Reproducible pipelines in R

with `targets`

HPC SChool 2021 PS11: R session

A. Ginolhac

DLSM University of Luxembourg

High Performance Computing & Big Data Services

hpc.uni.lu

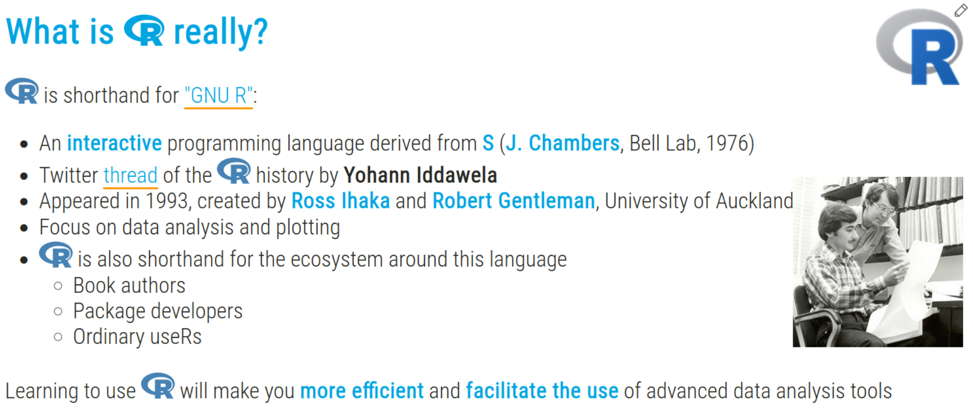
hpc@uni.lu

@ULHPC

Introduction to R

Not the scope of this session

Beginner user, check out this lecture



What is R really?

R is shorthand for "GNU R":

- An **interactive** programming language derived from **S** (**J. Chambers**, Bell Lab, 1976)
- Twitter thread of the R history by **Yohann Iddawela**
- Appeared in 1993, created by **Ross Ihaka** and **Robert Gentleman**, University of Auckland
- Focus on data analysis and plotting
- R is also shorthand for the ecosystem around this language
  - Book authors
  - Package developers
  - Ordinary useRs

Learning to use R will make you **more efficient** and **facilitate the use** of advanced data analysis tools

Advanced user, interested in programming, check out this lecture

Evaluation in programming

`tidyeval`

A. Ginolhac | rworkshop | 2020-11-30

targets

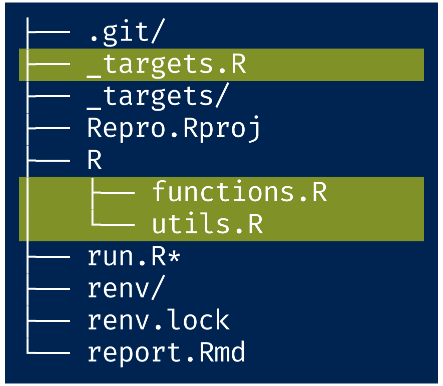
a Make-like workflow manager for R

targets and companion package tarchetypes

A workflow manager for R

- Saving you time and stress
- Understand how it is implemented in `targets`
  - Define your `targets`
  - Connect `targets` to create the **dependencies**
  - Check **dependencies** with `visnetwork`
  - Embrace **dynamic** branching
  - Run **only** what needs to be executed
  - Bundle **dependencies** in a Rmarkdown document with `tar_render()`
  - Increase reproducibility with the package manager `renv`
- Example with RNA-seq data from **Wendkouni Nadège MINOUNGOU**

Folder structure



```
├── .git/
├── _targets.R
├── _targets/
├── Repro.Rproj
├── R
│   ├── functions.R
│   └── utils.R
├── run.R*
├── renv/
├── renv.lock
└── report.Rmd
```

- With renv. Snapshot your package environment (and restore! 😌)
- _targets.R is the only mandatory file
- Use a R sub-folder for functions, gets closer to a R package
- In a RStudio project
- Version tracked with git
- Rmarkdown file allows to gather results in a report
- Optional: an executable run.sh allows to use Build Tools in RStudio

Targets Markdown

Bundle globals and pipeline inside a **Rmarkdown** document.

