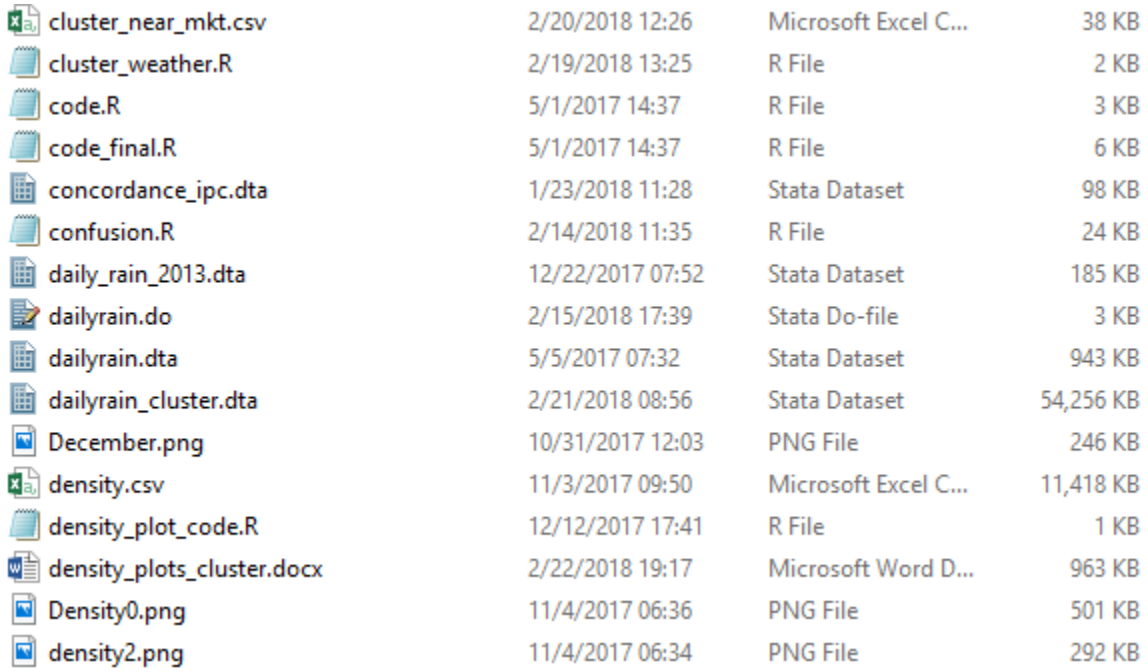
targets

Reproducible computation at scale in R

LSHTM R user group talk by Julian Matthewman

The case for targets

1. Data analysis can get messy



cluster_near_mkt.csv	2/20/2018 12:26	Microsoft Excel C...	38 KB
cluster_weather.R	2/19/2018 13:25	R File	2 KB
code.R	5/1/2017 14:37	R File	3 KB
code_final.R	5/1/2017 14:37	R File	6 KB
concordance_ipc.dta	1/23/2018 11:28	Stata Dataset	98 KB
confusion.R	2/14/2018 11:35	R File	24 KB
daily_rain_2013.dta	12/22/2017 07:52	Stata Dataset	185 KB
dailyrain.do	2/15/2018 17:39	Stata Do-file	3 KB
dailyrain.dta	5/5/2017 07:32	Stata Dataset	943 KB
dailyrain_cluster.dta	2/21/2018 08:56	Stata Dataset	54,256 KB
December.png	10/31/2017 12:03	PNG File	246 KB
density.csv	11/3/2017 09:50	Microsoft Excel C...	11,418 KB
density_plot_code.R	12/12/2017 17:41	R File	1 KB
density_plots_cluster.docx	2/22/2018 19:17	Microsoft Word D...	963 KB
Density0.png	11/4/2017 06:36	PNG File	501 KB
density2.png	11/4/2017 06:34	PNG File	292 KB

