



Cabinet Office

Matt Dray

Data Scientist

Government Digital Service

@mattdray_gds 

Can {drake} RAP?



Drake



drake ()

Via [Garrick Aden-Buie](#)

tl;dr

There are lots of workflow tools





Scale the work
you need.



Skip the work
you don't.



See evidence
of reproducibility.

Via [the {drake} manual](#)

Materials

This talk:

- [a blog post](#)
- [code for the demo](#)

For {drake}:

- [visit the website](#)
- [read the full manual](#)
- [learn from a course](#)
- [use it in an app](#)



[Will Landau](#)

Workflows

Inputs → 'Stuff happens' → Outputs

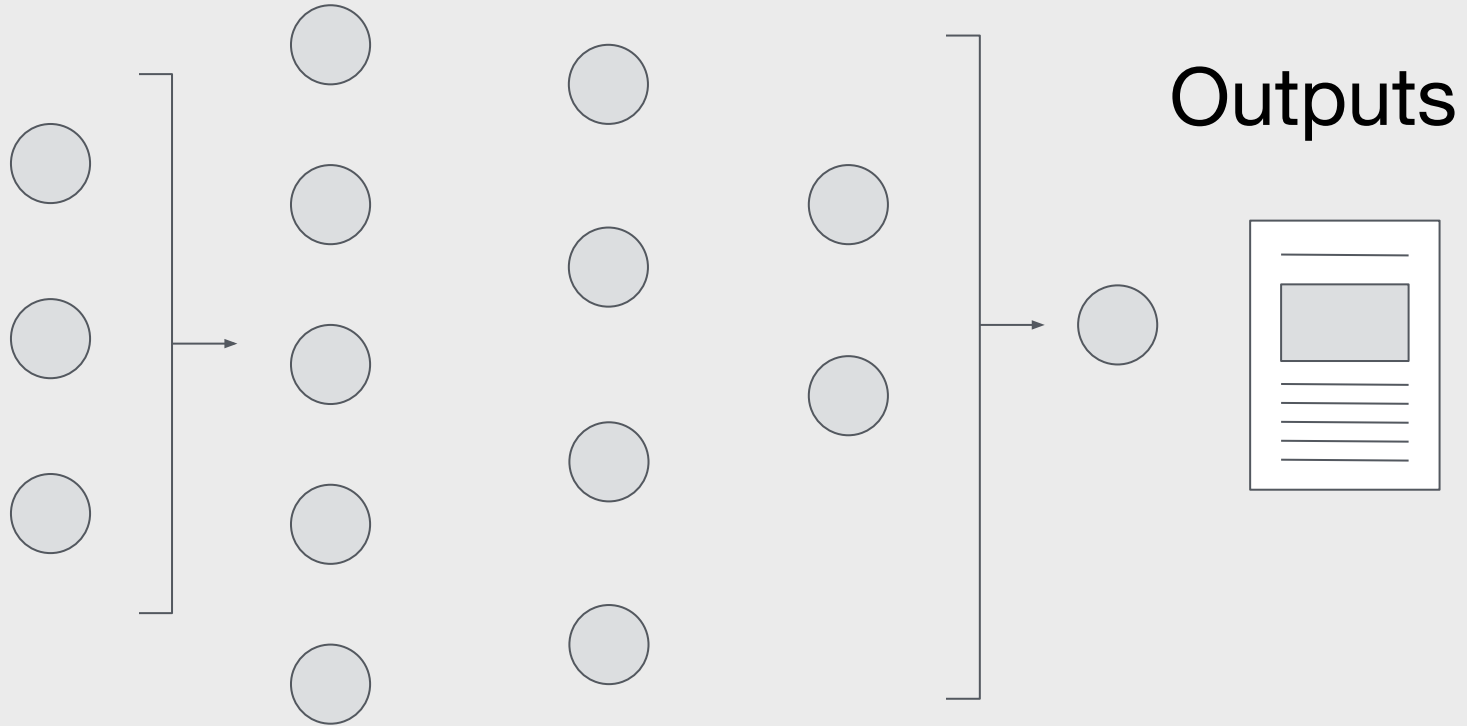
01010110
10101101
01000010
11101010
10101011
11010010



'Stuff happens'

Inputs

01010110
10101101
01000010
11101010
10101011
11010010



Changed



What happens if you
change something?

