R coding advice

Ivan Hanigan,
School of Population Health

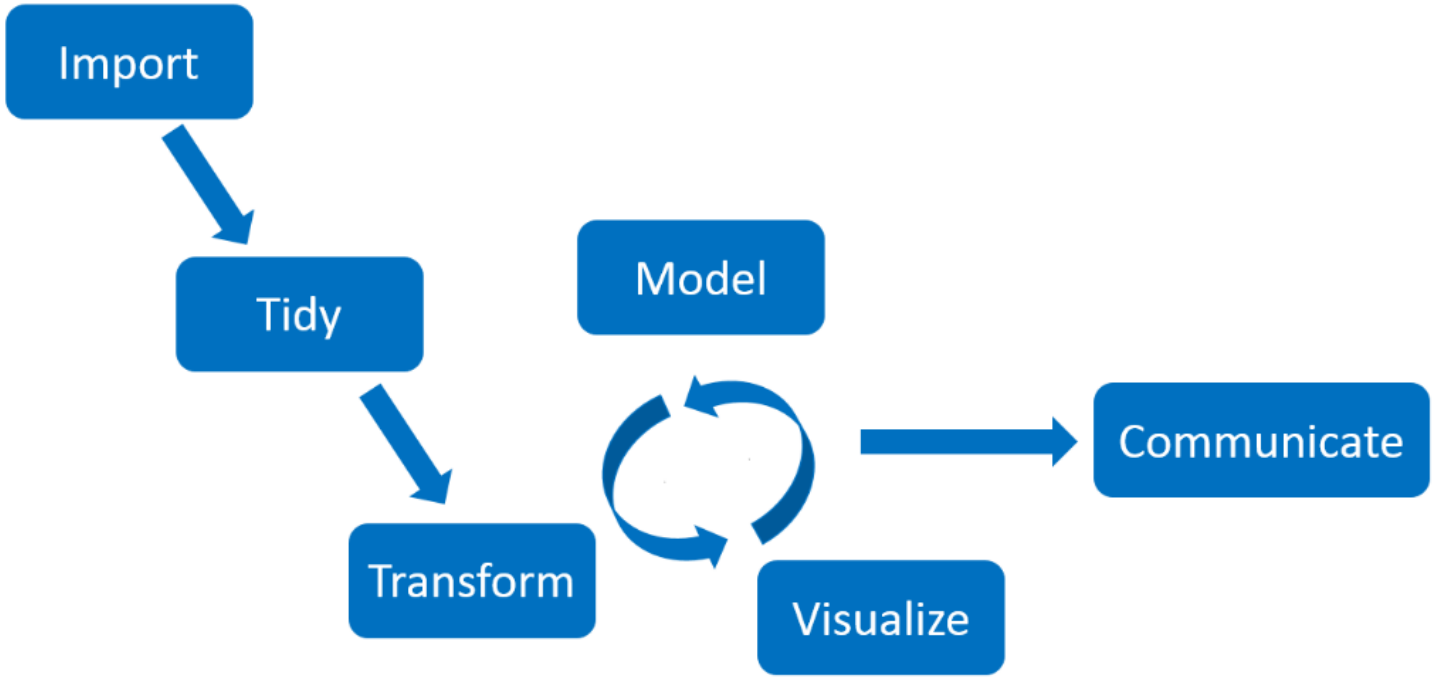
PUBH3007 Climate Change, Health and Big Data

There are better systems than a single script

- The single script approach is limited
- Advantages to shorter scripts include:
  - If you correct an error, you need to verify that none of the later commands are affected
  - It is easier to keep track of corrected results from shorter scripts
  - It is easier to review the results of shorter scripts, especially in collaborative work

- The unit learning material for this module is online: https://cardat.github.io/air-health-sws-r-targets-technique-tweaking-tinkering/index.html

