34 Reproducible Pipelines

R. Heller · Date: 2026-05-09 · Last changed: 2026-10-31

34.1 Learning objectives

After completing this chapter, you will be able to:

- Explain why ad hoc scripts are insufficient for reproducible bibliometric analysis
- Define a targets pipeline with dependency tracking and selective re-execution
- Use renv to snapshot and restore R package environments
- Understand how Docker containers ensure cross-platform reproducibility
- Structure a bibliometric project for long-term reproducibility

34.2 Setup

```
library(tidyverse)
library(glue)

set.seed(20260509)

source(here::here("R", "sci_palette.R"))
```

34.3 Conceptual background

A bibliometric analysis is a multi-step process: fetch data from an API, clean and deduplicate records, compute indicators, build networks, fit models, and generate figures. Each step depends on earlier ones. When the upstream data changes (a new API query, an updated corpus), downstream results must be regenerated — but only the affected parts.

Ad hoc scripts (running scripts manually in sequence) fail at scale: they do not track dependencies, they re-run everything even when only one input changed, and they break when run on a different machine with different package versions.

`targets` is an R pipeline tool that solves the dependency problem. You define each analysis step as a "target" (a function that takes inputs and produces outputs). `targets` builds a dependency graph, executes only the targets whose inputs have changed, and caches results. This is both faster (no unnecessary re-computation) and safer (no stale intermediate results).

`renv` snapshots the exact package versions used in your analysis into a `renv.lock` file. A collaborator (or your future self) can restore the identical package environment with `renv::restore()`. This eliminates the "works on my machine" problem for R packages.

Docker takes reproducibility further by containerising the entire compute environment: operating system, R version, system libraries, and package versions. A `Dockerfile` defines the environment declaratively; anyone with Docker can rebuild it identically. Docker is essential for long-term reproducibility, since R versions and system libraries evolve.

These tools complement each other: `targets` manages the computation, `renv` manages the packages, and Docker manages the environment. Together, they make a bibliometric analysis fully reproducible years after it was originally run.

34.4 Worked example

34.4.1 Defining a targets pipeline

The `_targets.R` file defines the pipeline. Here we show the structure for a typical bibliometric analysis.

```
# _targets.R — pipeline definition
library(targets)

tar_option_set(packages = c("tidyverse", "openalexR", "igraph"))

list(
  tar_target(raw_works, {
    fetch_openalex(
      entity = "works",
      primary_location.source.id = "S148561398",
      from_publication_date = "2020-01-01",
      to_publication_date = "2023-12-31",
      options = list(sample = 300, seed = 42)
    )
  }),

  tar_target(clean_works, {
    raw_works |>
      filter(!is.na(doi)) |>
      distinct(doi, .keep_all = TRUE) |>
      mutate(year = year(publication_date))
  }),

  tar_target(citation_stats, {
    clean_works |>
      group_by(year) |>
      summarise(
        n = n(),
        mean_cites = mean(cited_by_count),
        .groups = "drop"
      )
  }),

  tar_target(fig_trends, {
    ggplot(citation_stats, aes(x = year, y = mean_cites)) +
      geom_line() + geom_point() +
      labs(x = "Year", y = "Mean citations")
  })
)
```

34.4.2 Running and inspecting the pipeline

```
# Run the pipeline
targets::tar_make()

# Visualise the dependency graph
targets::tar_visnetwork()

# Read a specific target
targets::tar_read(citation_stats)
```

34.4.3 Project structure

```
# Recommended project layout:
#
# my-bibliometric-project/
# ├── _targets.R          # Pipeline definition
# ├── R/                  # Helper functions
# │   ├── fetch.R
# │   ├── clean.R
# │   └── analyse.R
# ├── renv.lock           # Package snapshot
# ├── Dockerfile          # Environment definition
# ├── data/               # Cached data (Parquet)
# ├── output/             # Figures and tables
# └── report.qmd          # Quarto report consuming targets
```

34.4.4 Using renv

```
# Initialise renv in a new project
renv::init()

# Snapshot current package state
renv::snapshot()

# Restore packages on a new machine
renv::restore()
```

34.4.5 Dockerfile for bibliometric analysis

```
# Example Dockerfile (not R code — shown for reference)
#
# FROM rocker/verse:4.4.1
# RUN install2.r targets renv openalexR igraph
# COPY renv.lock /project/renv.lock
# WORKDIR /project
# RUN R -e "renv::restore()"
# CMD ["R", "-e", "targets::tar_make()"]
```

34.5 Diagnostics and interpretation

- **Target status**: `tar_outdated()` lists which targets need re-running. If everything is up to date, `tar_make()` completes instantly.