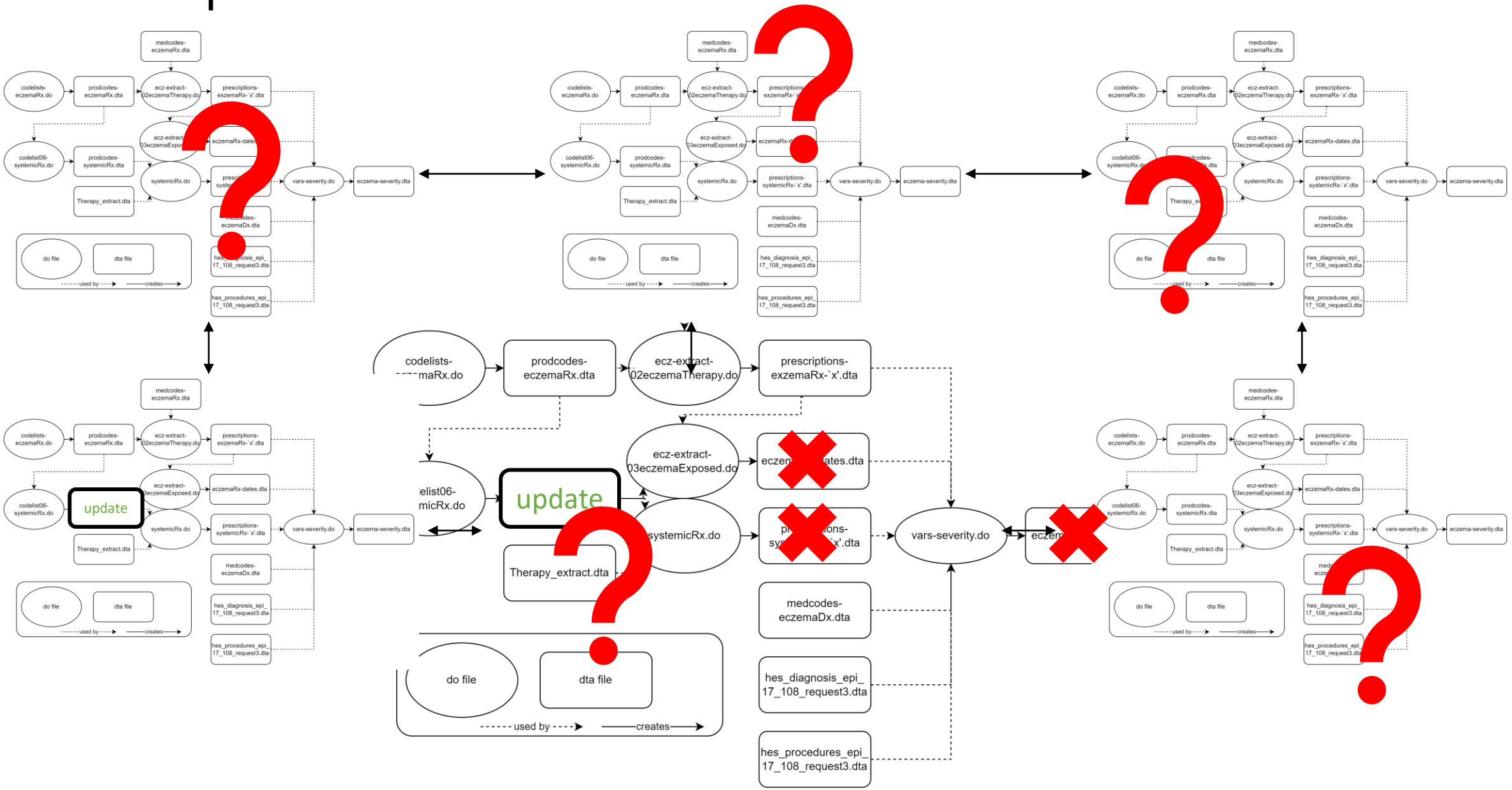
From https://www.yujun.org/post/project/

A reproducible workflow

master.R

```
source 01_import.R
source 02_clean.R
source 03_model.R
source 04_visualise.R
source 05_report.R
```