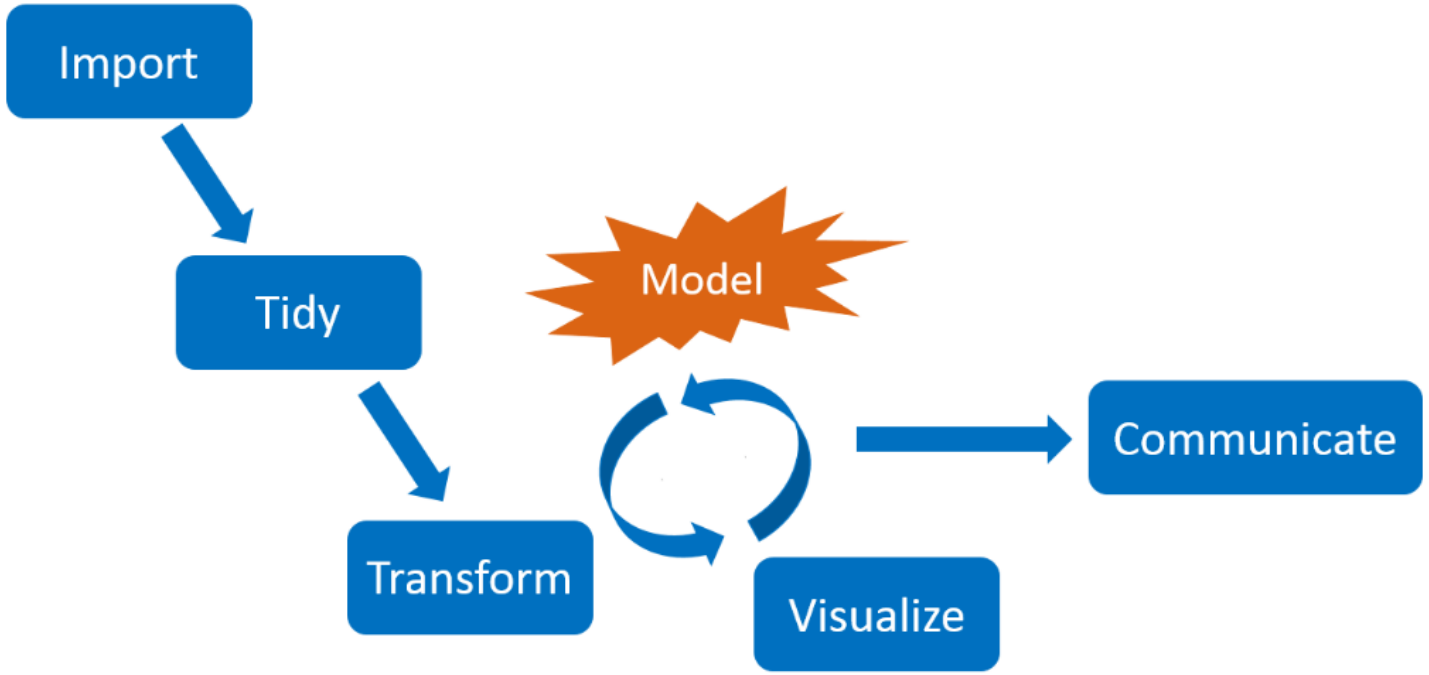
Interconnected tasks



This is called a Workflow

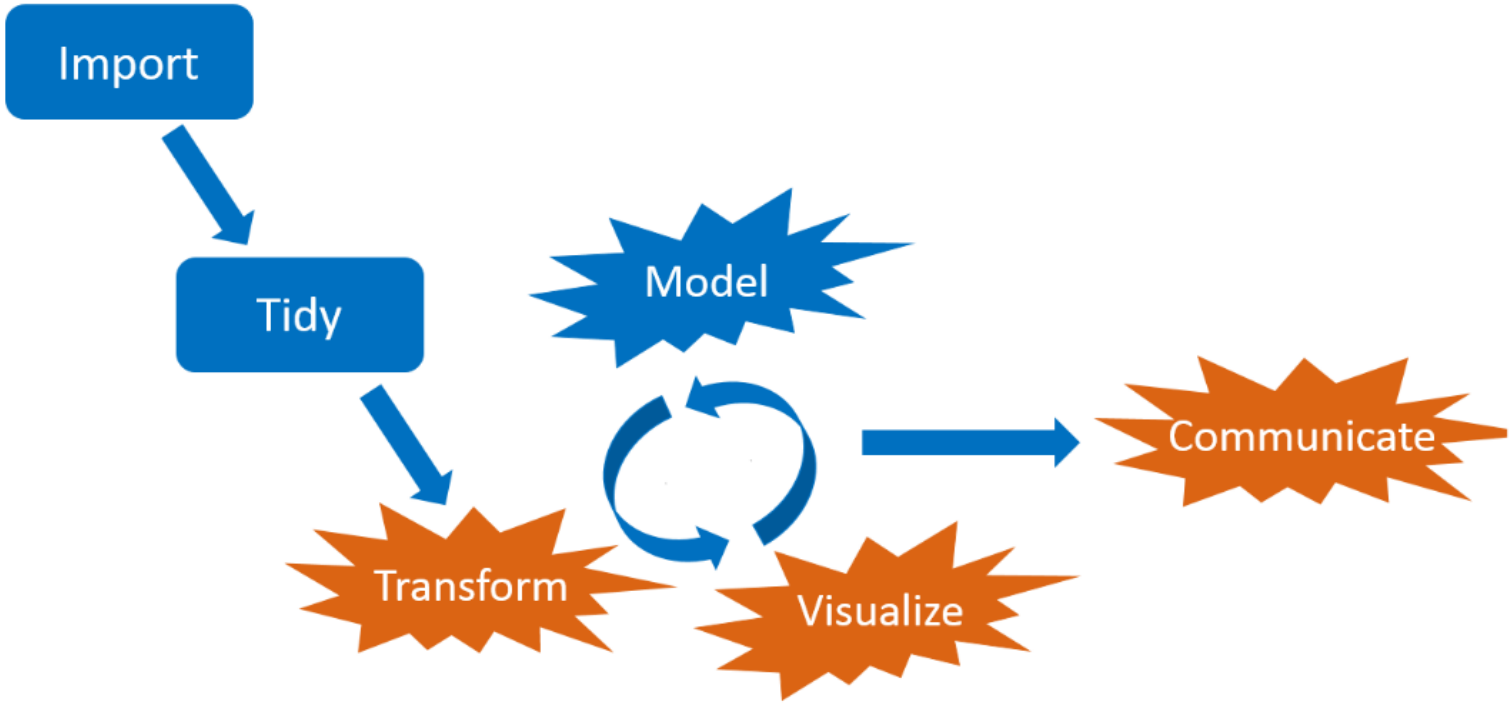