- Makes development easier
- Documentation can be embedded
- targets engine recognizes by knitr and takes care of writing all R scripts

renv features

- hydrate() parses your code and finds library calls
- install() from **CRAN** with dependencies (also from)
- snapshot() registers changes, hashes and origin
- restore() to a certain point in time

```
> renv::snapshot()
The following package(s) will be updated in the lockfile:

# CRAN ===============================
- RcppParallel   [5.0.2 -> 5.0.3]
- cli            [2.3.0 -> 2.3.1]
- pkgload        [1.1.0 -> 1.2.0]
- tint           [0.1.3 -> *]

# GitHub =============================
- targets        [ropensci/targets@main: 598d7a23 -> bdc1b29c]

Do you want to proceed? [y/N]:
```

renv.lock file after a snapshot

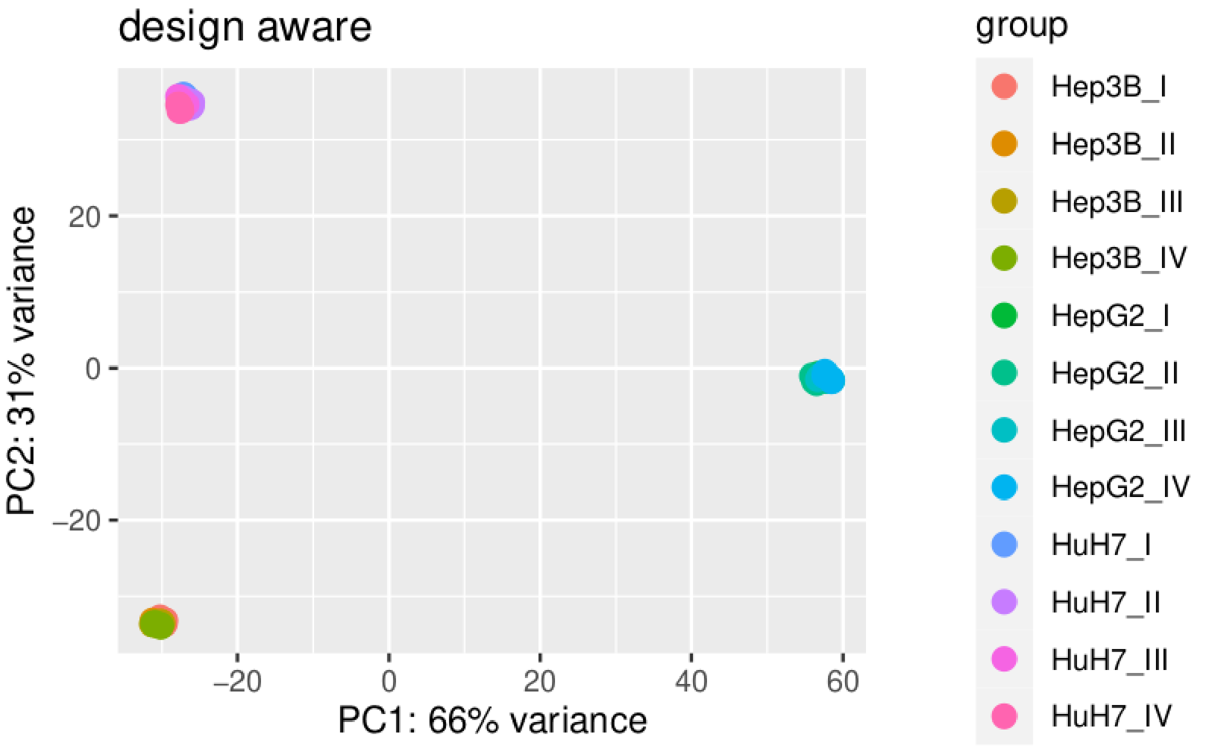
```
  "R": {
    "Version": "4.0.3",
    "Repositories": [
      {
        "Name": "CRAN",
        "URL": "https://cloud.r-project.org"
      }
    ]
  },
  "Bioconductor": {
    "Version": "3.12"
  },
  "Packages": {
    "AnnotationDbi": {
      "Package": "AnnotationDbi",
      "Version": "1.52.0",
      "Source": "Bioconductor",
      "Hash": "ca5106b296b3aa6af713ce197be547c1"
    },
    "BH": {
      "Package": "BH",
      "Version": "1.75.0-0",
      "Source": "Repository",
      "Repository": "CRAN",
      "Hash": "e4c04affc2cac20c8fec18385cd14691"
    },
    "targets": {
      "Package": "targets",
      "Version": "0.1.0.9000",
      "Source": "GitHub",
      "RemoteType": "github",
      "RemoteUsername": "ropensci",
```

