Works with Google Docs or LaTeX.
- ➔ Generate bibliography automatically
- ➔ Handles all formatting for you.
- ➔ Amazingly fast



To install:

- Set up account at paperpile.com
- Download Google chrome browser extension

What to cite?

- Cite what is important.
- Cite (only) what is relevant.
- Avoid lists of gratuitous references.
- Include proper citations for all packages and software you use.



What to cite?

- Cite what is important.
- Cite (only) what is relevant.
- Avoid lists of gratuitous references.
- Include proper citations for all packages and software you use.

When using R

```
citation("packagename")
```



Sight what you cite

- Every article cited should be sighted, & preferably read.
- Check that the article cited really does say what you think it says.
- Check the reference information yourself. Don't copy.
- Don't just cite what other people say about citations.
- Store accurate reference info from the start.
- Give credit where it is due.



Sight what you cite

- Every article cited should be sighted, & preferably read.
- Check that the article cited really does say what you think it says.
- Check the reference information yourself. Don't copy.
- Don't just cite what other people say about citations.
- Store accurate reference info from the start.
- Give credit where it is due.
 - ▶ Diebold did not invent PITs.
 - ▶ Hyndman did not invent exponential smoothing or ARIMA models.



Outline

- 1 Citing
- 2 Searching
- 3 Scripting
- 4 Saving
- 5 Caching
- 6 Writing

- Searching for papers
- Use advanced search
- Link GS to your reference manager
- Track citations of key papers
- Star papers for your own library
- Check recommended articles
- Check author profiles, especially highly cited authors
- Create your own GS profile once you have (at least) one paper
- Follow key authors in your area



SEMANTIC SCHOLAR

A free, AI-powered research tool for scientific literature

Search 211,093,141 papers from all fields of science

Search 🔍

Try: [Jean Louise Cohen](#) • [Market Structure](#) • [Cultural Universals](#)

Outline

- 1 Citing
- 2 Searching
- 3 Scripting**
- 4 Saving
- 5 Caching
- 6 Writing

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



**STOP CLICKING
AND START SCRIPTING**

Reproducibility

Someone should be able to reproduce your thesis or paper without having to guess what software you had installed, what versions, which files do what, etc.

- Stay organized.
- Write a readme file to explain how to reproduce your thesis or paper.
- Use a scripting language such as R, Python, or Julia.
- Use a reproducible environment to lock software versions.

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`.

Outline

- 1 Citing
- 2 Searching
- 3 Scripting
- 4 Saving**
- 5 Caching
- 6 Writing

Version control

- `thesis_v1`, `thesis_v2`, etc., is not adequate version control.
- You need to track changes over time, have a *remote* repository, and be able to roll back as required.
- Your repository should contain *everything* required to produce your thesis including computer code, references, writing.
- Your repository should have an obvious structure and be fully documented.
- **Github** solves these problems
- Read “Happy git with R”: happygitwithr.com



Version control with git

- RStudio integrates with github, so version control built in.
- But github can be used with *any* text-based language including Stata, Python, LaTeX, R, Rmarkdown, Quarto, markdown, etc.
- Git allows you to:
 - ▶ track changes
 - ▶ experiment in branches
 - ▶ undo
- Github provides:
 - ▶ backup and restore
 - ▶ synchronisation