Source: https://wlandau.github.io/targets-tutorial/

Changes



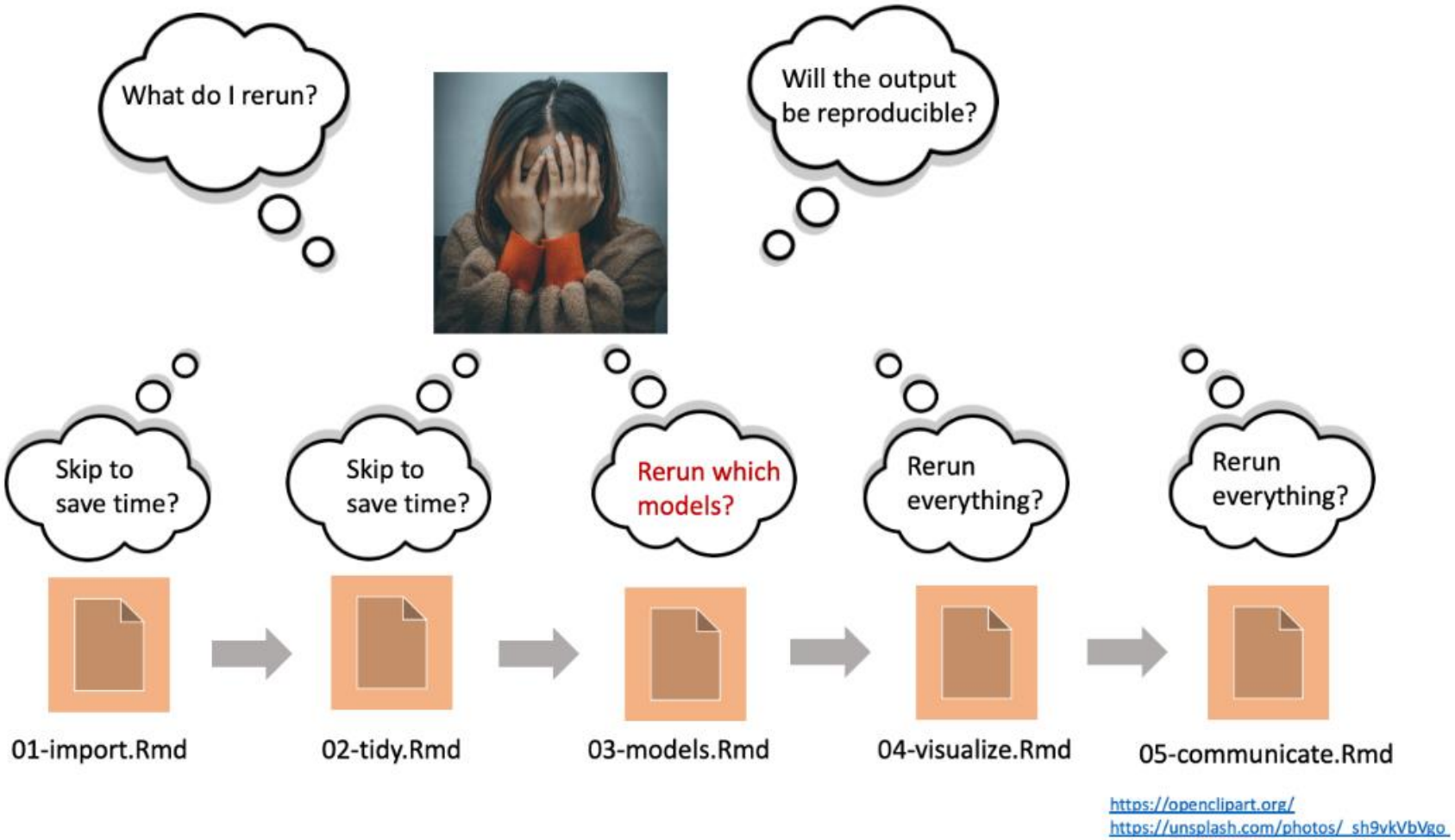
Source: https://wlandau.github.io/targets-tutorial/