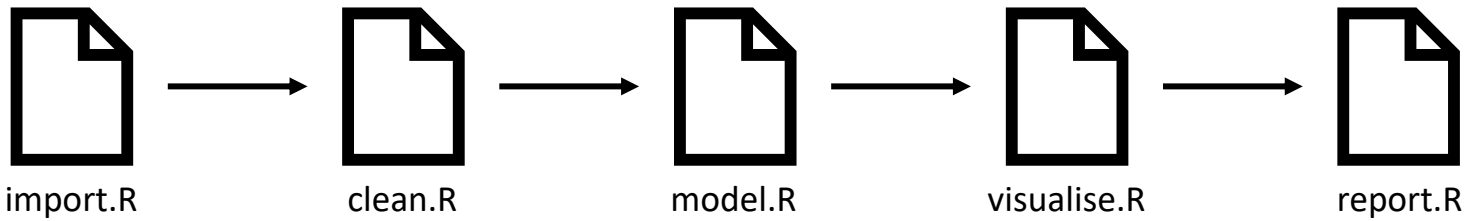
Complex workflows



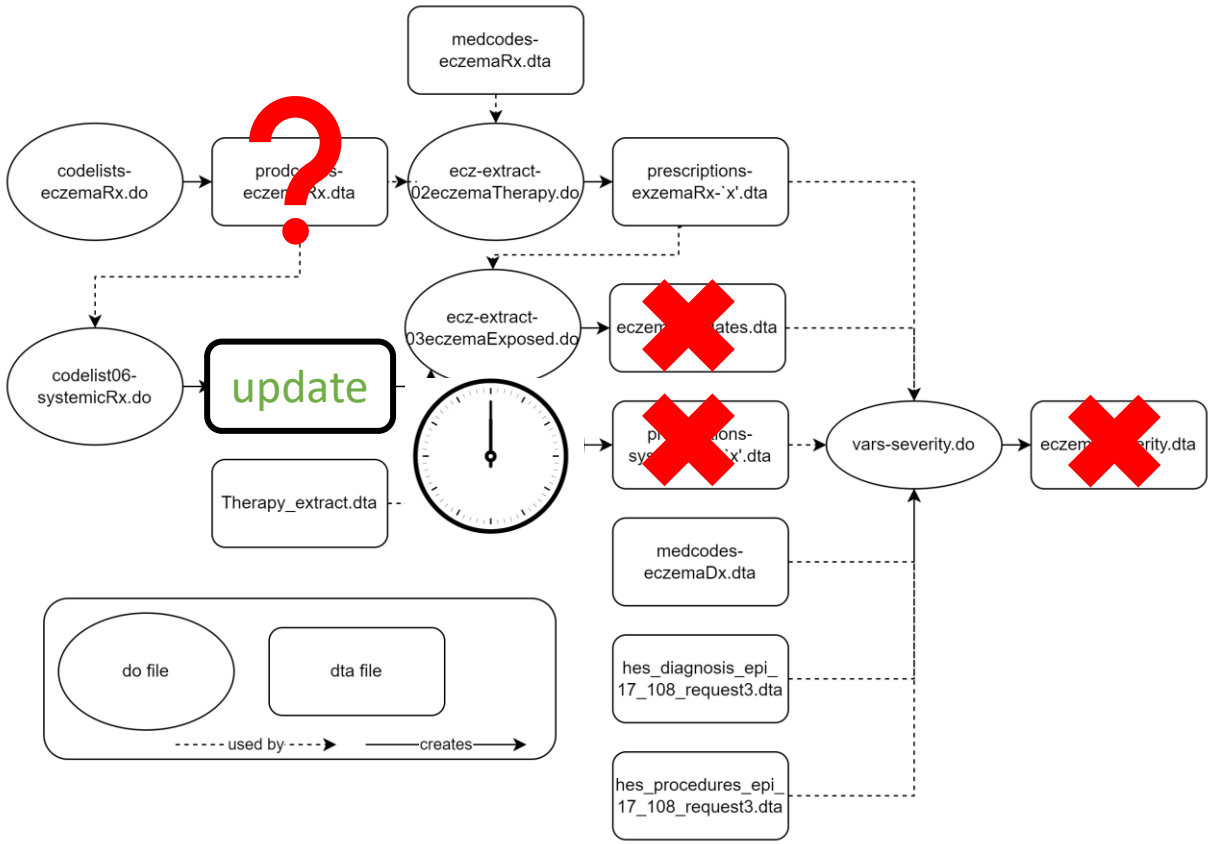
A reproducible workflow

master.R

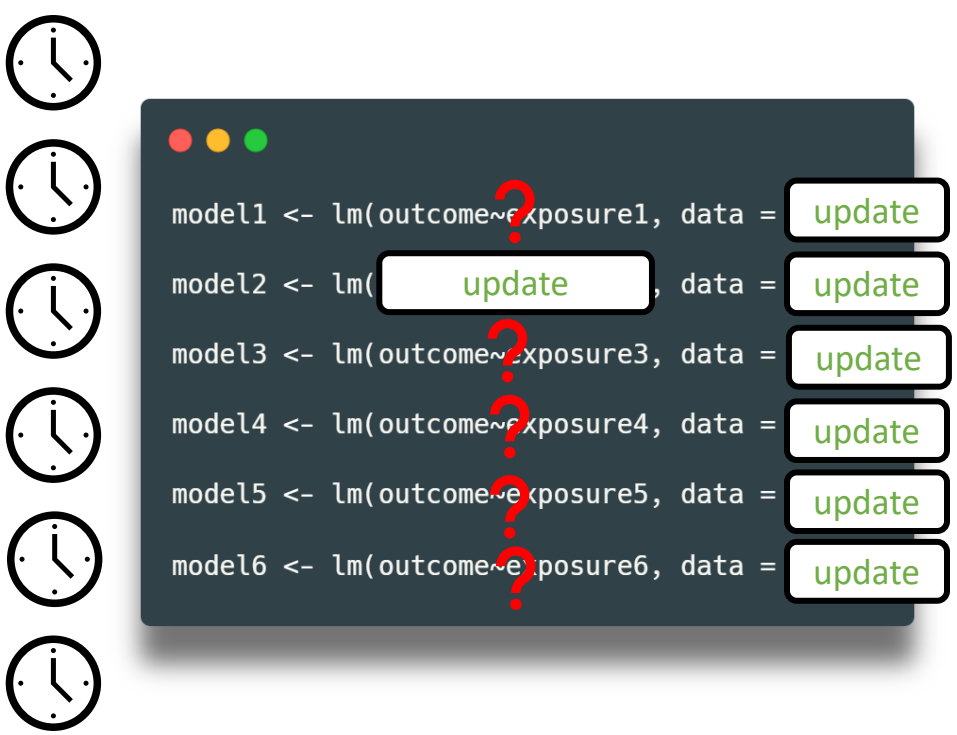
```
source 01_import.R
source 02_a_extract.R
source 02_b_codelists.R
source 02_c_merge.R
source 02_d_algorithm.R
source 03_model.R
source 04_visualise.R
source 05_report.R
```

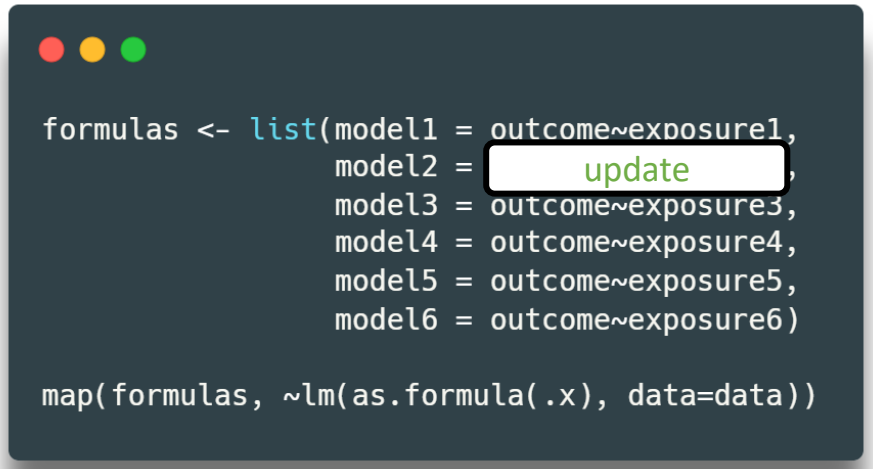


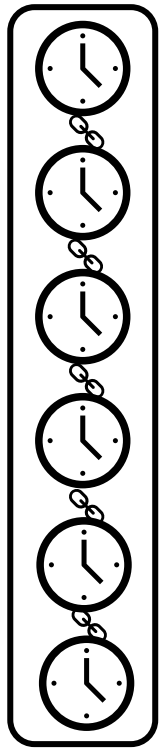
2. Data analysis can be slow



Many models







(Hypothetical) time savings

Reproducibility

Big data and slow methods

- Large data
  - Large population based health data
  - Genomics data
  - Biobanks
  - Climate data
  - …
- Large statistical computation
  - Bayesian data analysis
  - Bayesian network meta-analysis
  - Graph-based multiple comparison procedures
  - Subgroup identification
  - Predictive modeling
  - Deep neural networks
  - PK/PD modeling
  - Clinical trial simulation
  - Target identification