Example with RNA-seq data across 3 cell lines

PCA shows that differences between cells >> biological effect (roman numbers)

Solution: Split counts and metadata for each cell

Do we copy code 3 times?



Define targets = explicit dependencies

_targets.R, define 4 targets

Last target depends on the **3** upstreams

```r
library(targets)
source("R/functions.R")
source("R/plotting.R")

list(
  tar_target(cells, c("HepG2", "HuH7", "Hep3B")),
  tar_qs(dds, read_rds(here::here("data", "all.rds")),
         packages = "DESeq2"),
  tar_fst_tbl(annotation, gtf_to_tbl(here::here("data",
                                                "gencode.v36.an
              packages = c("tibble", "rtracklayer")),
  tar_qs(sub_dds, subset_dds(dds,
                             filter(annotation, type == "gene")
                             .cell = cells),
         pattern = map(cells), # dynamic branching
         packages = c("DESeq2", "tidyverse"))
[...]
)
```

Dynamic branching makes dependencies easier to read.

> **Of course, someone has to write for loops, it doesn't have to be you**

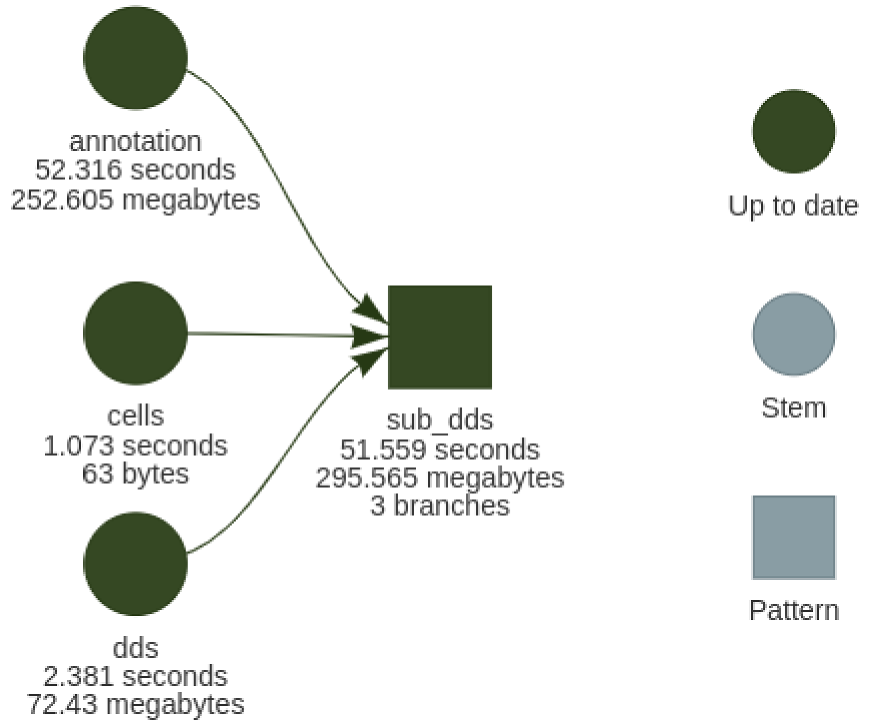
— *Jenny Bryan*

Figure from tar_visnetwork()