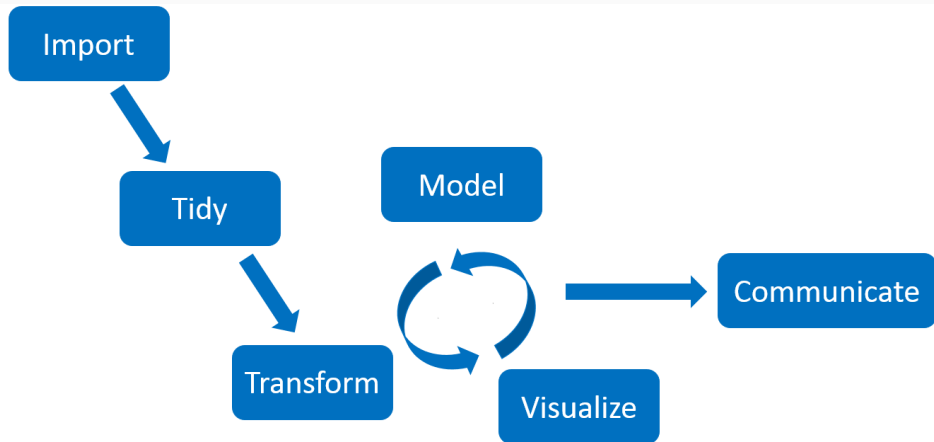
Outline

- 1 Citing
- 2 Searching
- 3 Scripting
- 4 Saving
- 5 Caching**
- 6 Writing

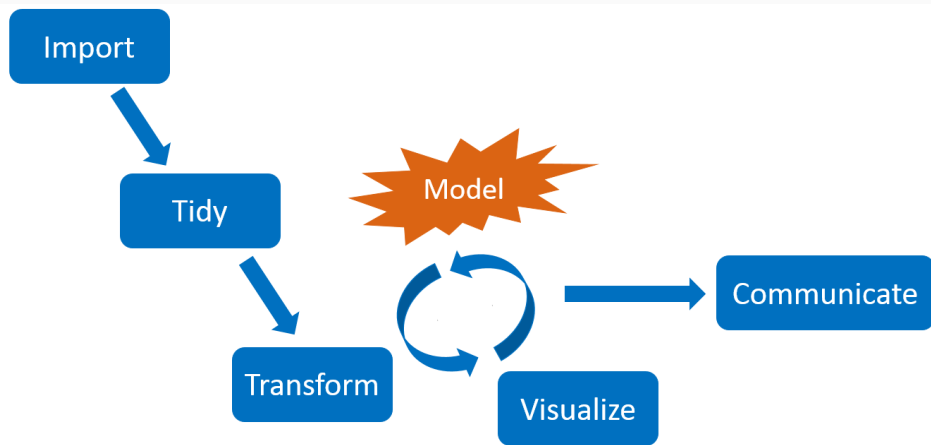


- Supports a clean, modular, function-oriented programming style.
- Learns how your pipeline fits together.
- Runs only the necessary computation.
- Abstracts files as R objects.
- Similar to Makefiles, but with R functions.