The overlooked bane of long computation

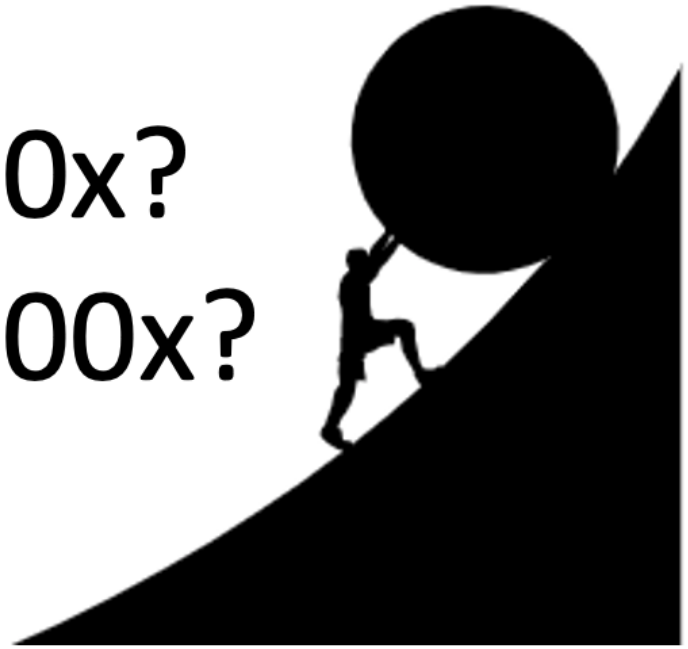
Perception

1x

Reality

10x?

100x?