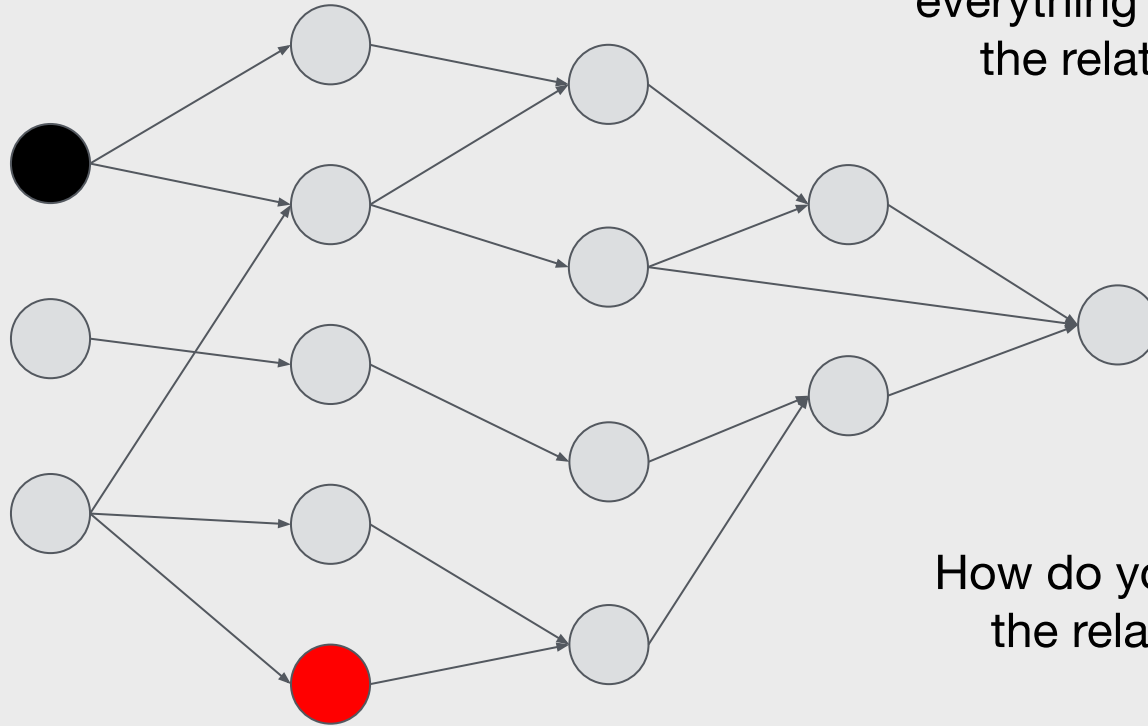
Re-run everything
from scratch?