Running targets

```
• run target annotation
• run target cells
• run target dds
• run branch sub_dds_3078b1e0
          condition time_h
HepG2_I1    control      0
HepG2_I2       HIL6      2
using pre-existing size factors
estimating dispersions
gene-wise dispersion estimates: 2 workers
mean-dispersion relationship
final dispersion estimates, fitting model and testing: 2 worker
• run branch sub_dds_d05c5da7
         condition time_h
HuH7_I1    control      0
HuH7_I2       HIL6      2
using pre-existing size factors
estimating dispersions
gene-wise dispersion estimates: 2 workers
mean-dispersion relationship
final dispersion estimates, fitting model and testing: 2 worker
• run branch sub_dds_c60d7096
          condition time_h
Hep3B_I1    control      0
Hep3B_I2       HIL6      2
using pre-existing size factors
estimating dispersions
gene-wise dispersion estimates: 2 workers
mean-dispersion relationship
final dispersion estimates, fitting model and testing: 2 worker
• end pipeline
```

Options to display time and object sizes



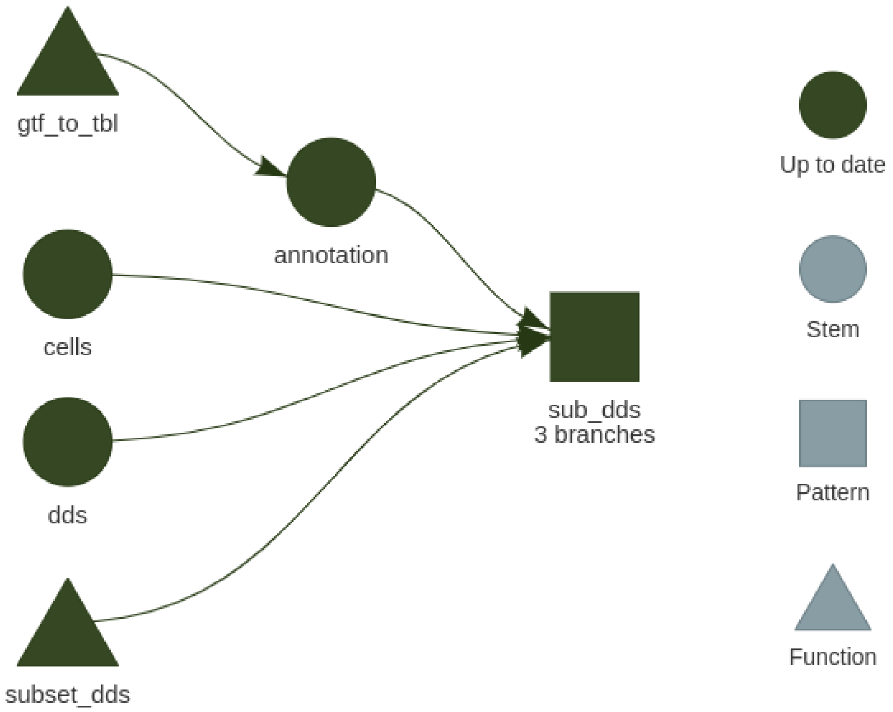
Re-running

```
✓ skip target annotation
✓ skip target cells
✓ skip target dds
✓ skip branch sub_dds_3078b1e0
✓ skip branch sub_dds_d05c5da7
✓ skip branch sub_dds_c60d7096
✓ skip pipeline
```

All good, nothing to be done ✔️.