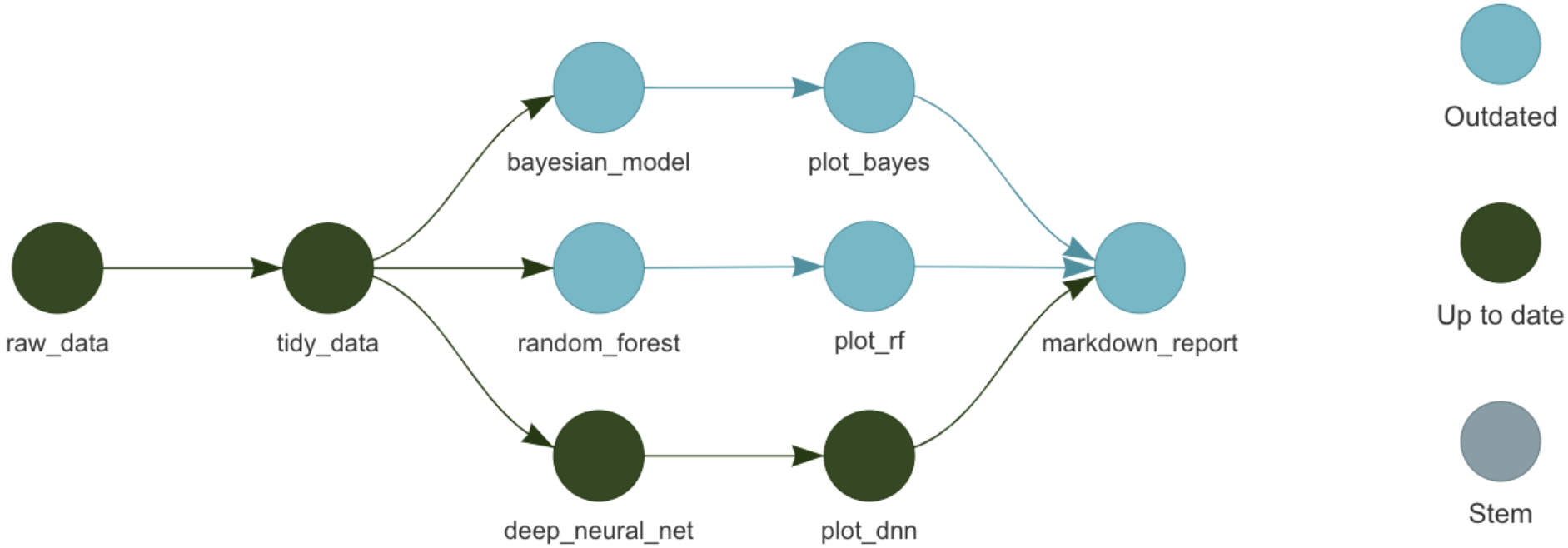


https://openclipart.org/


Re-run everything?
What do I re-run?
Skip to save time?
Is this up to date?
import.R
clean.R
model.R
visualise.R
report.R

The solution: targets

Let a pipeline tool figure out what to rerun



Pipeline tools

- Language agnostic pipeline tools such as GNU make
- Targets
  - Fundamentally designed for R.
  - Supports a clean, modular, function-oriented programming style.
  - Abstracts files as R objects and automatically manages data.

Function-oriented programming style

- Functions take inputs to produce **one** output
- Good fit for organising data analysis

```r
import <- function(raw_data) {...}
clean <- function(imported_data) {...}
model <- function(cleaned_data) {...}
visualise <- function(model_data) {...}

imported_data <- import(raw_data)
cleaned_data <- clean(imported_data)
model_data <- model(cleaned_data)
plots <- visualise(model_data)

import(raw_data) %>%
    clean() %>%
    model() %>%
    visualise()
```

In R:
•Everything that exists is an object.
•Everything that happens is a function call.

John Chambers

As targets

```
import <- function(raw_data) {...}
clean <- function(imported_data) {...}
model <- function(cleaned_data) {...}
visualise <- function(model_data) {...}

list(
tar_target(imported_data, import(raw_data))
tar_target(cleaned_data, clean(imported_data))
tar_target(model_data, model(cleaned_data)),
tar_target(plots, visualise(model_data))
)
```

Folder structure

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

Folder structure

- `_targets.R # Required top-level configuration file.`
- `R/`
- `└── import.R`
- `└── clean.R`
- `└── model.R`
- `└── visualise.R`
- `data/`
- `└── huge_dataset.csv`

_targets.R

```r
# _targets.R
library(targets)
source("R/functions.R")
tar_option_set(packages = c("tidyverse", "broom"))

list(
  tar_target(imported_data, import(raw_data)),
  tar_target(cleaned_data, clean(imported_data)),
  tar_target(model_data, model(cleaned_data)),
  tar_target(plots, visualise(cleaned_data))
  )
```

tar_make()

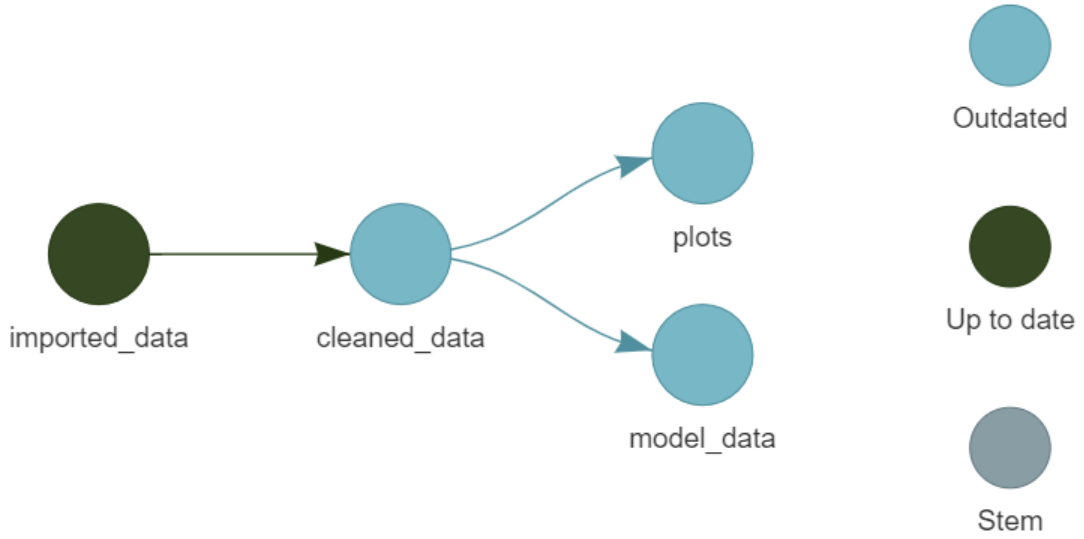
```
> tar_make()
* start target imported_data
* built target imported_data
* start target cleaned_data
* built target cleaned_data
* start target model_data
* built target model_data
* start target plots
* built target plots
* end pipeline
```