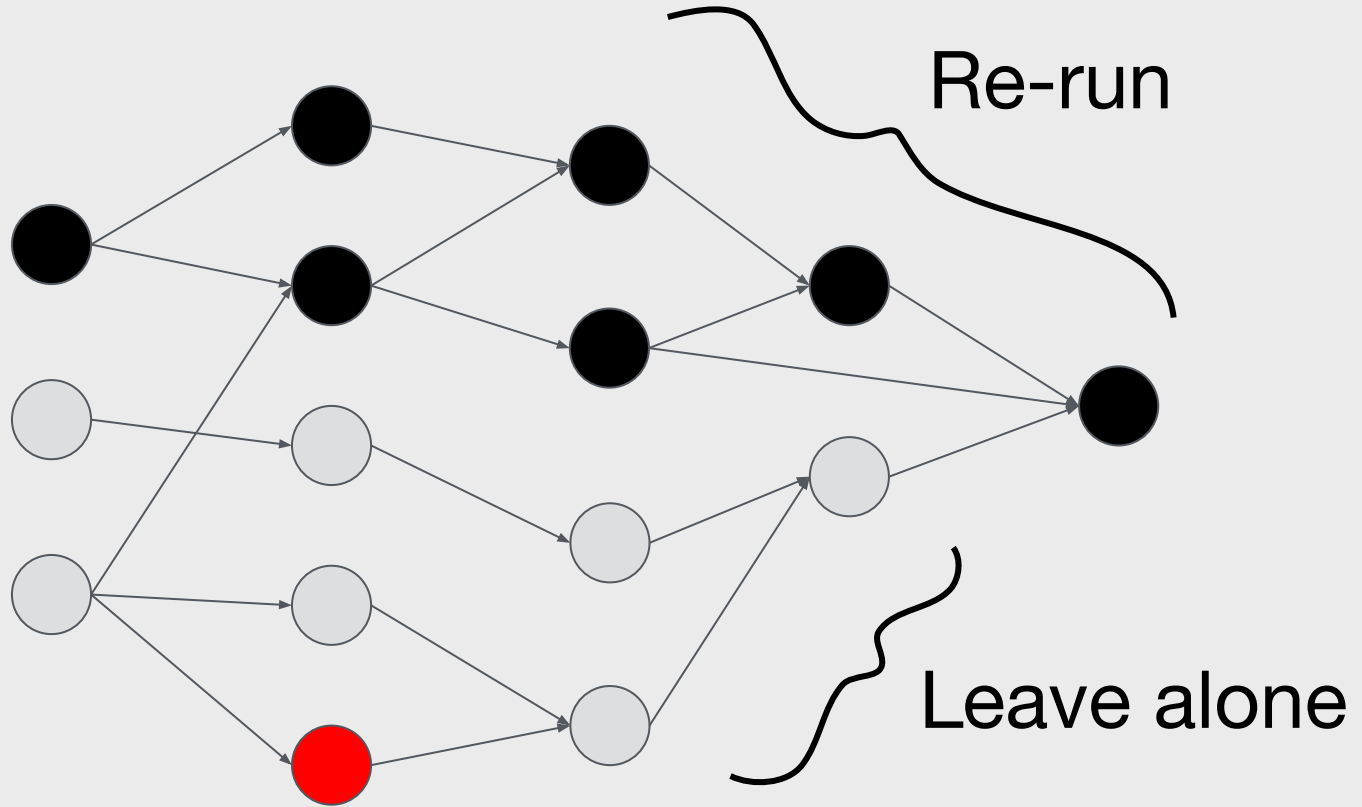
Expensive



What if something
takes hours to run?



How do you remember the relationships?



Solution:
{drake}



- Saves computation
- Remembers for you
- Improves reproducibility
- Visualises dependencies
- Allows for parallelisation
- Lets you do it from R

{drake} workflow

1. Create scripts and plan
2. Make the plan
3. Change stuff and re-make

When to {drake}?

01_read.R
02_import.R
03_tidy.R
04_clean.R
05_model.R
06_plot.R
07_report.Rmd

A good start to
organising your
workflow

But why isn't it
optimal?

RAP is for:

- reproducibility
- automation
- minimising error
- doing it faster
- building trust



BETA This is new – your [feedback](#) will help us to improve it.

RAP: Reproducible Analytical Pipelines

The Reproducible Analytical Pipeline (RAP) is a methodology for automating the bulk of steps involved in creating a statistical report.

RAP is also a community of people who work with data using methods adapted from software development. The RAP community promotes the use of programming languages, version control, automated testing, and other tools and methods.

This website is a place for the community to publish materials that it finds useful. Materials that include code can be published on this website.

Other RAP websites

The [RAP Champions](#) are a group of people, including the Government Statistical Service, who maintain a list of people to contact for help. They also maintain a [list of resources](#), and links to blog posts, guides and courses.

How to contribute to this website

You can contribute to this website by discussing it in the [Slack channel](#) ([#rap_collaboration](#)), or by opening an [issue on GitHub](#).

The website is built using the R package [govdown](#). It supports code written in R, Python, and can support other languages that the knitr package supports, as long as [Travis](#) is able to run the code to build the website.

Attribution

The warp pipe logo by <http://delapouite.com/> was obtained from <https://game-icons.net/1x1/delapouite/warp-pipe.html> licensed CC BY 3.0 <http://creativecommons.org/licenses/by/3.0/> and is used unaltered.

BETA This is new – your [feedback](#) will help us to improve it.

Egg stats:

- [publication](#)
- [my code](#)
- [my report](#)

UK egg statistics

Background

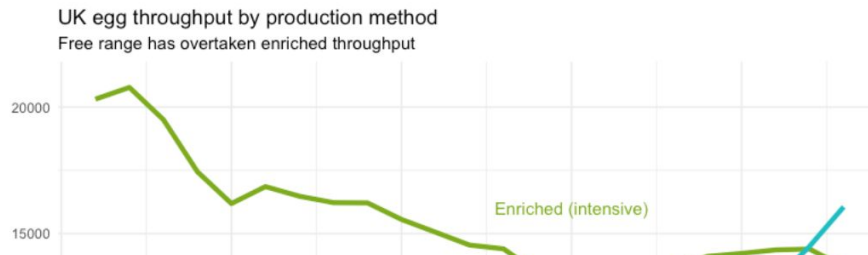
The '[latest UK egg statistics](#)' [publication](#) contains the latest quarterly UK statistics about eggs.