Consequences



Source: https://wlandau.github.io/targets-tutorial/

Dilemma:



Source: https://wlandau.github.io/targets-tutorial/

There is a long tradition of using "numbered imperative scripts"

- Parent Directory {The project directory}
  - main.R {This is where you set global variables, and call the other scripts.}
  - data_provided/ {A data directory to hold your data files.}
  - code/ {A code directory containing the other code files.}
    - load.R {This is where you insert code to load in libraries and data files.}
    - clean.R {This is where you perform any messy data cleanup.}
    - func.R {This is where you write all the functions that you need.}
    - do.R {This is where the actual work is done.}

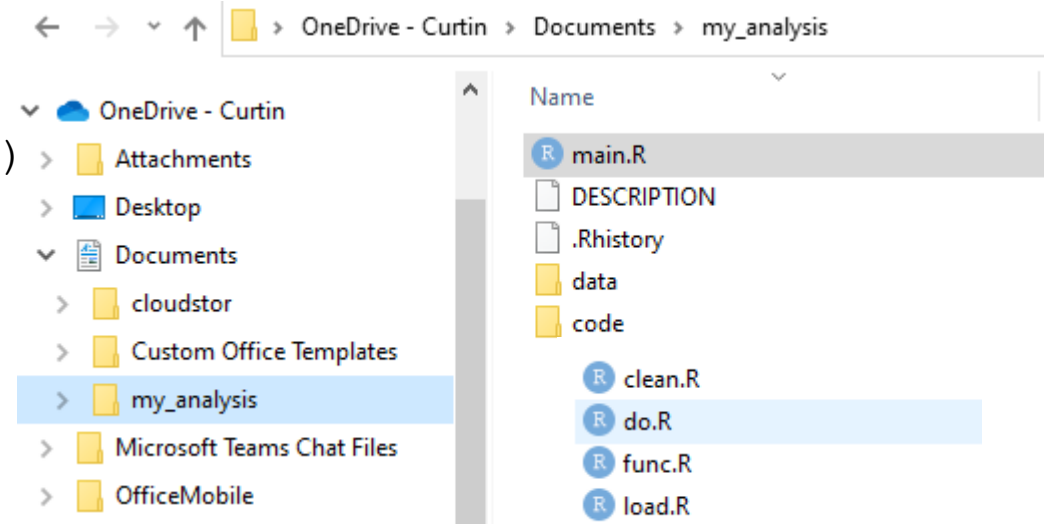
https://github.com/cran/makeProject

```
> install.packages("makeProject")
> makeProject::makeProject("my_analysis")
```

Concept: use small scripts and a ‘master’

https://github.com/cran/makeProject

```
> install.packages("makeProject")
> makeProject::makeProject("my_analysis")
```



- The numbered imperative scripts approach has some nice features
- Start by converting the single script into smaller components