Actually `targets` tracks all objects and so functions

A more complete dependency graph shows **functions**



Let's add the PCA per cell type now

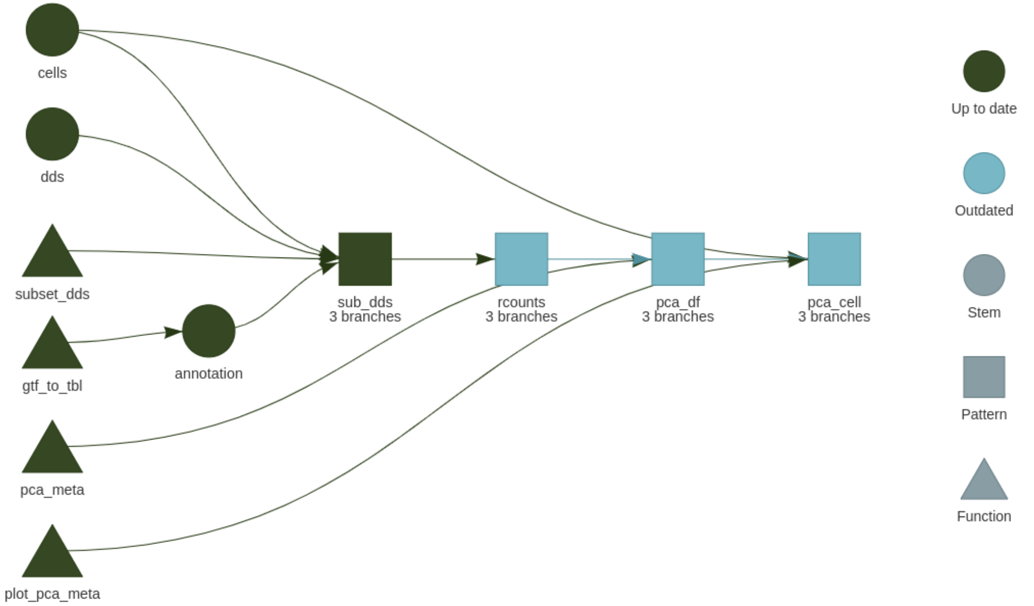
PCA, add 4 targets

Smaller targets avoid unnecessary re-running steps

```
[...]
tar_target(rcounts, vst(sub_dds, blind = TRUE),
           pattern = map(sub_dds),
           packages = c("DESeq2")),
tar_target(pca_df, pca_meta(rcounts),
           pattern = map(rcounts),
           packages = c("DESeq2", "tidyr", "dplyr")),
tar_target(pca_cell, tibble(cell = cells,
                            pca = list(plot_pca_meta(pca_df))),
           pattern = map(cells, pca_df),
           packages = c("ggplot2", "tibble"))
[...]
```

Translate into:

- For every cell data, compute regularized counts (`vst`: variance stabilization)
- For every regularized counts, compute PCA (`df`: data.frame, *i. e* a table)
- For every cell names / PCA tables, plot PCA in a table for easier labeling

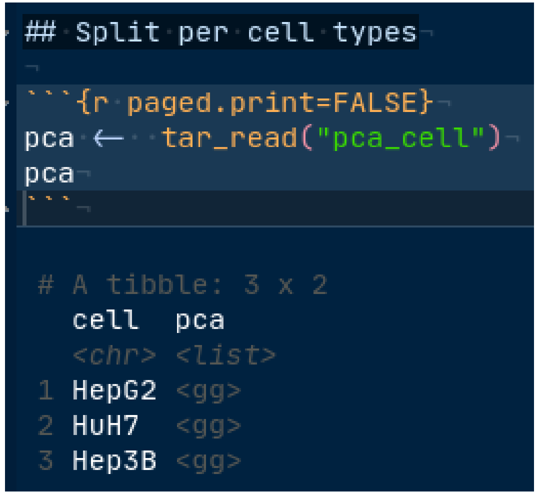


PCA results

Running

Awesome feature: load results IN a Rmarkdown document

Separate code from content

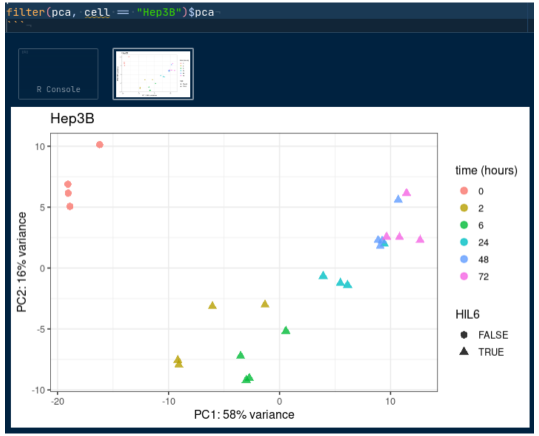


~~~~
## Split per cell types

```{r paged.print=FALSE}
pca <-  tar_read("pca_cell")
pca
```

# A tibble: 3 x 2
  cell  pca
  <chr> <list>
1 HepG2 <gg>
2 HuH7  <gg>
3 Hep3B <gg>
~~~~