It's published by the [Department for Environment, Food and Rural Affairs](#).

This report is the output from a demo of using [the {drake} package](#) for R. It's not an official government publication.

Throughput

Below is a recreation of Figure 2 from the [UK egg statistics notice](#) document. It shows egg production over time, split by production methods.



Live demo

Access the [demo code](#) in
RStudio in your browser:



Step-by-step

{drake} workflow

1. Create scripts and plan
2. Make the plan
3. Change stuff and re-make

{drake} workflow


1. Create scripts and plan
2. Make the plan
3. Visualise
4. Change stuff
5. Check changes
6. Re-make

1. Create scripts and plan

This is like prepping ingredients

packages.R


```
library(drake)  
library(dplyr)  
library(readr)  
library(ggplot2)  
...
```



Load packages

functions.R

```
clean_data <- function(raw_data) { ... }  
create_plot <- function(data) { ... }
```



Create functions

This is like writing a recipe

```
# plan.R
```

```
plan <- drake_plan(  
  
  raw_data = read_csv("data.csv"),  
  data = clean_data(raw_data),  
  plot = create_plot(data),  
  
  report = render(  
    knitr_in("report.Rmd"),  
    output_file = file_out("report.html")  
  )  
)
```

Declare
targets and
commands

Prepare
steps into
a plan
dataframe

Special `knitr_in()` and `file_out()`
functions to mark dependencies

2. Make the plan

*This is like making the recipe
and getting a delicious cake*

This script orchestrates everything

make.R

```
source(R/packages.R)  
source(R/functions.R)  
source(R/plan.R)
```

} Source the R scripts

```
make(plan)
```


Generates a dataframe of the
targets and commands

Make the plan – your output will be
created at the filepath specified in the plan

3. Visualise

```
config <- drake_config(plan)
```

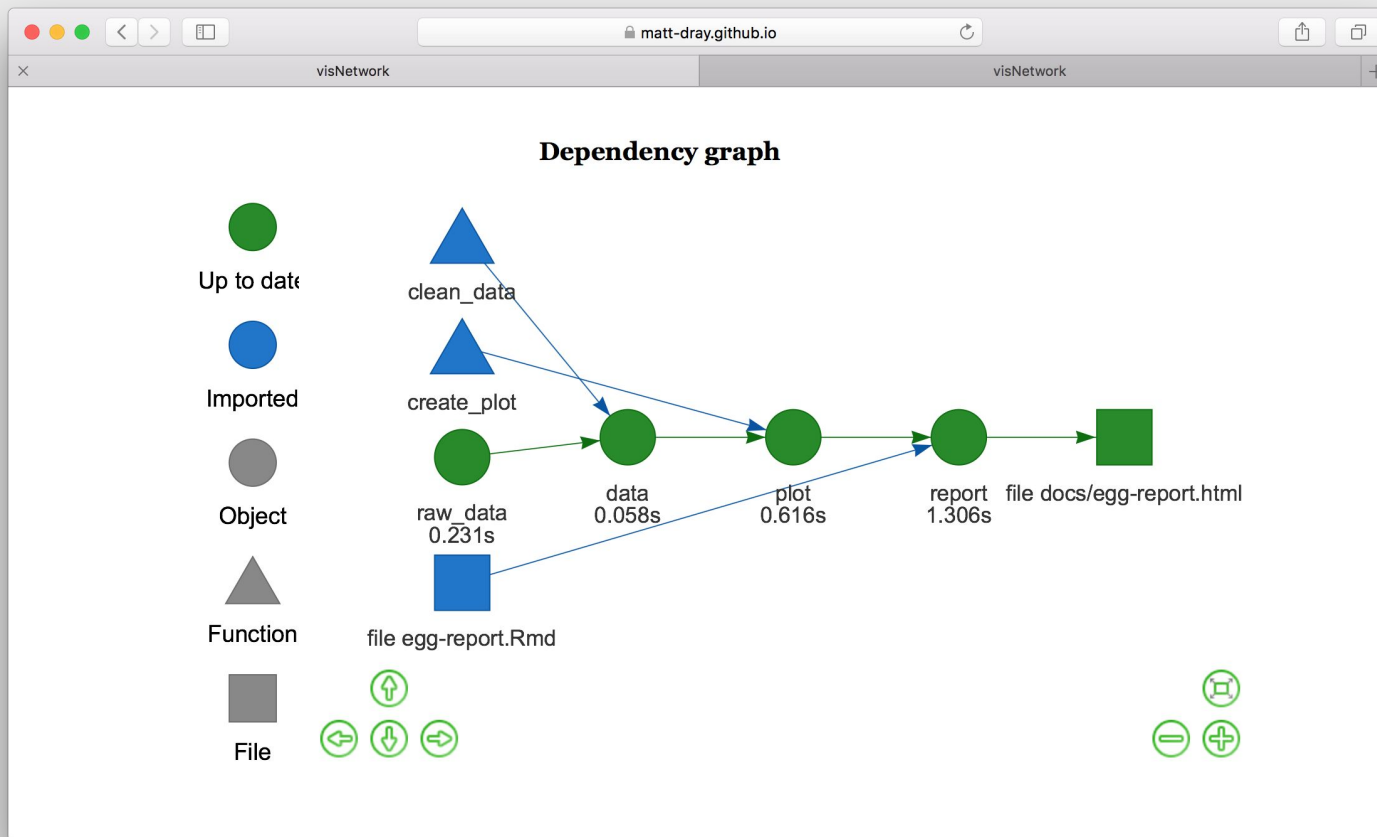
Create configuration file
(contains graph element)



```
vis_drake_graph(config)
```