- **Pipeline visualisation**: `tar_visnetwork()` shows the dependency graph. Disconnected subgraphs indicate independent analysis streams.
- **Cache management**: Targets stores cached results in `_targets/`. This directory can grow large. Use `tar_prune()` to remove unused targets.
- **renv consistency**: Run `renv::status()` to check if your lockfile matches the installed packages.

34.6 Limitations and responsible use

34.7 Limitations and responsible use

- **API rate limits**: targets will re-run API calls when inputs change. Use file-based caching (Parquet) to avoid hitting API rate limits on every pipeline run.
- **Docker learning curve**: Docker adds complexity. For solo projects, renv alone may suffice. Docker becomes essential for team reproducibility and long-term archiving.
- **Overhead for small projects**: A 50-line analysis script does not need targets. Use pipelines when the analysis has 5+ interdependent steps or will be re-run multiple times.
- **Reproducibility is not eternal**: Even with Docker, operating system patches and hardware changes can affect results. Archive final outputs alongside the pipeline.

34.8 Common pitfalls

34.9 Common pitfalls

- **Not separating data retrieval from analysis.** API calls in analysis targets re-fetch data on every change. Separate fetching into its own target with file-based caching.
- **Forgetting to commit renv.lock.** The lockfile is the reproducibility guarantee. Always version-control it.
- **Ignoring the _targets/ directory in .gitignore.** Cached results should not be committed to git (they can be regenerated). Add `_targets/` to `.gitignore`.
- **Writing monolithic targets.** Each target should do one thing. Fine-grained targets enable selective re-execution.

34.10 Exercises

1. **Build a pipeline.** Create a `_targets.R` file for a simple bibliometric analysis: fetch → clean → summarise → plot. Run it and verify that changing the summary function only re-runs downstream targets.
2. **renv snapshot.** Initialise renv in a project, install a new package, and snapshot. Delete the package and restore from the lockfile.
3. **Branching.** Use `tar_target()` with dynamic branching to run the same analysis across multiple journals in parallel.

34.11 Solutions

Solutions are provided in E.

34.12 Further reading

- Hicks et al. (2015) — The Leiden Manifesto: transparency and reproducibility in research evaluation.
- Priem et al. (2022) — OpenAlex as a reproducible data source for bibliometric pipelines.