How to display a plot



~~~~
filter(pca, cell == "Hep3B")$pca
```
~~~~

The full picture

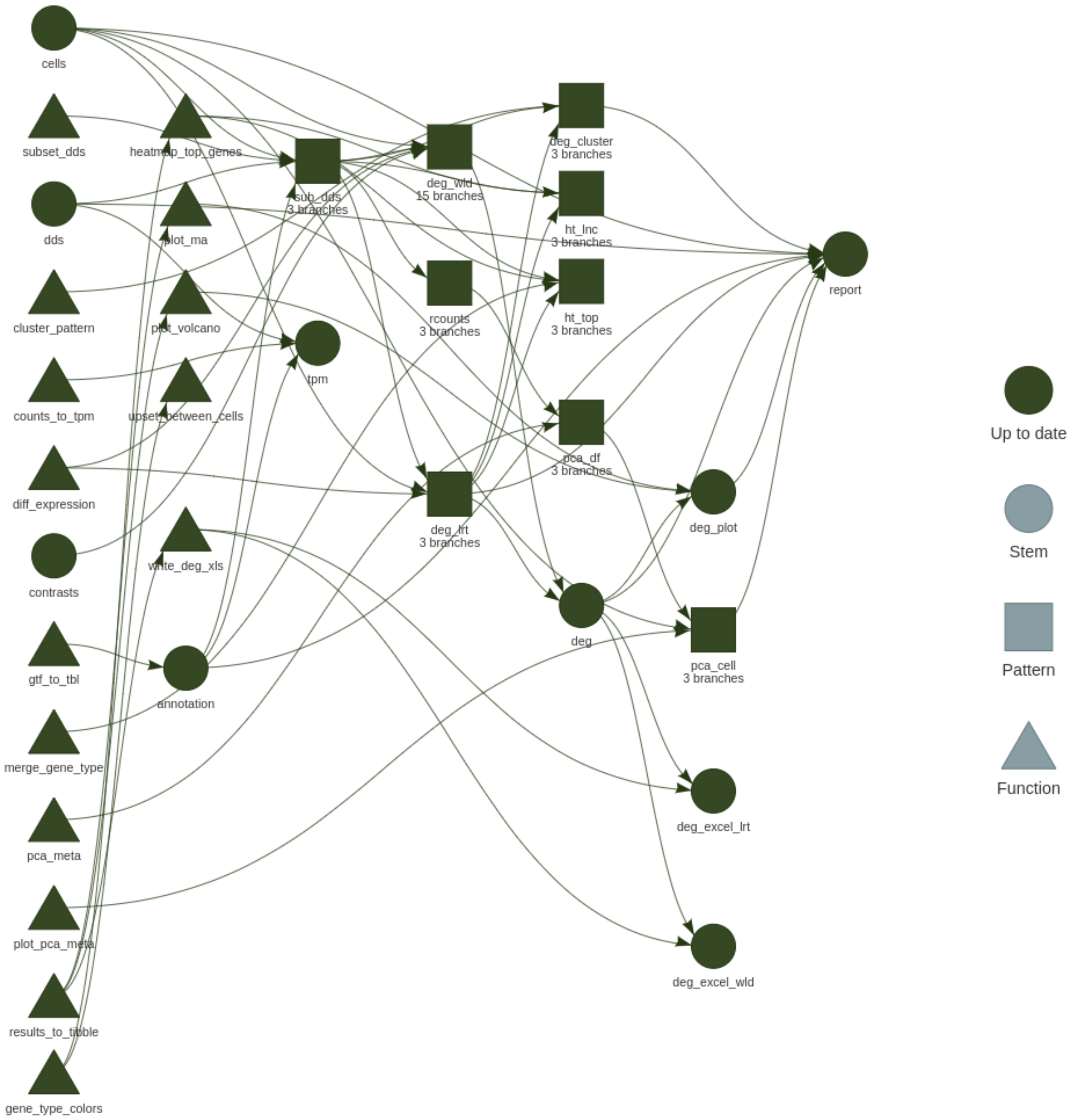
Adding step by step

desired analyses

Whole analysis takes 24 minutes and 4.54 seconds

> Of course, someone has to remember the dependencies, it doesn't have to be you

— *could be William Landau via **Jenny Bryan***



Is it worth the effort?

Yes

For you

- Autonomy
- Skills
- *Free* time
- Confidence over results
- Reproducibility
- Fun 🥳

Better project design

Thinking at what is a good `targets` helps tremendously the coding

> 1. Are large enough to subtract a decent amount of runtime when skipped.
> 2. Are small enough that some targets can be skipped even if others need to run.
> 3. Invoke no side effects (tar_target(format = "file") can save files.) 4.Return a single value that is
>   - Easy to understand and introspect.
>   - Meaningful to the project [...]

William Landau

Reproducibility

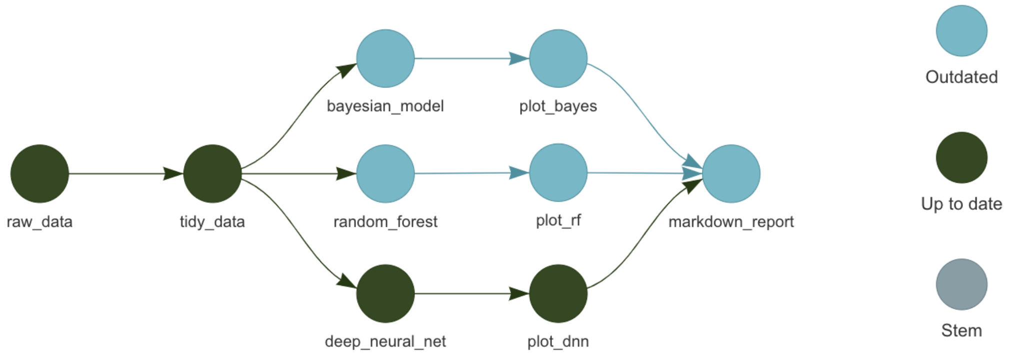
Both thanks to `targets` and `renv` via `git`