Create plot from config





[Live interactive version](#)

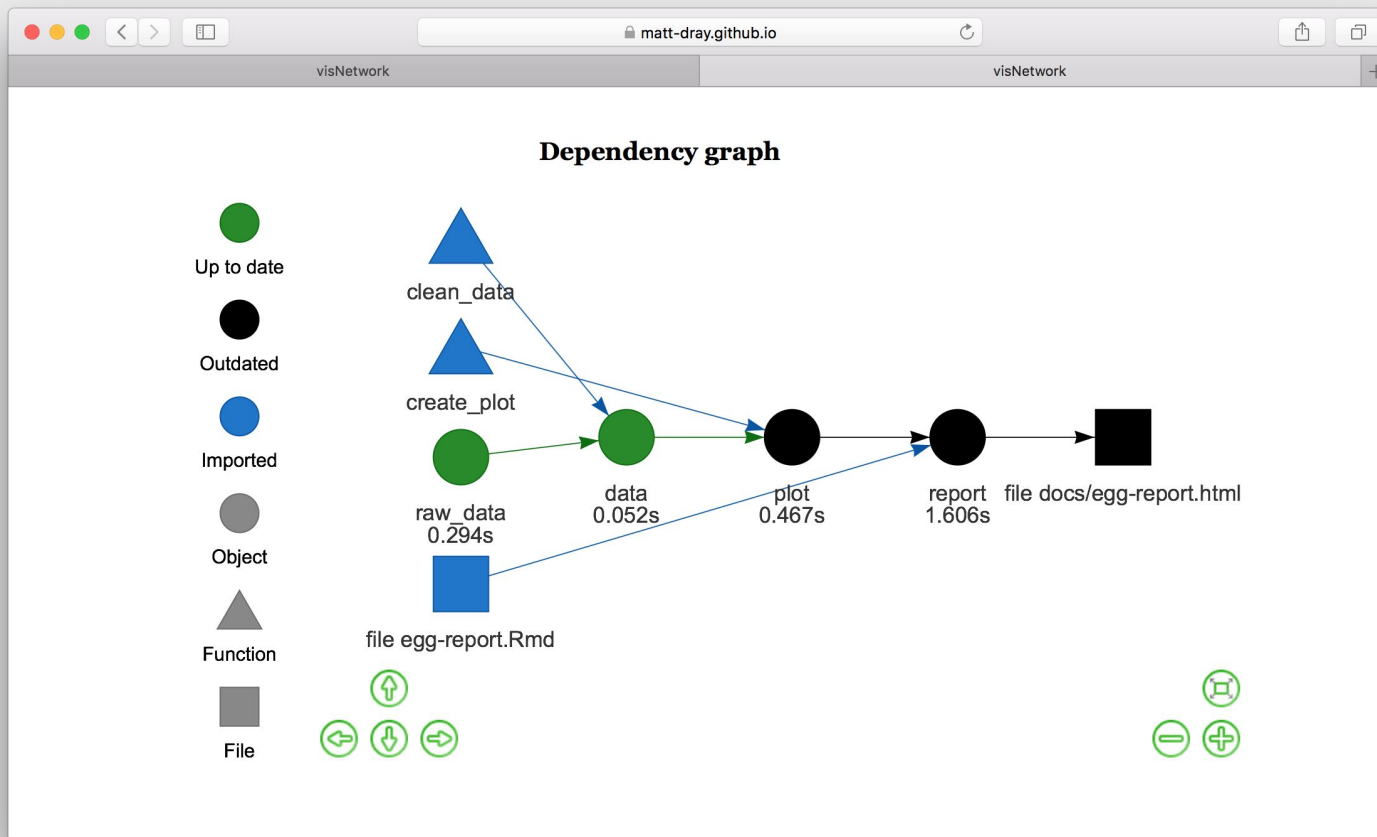
4. Change stuff
5. Check changes

Make a **change**, e.g. in functions.R

source(functions.R)  Re-source the change

outdated(config)  Check the outdated targets

config <- drake_config(plan)
vis_drake_graph(config) } Recreate config, re-visualise



[Live interactive version](#)

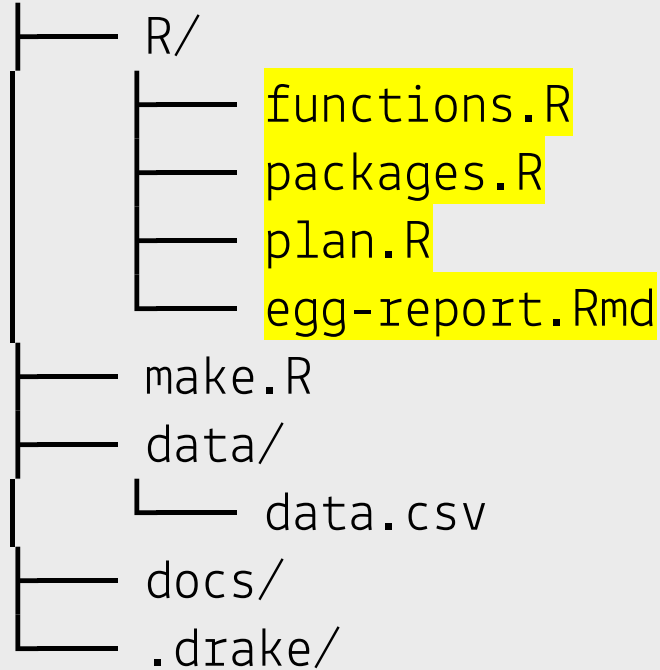
6. Re-make

make(plan) ← Re-make() the plan and the