34.13 Session info

```
sessionInfo()
```

```
#> R version 4.6.1 (2026-06-24)
#> Platform: x86_64-pc-linux-gnu
#> Running under: Ubuntu 24.04.5 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
#> LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so;  LAPACK
version 3.12.0
#>
#> locale:
#>  [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C           LC_TIME=C.UTF-8
#>  [4] LC_COLLATE=C.UTF-8     LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
#>  [7] LC_PAPER=C.UTF-8       LC_NAME=C              LC_ADDRESS=C
#> [10] LC_TELEPHONE=C         LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C
#>
#> time zone: UTC
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] uwot_0.2.5                Matrix_1.7-5
#>  [3] word2vec_0.4.1            stm_1.3.10
#>  [5] topicmodels_0.2-17        quanteda.textstats_0.97.2
#>  [7] visNetwork_2.1.4          ggraph_2.2.2
#>  [9] tidygraph_1.3.1           igraph_2.3.4
#> [11] quanteda_4.5.1            pdftools_3.9.1
#> [13] arrow_26.0.1.1            bibliometrix_5.5.0
#> [15] RefManageR_1.4.0          bib2df_1.1.2.0
#> [17] rcrossref_1.2.1           gt_1.4.0
#> [19] tidytext_0.4.3            glue_1.8.1
#> [21] openalexR_3.1.2           lubridate_1.9.5
#> [23] forcats_1.0.1             stringr_1.6.0
#> [25] dplyr_1.2.1               purrr_1.2.2
#> [27] readr_2.2.0               tidyr_1.3.2
#> [29] tibble_3.3.1              ggplot2_4.0.4
#> [31] tidyverse_2.0.0
#>
#> loaded via a namespace (and not attached):
#>   [1] bibtex_0.5.3           RColorBrewer_1.1-3     rstudioapi_0.19.0
#>   [4] jsonlite_2.0.0         magrittr_2.0.5         modeltools_0.2-25
#>   [7] farver_2.1.2           rmarkdown_2.32         fs_2.1.0
#>  [10] vctrs_0.7.3            memoise_2.0.1          askpass_1.2.1
#>  [13] base64enc_0.1-6        htmltools_0.5.9        contentanalysis_1.1.1
#>  [16] curl_8.1.0             broom_1.0.13           janeaustenr_1.0.0
#>  [19] cellranger_1.1.0       sass_0.5.0             bslib_0.12.0
#>  [22] htmlwidgets_1.6.4      tokenizers_0.3.0       plyr_1.8.9
#>  [25] httr2_1.3.0            plotly_4.12.1          cachem_1.1.0
#>  [28] dimensionsR_0.0.3      mime_0.13              lifecycle_1.0.5
#>  [31] pkgconfig_2.0.3        R6_2.6.1               fastmap_1.2.0
#>  [34] shiny_1.15.0           digest_0.6.39          patchwork_1.3.2
#>  [37] shinycssloaders_1.1.0  rprojroot_2.1.1        RSpectra_0.16-2
#>  [40] SnowballC_0.7.1        labeling_0.4.3         urltools_1.7.3.1
#>  [43] timechange_0.4.0       mgcv_1.9-4             polyclip_1.10-7
#>  [46] httr_1.4.9             compiler_4.6.1         here_1.0.2
#>  [49] bit64_4.8.6            withr_3.0.3            backports_1.5.1
#>  [52] S7_1.0.0               viridis_0.6.5          ggforce_0.5.0
#>  [55] MASS_7.3-65            bibliometrixData_0.3.0 tools_4.6.1
#>  [58] otel_0.3.0             stopwords_2.3          zip_3.0.2
#>  [61] httpuv_1.6.17          rentrez_1.2.4          nlme_3.1-169
#>  [64] promises_1.5.0         grid_4.6.1             stringdist_0.9.17
#>  [67] reshape2_1.4.5         generics_0.1.4         gtable_0.3.6
```

```
#>  [70] tzdb_0.5.0             rscopus_0.9.0          ca_0.72
#>  [73] data.table_1.18.6.1    hms_1.1.4              xml2_1.6.0
#>  [76] utf8_1.2.6             ggrepel_0.9.8          pillar_1.11.1
#>  [79] nsyllable_1.0.1        vroom_1.7.1            later_1.4.8
#>  [82] splines_4.6.1          tweenr_2.0.3           lattice_0.22-9
#>  [85] FNN_1.1.4.1            bit_4.6.0              tidyselect_1.2.1
#>  [88] tm_0.7-20              miniUI_0.1.2           downlit_0.4.5
#>  [91] knitr_1.52             gridExtra_2.3.1        NLP_0.3-3
#>  [94] bookdown_0.48          stats4_4.6.1           crul_1.6.0
#>  [97] xfun_0.62              graphlayouts_1.2.5     matrixStats_1.5.0
#> [100] DT_0.34.0              humaniformat_0.6.0     stringi_1.8.9
#> [103] qpdf_1.4.1             yaml_2.3.12            evaluate_1.0.5
#> [106] httpcode_0.3.0         cli_3.6.6              xtable_1.8-8
#> [109] jquerylib_0.1.4        Rcpp_1.1.2             readxl_1.6.0
#> [112] triebeard_0.4.1        XML_3.99-0.25          parallel_4.6.1
#> [115] assertthat_0.2.1       pubmedR_1.1.0          slam_0.1-56
#> [118] viridisLite_0.4.3      scales_1.4.0           crayon_1.5.3
#> [121] openxlsx_4.2.9         rlang_1.3.0            fastmatch_1.1-8
```