```
> renv::history()
   commit         author_date      committer_date                                  subject
1e8dd2278 2021-02-23 15:29:57 2021-02-23 15:29:57             reformat creating config files
24c1222db 2021-02-15 17:07:01 2021-02-15 17:07:01     highlight gene type in the DEG patterns
326c8a726 2021-02-04 16:16:38 2021-02-04 16:16:38 cluster LRT genes by they dynamic patterns
4c6791796 2021-01-26 13:08:15 2021-01-26 13:08:15      gene types in upset plots for lengths
5865ee70b 2021-01-21 16:36:48 2021-01-21 16:37:08                           add upset plots
[...]
```

Scalability and parallelization

- Scale-up with **dynamic** branching
- Parallelization on **HPC** using:
  - `tar_make_clustermq(workers = 3L)` (`clustermq` by **Michael Schubert**)
  - `tar_make_future(workers = 3L)` (`future` by **Henrik Bengtsson**)
- **Static** branching

to get explicit branch names.



Source: **William Landau**: talk at Bayes Lund

Reports as Rmarkdown documents

`targets`, written by William Landau (pictured), is flexible, robust and still allows for a customized report.

All computing is done only when needed, and code is away from writing content.

Pipelines can now also be a **Rmd**!

Once `knitted` the report can be sent to the inquirer.

Targets Markdown

