MONASH University

MONASH BUSINESS SCHOOL

Research tools

`robjhyndman/research_tools`

Rob J Hyndman

30 December 2024

Outline

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.

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.
- At the very least, check that the article cited really does say what you think it says.
- Type the reference information yourself.
- 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

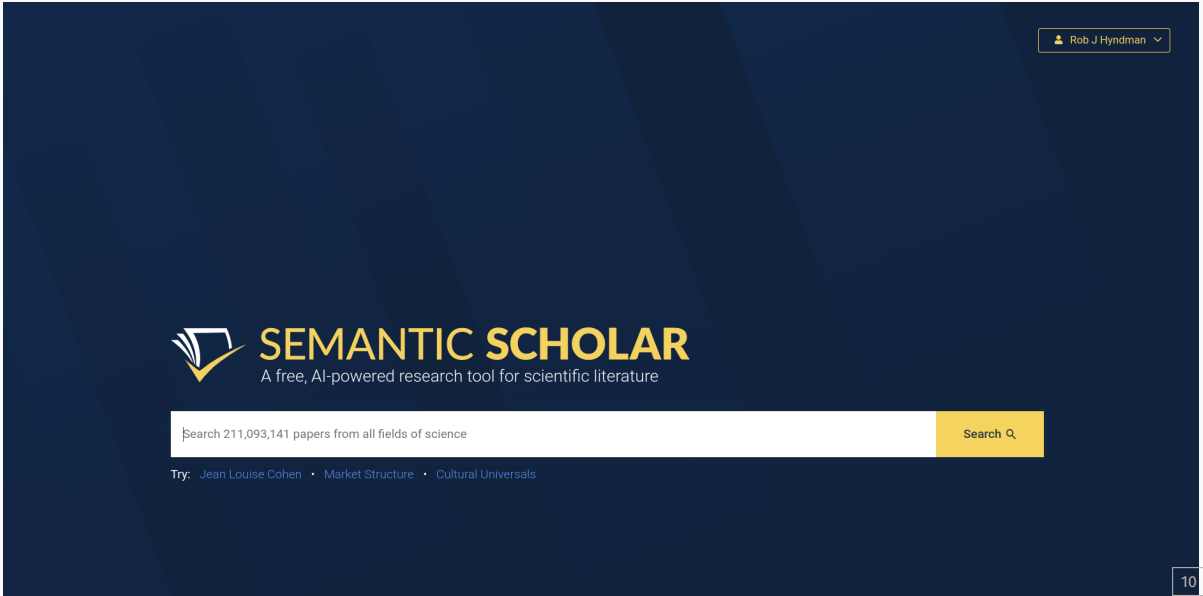
- Every article cited should be sighted, & preferably read.
- At the very least, check that the article cited really does say what you think it says.
- Type the reference information yourself.
- Don't just cite what other people say about citations.
- Store accurate reference info from the start.
- Give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

Outline

Google Scholar

- 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



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

Reproducibility

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

Reproducibility

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

- Stay organized.
- One system for doing this using R is to write your thesis in an Rmarkdown or Quarto file.
- Track software versions

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`

GitHub

Version control with git

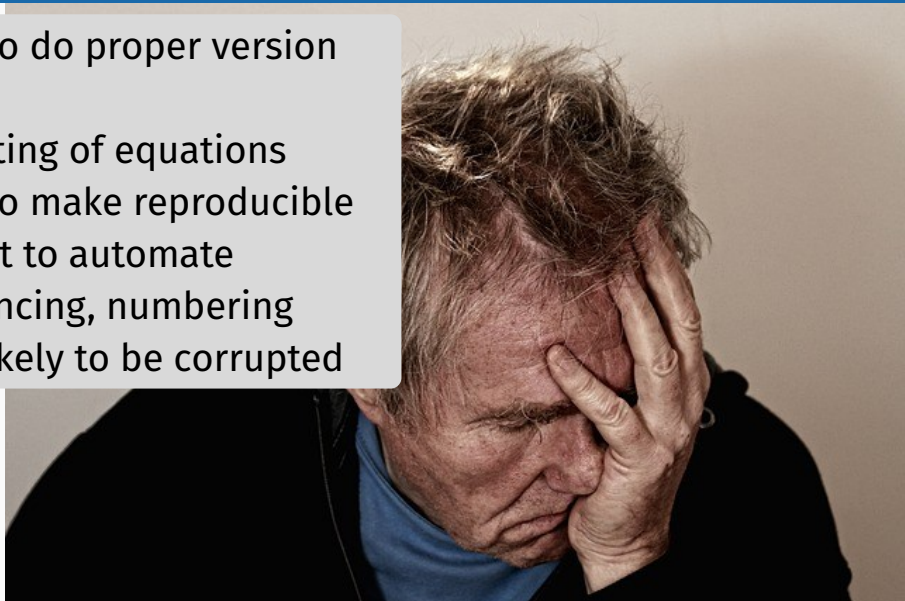
- RStudio integrates with github, so version control built in.
- But github can be used with *any* text-based language including Stata, Python, LaTeX, R, Rmarkdown, Quarto, markdown, etc.
- Git allows you to:
  - track changes
  - experiment in branches
  - undo
- Github provides:
  - backup and restore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