tar_read()

```
> tar_read(cleaned_data)
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Fiat 128         32.4   4 78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic      30.4   4 75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla   33.9   4 71.1  65 4.22 1.835 19.90  1  1    4    1
Lotus Europa     30.4   4 95.1 113 3.77 1.513 16.90  1  1    5    2

> tar_load(cleaned_data) # To load into memory
```

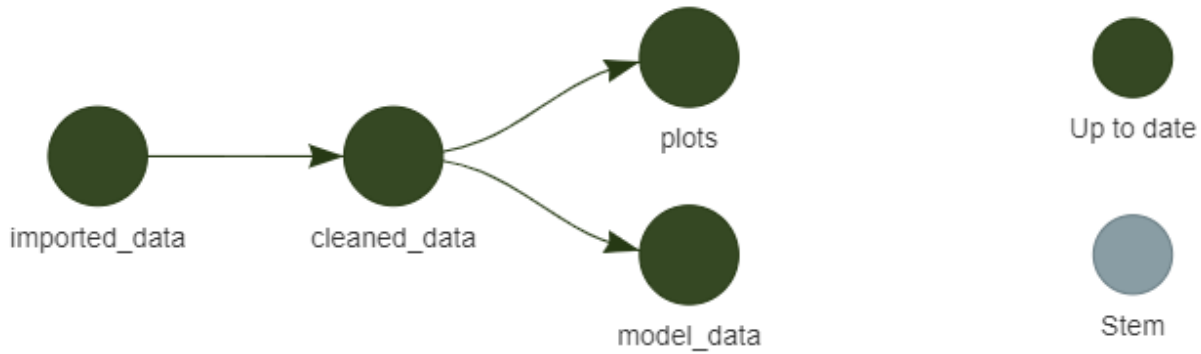
tar_visnetwork()



tar_make()

```
> tar_make()
✓ skip target imported_data
* start target cleaned_data
* built target cleaned_data
* start target model_data
* built target model_data
* start target plots
* built target plots
* end pipeline
```

tar_visnetwork()



“When all targets are up to date, this is evidence that the results match the underlying code and data, which helps us trust the results and confirm the computation is reproducible.”

Many models - Branching

```
tar_target(formulas,
           c("mpg ~ hp",
             "mpg ~ cyl",
             "mpg ~ drat")),

tar_target(model_data,
           model(cleaned_data, formulas),
           pattern = formulas)
```

```
> tar_make()
* start branch model_data_63cc8298
* built branch model_data_63cc8298
✓ skip branch model_data_f50cdca0
✓ skip branch model_data_87695c67d
* built pattern model_data
```

```
formulas <- list(model1 = outcome~exposure1,
                 model2 = outcome~exposure2,
                 model3 = outcome~exposure3,
                 model4 = outcome~exposure4,
                 model5 = outcome~exposure5,
                 model6 = outcome~exposure6)

map(formulas, ~lm(as.formula(.x), data=data))
```

Time savings  Reproducibility

Caveats

- Learning curve
- Not worth the time investment for small data/fast methods
- Need to adopt functional programming style (this is good actually)
- Can make debugging more difficult

There’s more

- File tracking
- R Markdown integration
- High performance computing
- Cloud storage
- Dynamic and static branching
- Tarchetypes (express complicated pipelines with concise syntax)

Resources

- Website: https://docs.ropensci.org/targets/
  - With “How to get started” Section
- Short course: https://github.com/wlandau/targets-tutorial/
  - Learn about Functions, Pipelines, Changes, Debugging, Files, Branching
- Manual: https://books.ropensci.org/targets/
- Images and Inspiration for this presentation from https://wlandau.github.io/targets-tutorial/

Demo Time!