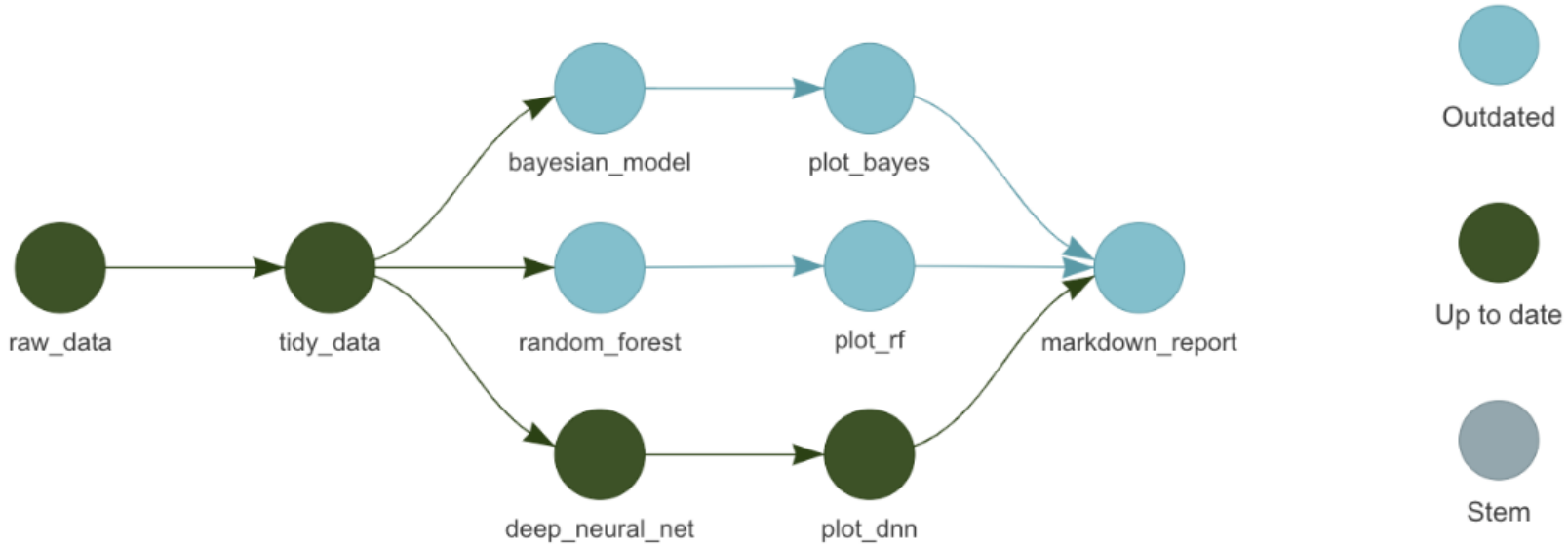
Recommendation:
Keep track of Workflow steps in a table

- In a big project this can be incredibly useful
- You can do this with a pen and paper notebook
- I have mocked up a spreadsheet of what the Case Study could look like

	A	B	C	D	E	F
1	step	code	input	output	notes	status
2	1	functions.R			This is the R packages we need	DONE
3	2	load_data.R	Chicago and LA mort and weather	merged file (indat)	LA data downloaded separately	DONE
4	3	clean_data.R	merged file (indat)	cleaned	Just some formatting	DONE
5	4	do_exploratory_figures.R		simple plots		TODO
6	5	do_PoissonModel.R		the models	the univariate model is wrong	TODO
7	6	do_PostEstimationSummary.R	the models	estat	we see the change in RRs clearly	TODO
8	7	do_Non-linearResponse.R		fit_gam	This is a sophisticated approach	TODO

Finally, The most advanced types of data science projects need serious tools! Use of a ‘pipeline approach’ has emerged as the best practice (R-targets).

Let a pipeline tool figure out what to rerun.



- Save time while ensuring computational reproducibility.
- Automatic parallel/distributed computing based on the directed acyclic graph.

Source: https://wlandau.github.io/targets-tutorial/

Above all else:

Just Know What You Did!

The Readings are essential:

- Peng, R. D. (2015). The reproducibility crisis in science: A statistical counterattack. *Significance*, *12*(3), 30–32. https://doi.org/10.1111/j.1740-9713.2015.00827.x
- Barnes, N. (2010). Publish your computer code: It is good enough. *Nature*, *467*(7317), 753–753. https://doi.org/10.1038/467753a