output will be updated

Only the things that need
re-running will be re-run

Folder view

drake-egg-rap/



} R scripts in R/ folder

drake-egg-rap/

- R/
 - functions.R
 - packages.R
 - plan.R
 - egg-report.Rmd

make.R

data/

- data.csv

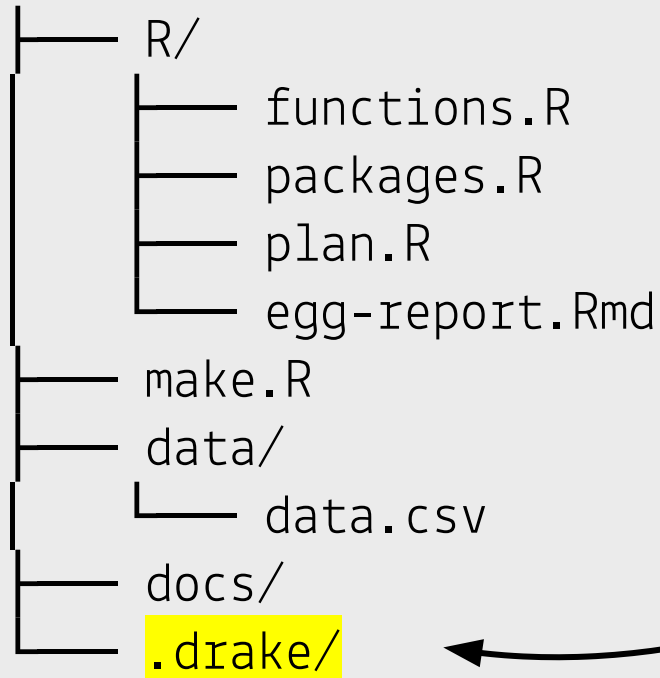
docs/

.drake/

Make script in root



drake-egg-rap/



Hidden cache — basically
drake's 'brain' for storing
relationships and objects



Hall of fame





Cabinet Office

Matt Dray

@mattdray_gds 