Outline

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



LaTeX

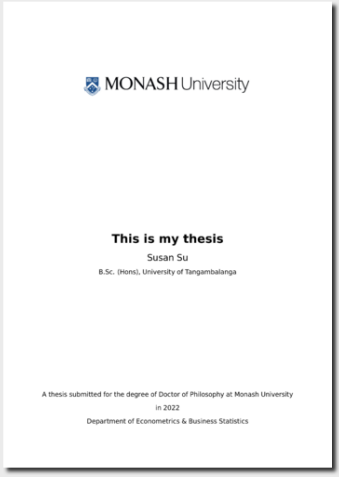
To install:

- Download MikTeX, MacTeX or TeXlive.
- Download TeXStudio from `texstudio.sourceforge.net/`

Document processing

- Free and open-source
- Available on all operating systems
- Used by every mathematical publisher
- Separate content from style
- Format complex equations
- Automatic numbering
- Automatic bibliography
- Almost every language

Quarto


MONASH University

This is my thesis

Susan Su

B.Sc. (Hons), University of Tangambalanga

A thesis submitted for the degree of Doctor of Philosophy at Monash University in 2022

Department of Econometrics & Business Statistics

Quarto

- Combines R, Python and LaTeX into one system
- Reproducible research
- Monash Thesis Template via `github.com/numbats/monash-quarto-thesis`
- Useful for assignments too
- See `quarto.org` for help

