MONASH University

MONASH BUSINESS SCHOOL

ETC4500/ETC5450
Advanced R programming

Week 8: Quarto and targets – efficient reproducible workflows

`arp.numbat.space`

Outline

Outline

Assignments

- Assignment 2 feedback
- Assignment 3 due 10 May
- Assignment 4 due 24 May

Outline

Quarto

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

Choose your engine

Specify the engine in the yaml header:

```
---
engine: knitr
---
```

```
---
engine: jupyter
jupyter: python3
---
```

Default: If any `{r}` blocks found, use `knitr` engine; otherwise use `jupyter` (with kernel determined by first block).

Execute options

- `execute` option in yaml header can be used instead of a `setup` chunk:

```
execute:
  cache: true
  echo: false
  warning: false
```

- `setup` chunk still allowed.

Chunk options

Rmarkdown syntax recognized for R chunks.

More consistent 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`.

Chunk options

- Quarto consistently uses hyphenated options (`fig-width` rather than `fig.width`)
- The Rmarkdown `knitr` options are recognized for backwards compatibility.
- Options that are R expressions need to be prefaced by `!expr`

````
```{r}
#| fig-cap: !expr paste("My figure", 1+1)
```
````

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://robjhyndman.com/hyndsight/quarto_templates.html

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 numbats/monash-quarto-memo`

Add a custom format

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

New folder/files added

```
├── _extensions
│   └── numbats
│       └── memo
│           └── ...
```

Add a custom format

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

New folder/files added

```
├── _extensions
│   └── numbats
│       └── memo
│           └── ...
```

Update YAML

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

Exercise

- Set up a new project.
- Create a quarto document using an html format.
- Add a code chunk to generate a figure with a caption.
- Reference the figure in the text using `@fig-chunklabel`.
- Add the monash memo extension and generate a pdf output.

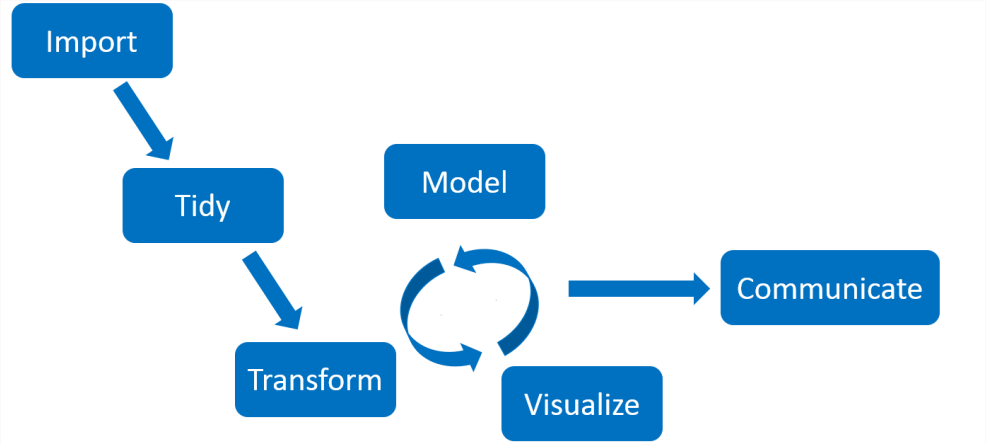
Outline

targets: reproducible computation at scale

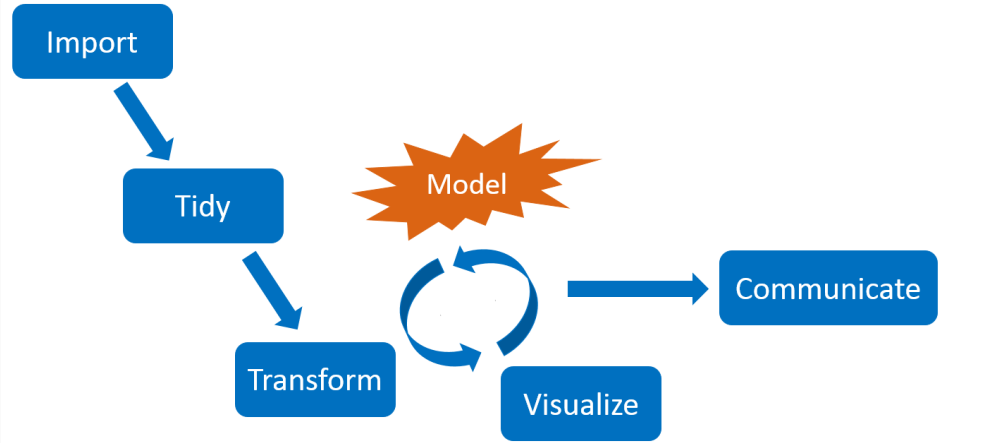
- Supports a clean, modular, function-oriented programming style.
- Learns how your pipeline fits together.
- Runs only the necessary computation.
- Abstracts files as R objects.
- Similar to Makefiles, but with R functions.