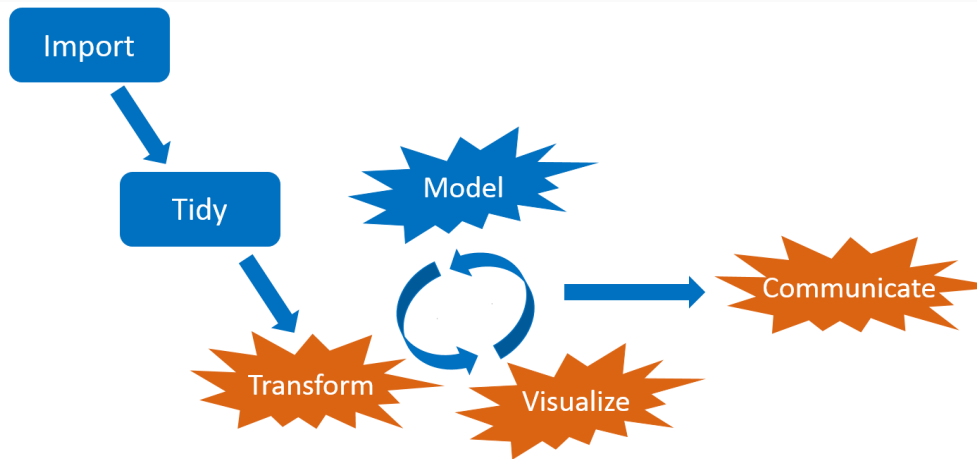
Interconnected tasks



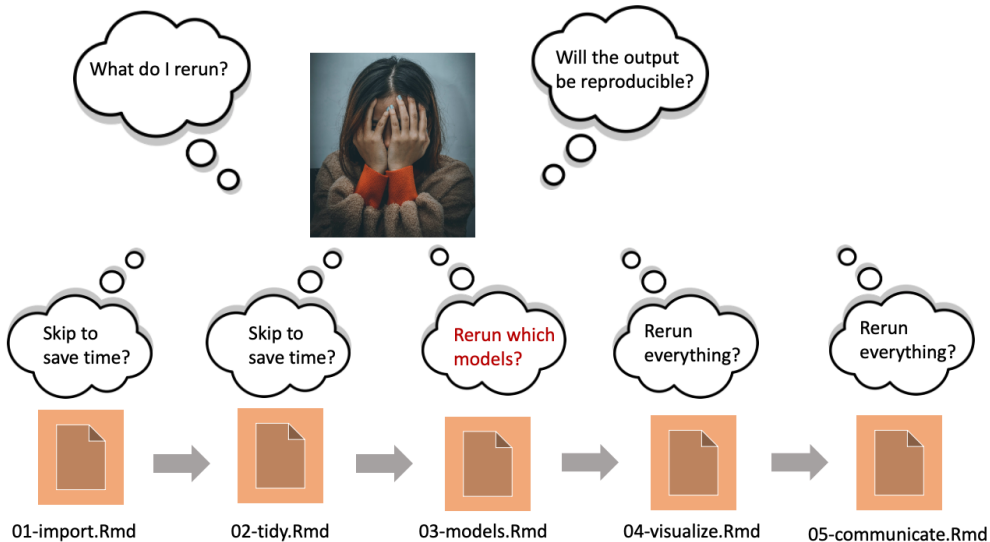
Interconnected tasks



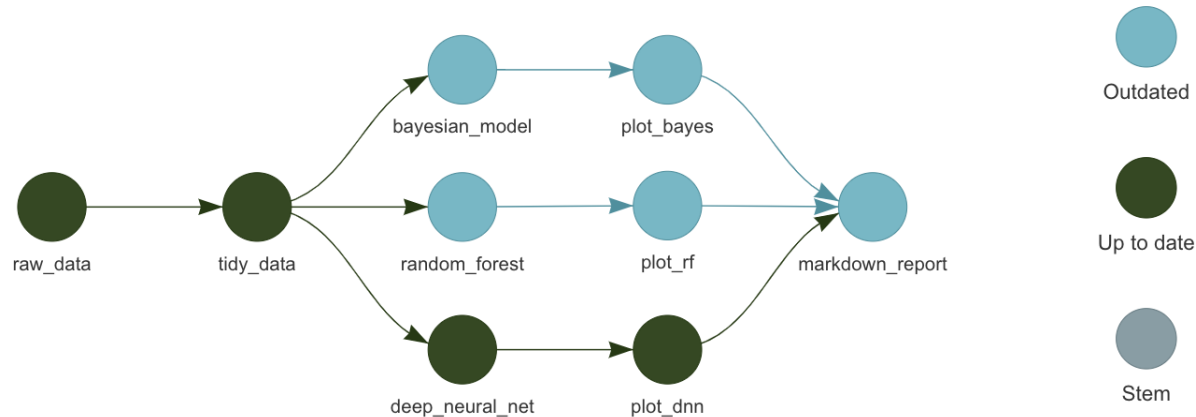
Interconnected tasks



Interconnected tasks



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.

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 = ???)`

Outline

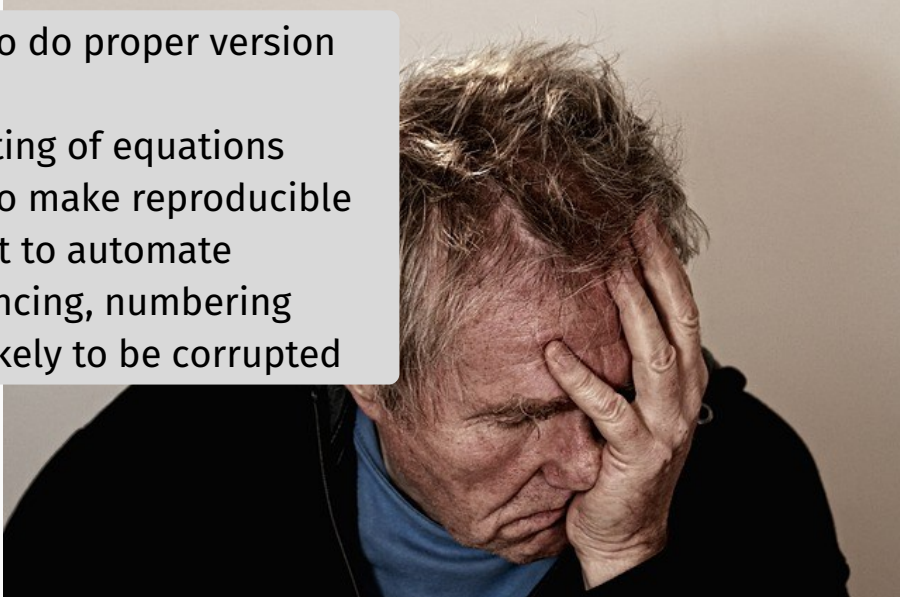
- 1 Citing
- 2 Searching
- 3 Scripting
- 4 Saving
- 5 Caching
- 6 Writing**

Microsoft Word



Microsoft Word

- Impossible to do proper version control
- Poor formatting of equations
- Impossible to make reproducible
- More difficult to automate cross-referencing, numbering
- Files more likely to be corrupted





- 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



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.
- **Quarto 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>

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.

Example paper



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



 [bahmanrostamitabar/forecasting_interrupted_time_series](#)

robjhyndman.com/research_tools