Outline

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

Tools

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

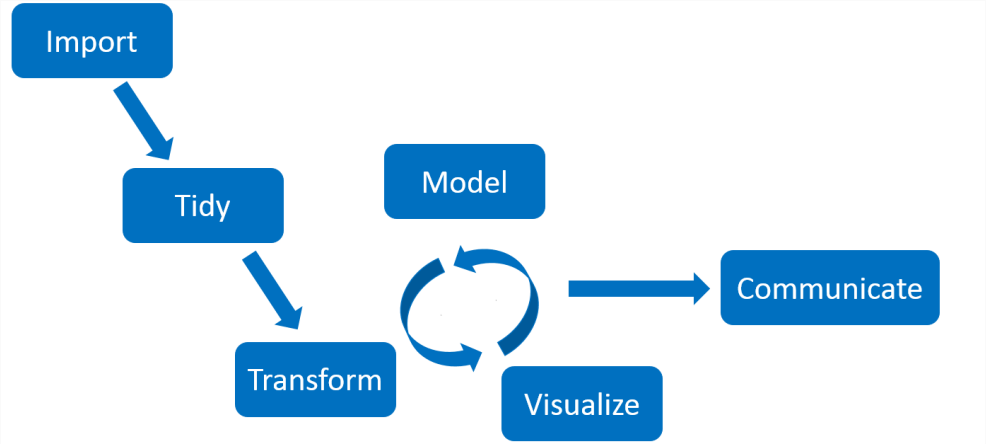


Image from https://wlandau.github.io/targets-tutorial

Interconnected tasks

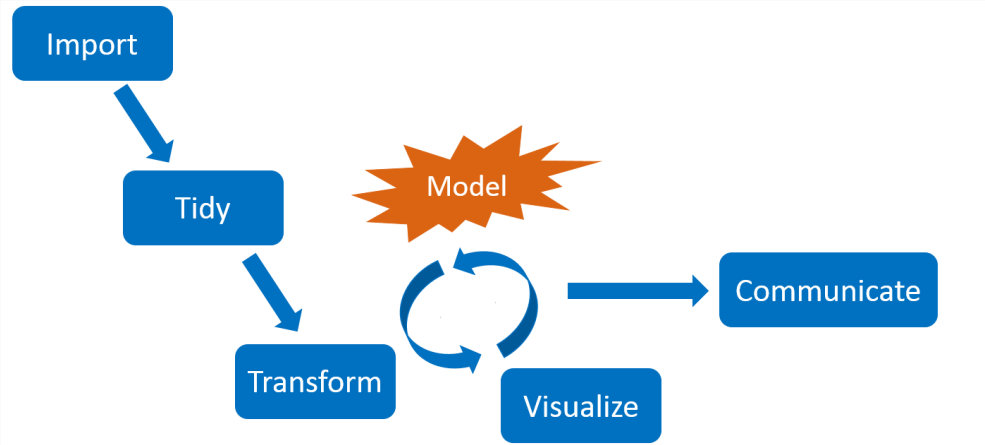


Image from https://wlandau.github.io/targets-tutorial

Interconnected tasks

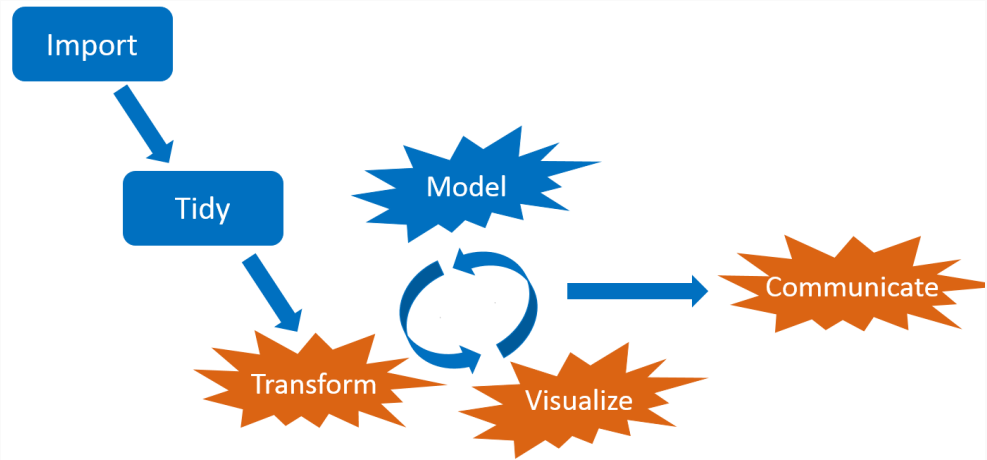


Image from https://wlandau.github.io/targets-tutorial

Interconnected tasks

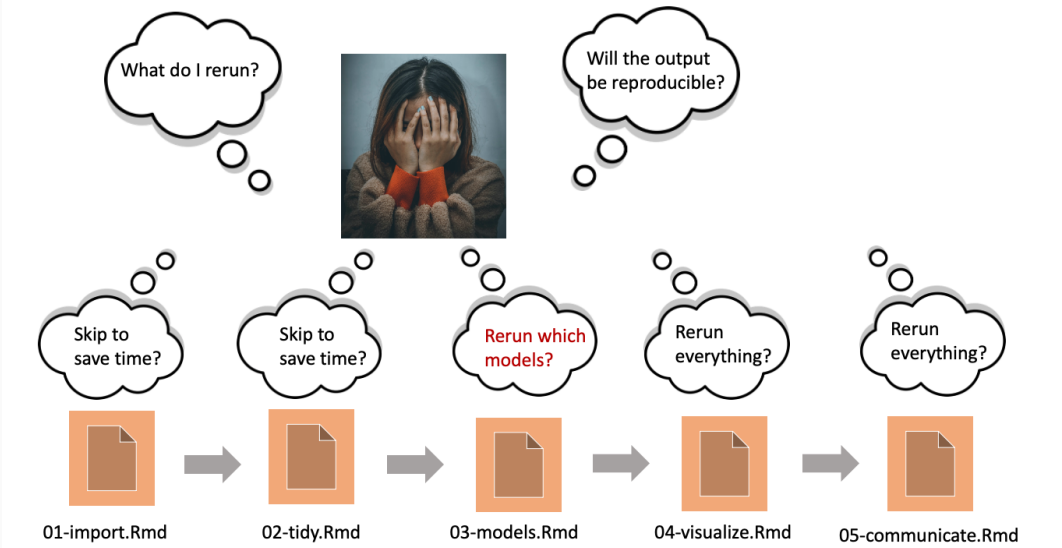
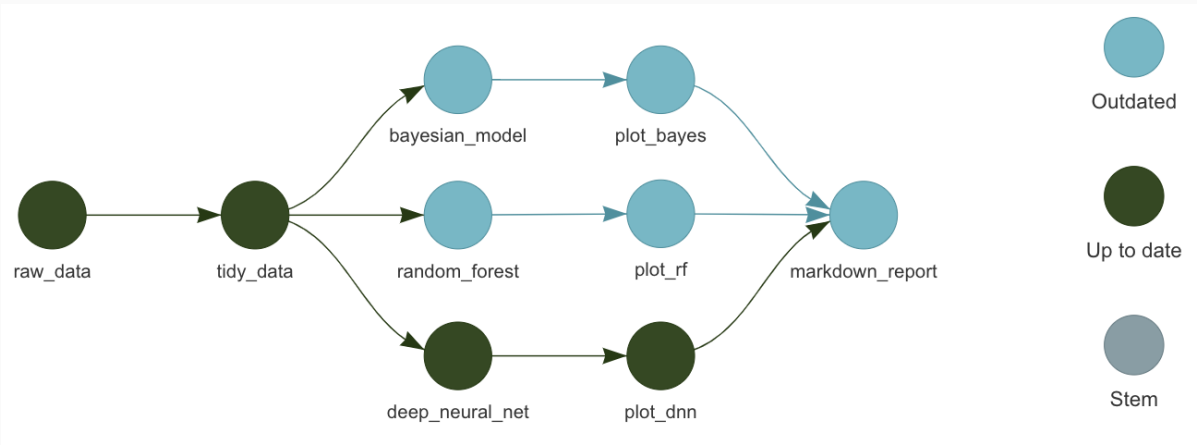


Image from https://wlandau.github.io/targets-tutorial

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.

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 = ???)`

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.
- **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`

Slides and links

`robjhyndman.com/research_tools`