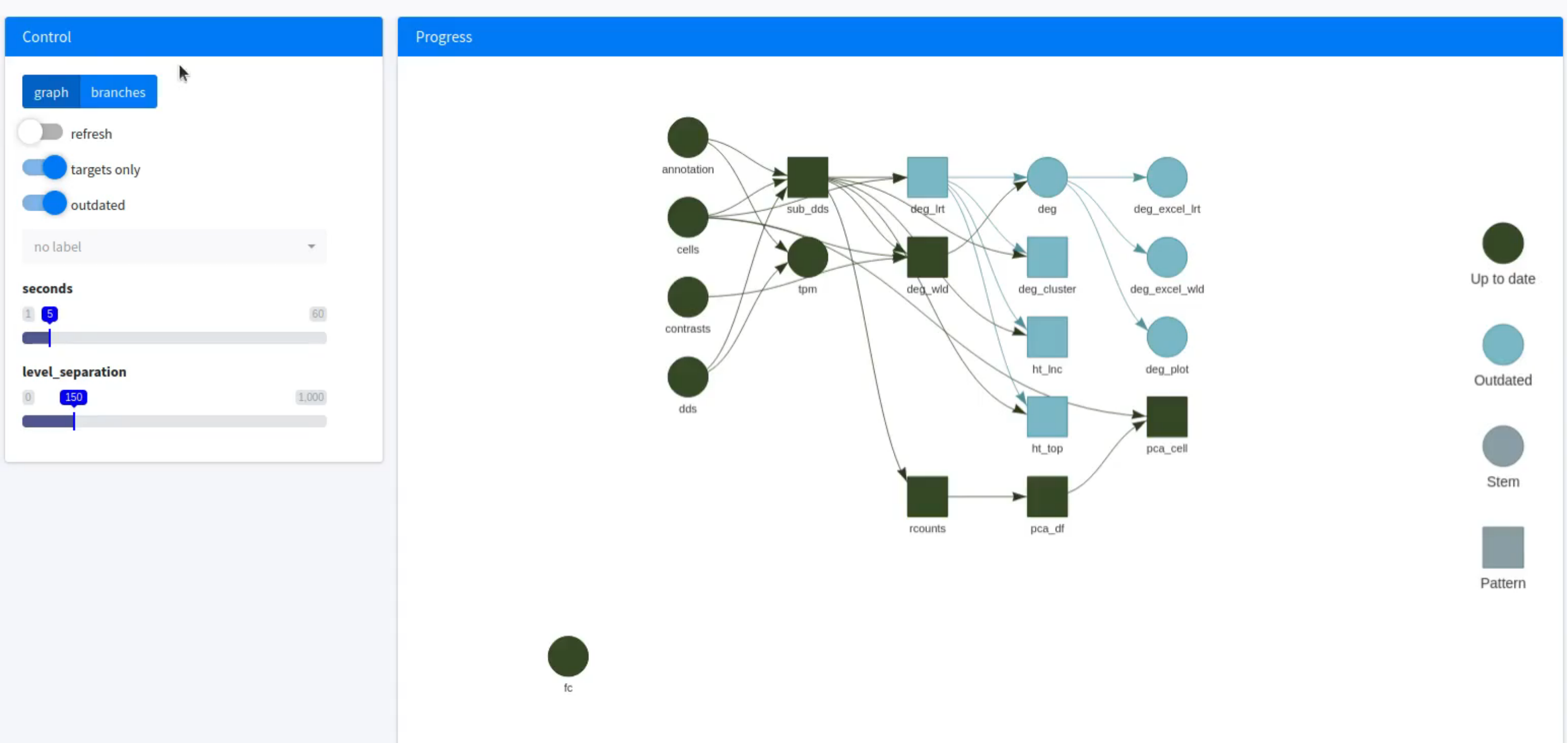
New in `targets` > **0.6**. Instructions at William bookdown

Test it as the Rmd template (and excellent video from R Lille meetup by **Landau**):

Bonus: watch the pipeline running live 🍿

- `targets` events watched live 🎞️
- Here, after changing a threshold in the LRT step
- `branches` can be monitored too
- 2 videos joined as I fixed an **error** at 1'42"
- Option to display functions (unset here)

`tar_watch()` shiny app from `targets`



Control

graph | branches

○ refresh

● targets only

● outdated

no label

seconds: 5 (1–60)

level_separation: 150 (0–1,000)

Progress

Before we stop

Highlights

- `targets`: a Makefile-like approach for project design
    - dependencies manager
    - re-run only what's needed

Acknowledgments 🙏 👏

- **Eric Koncina** early adopter of `targets`
- **Wendkouni N. Minoungou** for the RNA-seq data
- **William Landau** main developer of `targets`
- Xie Yihui and Garrick Aden-Buie for `xarigan`/`xaringanExtra`
- Jennifer Bryan

Further reading 📒

- Main website
- Targetopia **Landau** universe of targets-derived
- Video from R Lille meetup by **William Landau**. June 2021 45"
- Video from Bayes Lund by **William Landau**. October 2021
- Documentation as bookdown by **Landau**

Thank you for your attention!