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

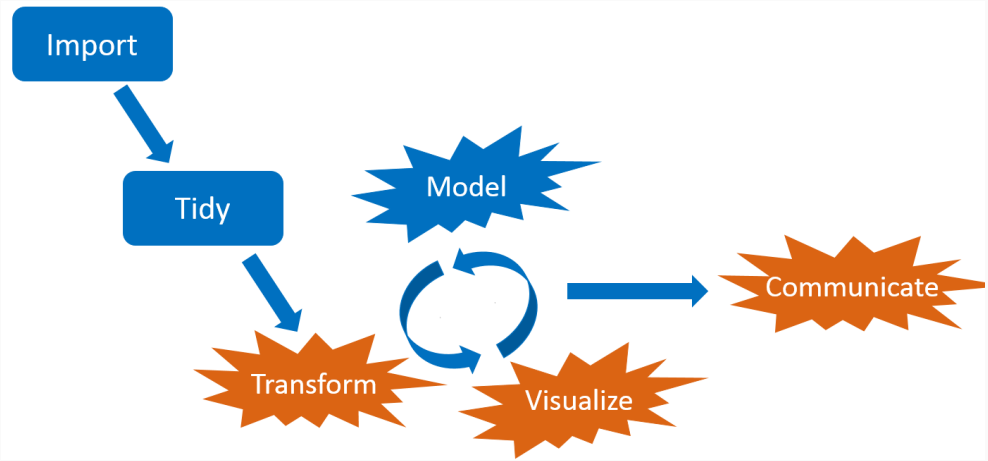
Interconnected tasks



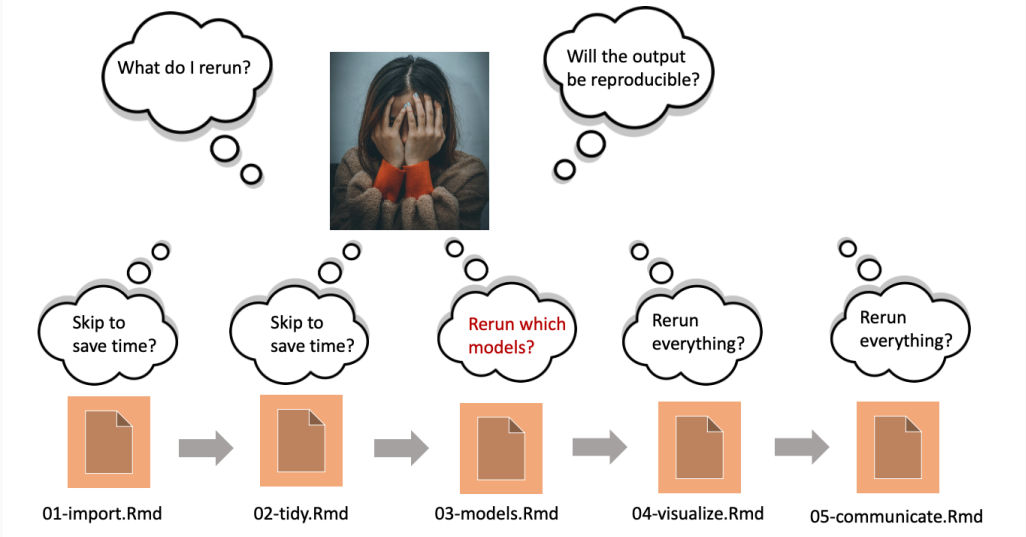
Interconnected tasks



Interconnected tasks

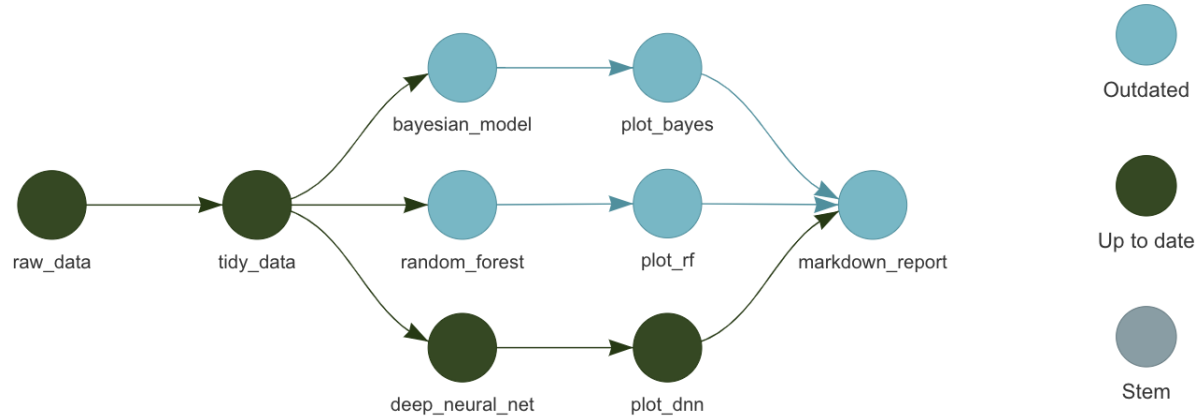


Dilemma: short runtimes or reproducible results?



https://openclipart.org/

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

```
_targets.R # Required top-level configuration file.
R/
└── functions.R
data/
└── my_data.csv
```

_targets.R

```
library(targets)
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))
)
```

Generate `_targets.R` in working directory

```
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.

Debugging

Errored targets to return `NULL` so pipeline continues.

```
tar_option_set(error = "null")
```

Debugging

Errored targets to return `NULL` so pipeline continues.

```
tar_option_set(error = "null")
```

See error messages for all targets.

```
tar_meta(fields = error, complete_only = TRUE)
```

Debugging

Errored targets to return `NULL` so pipeline continues.

```
tar_option_set(error = "null")
```

See error messages for all targets.

```
tar_meta(fields = error, complete_only = TRUE)
```

See warning messages for all targets.

```
tar_meta(fields = warnings, complete_only = TRUE)
```

Debugging

- Try loading all available targets: `tar_load_everything()`. Then run the command of the errored target in the console.
- Pause the pipeline with `browser()`
- Use the debug option: `tar_option_set(debug = "target_name")`
- Save the workspaces:
  - `tar_option_set(workspace_on_error = TRUE)`
  - `tar_workspaces()`
  - `tar_workspace(target_name)`

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 with quarto

```
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.

Exercise

- Add a targets workflow to your quarto document.
- Create a visualization of the pipeline network using `tar_visnetwork()`.

Assignment 4