# targets



install.packages("targets")

The targets2 package is a new pipeline toolkit for R. It recently cleared software review, and it is now on CRAN. targets is the long-term successor of drake3, which in turn succeeded Rich FitzJohn’s groundbreaking remake4 package. A chapter in the user manual explains the future of drake, the advantages of targets, and how to transition. The reference website explains how to get started, and the overview vignette describes the major features of targets and its user manual.

# How it works

In targets, a data analysis pipeline is a collection of target objects that express the individual steps of the workflow, from upstream data processing to downstream R Markdown reports5. These targets live in a special script called \_targets.R.

# \_targets.R file library(targets)

tar\_option\_set(packages = c("biglm", "dplyr", "ggplot2", "readr")) # Most workflows have custom functions to support the targets. read\_clean <- function(path) {

path %>%

read\_csv(col\_types = cols()) %>%

mutate(Ozone = replace\_na(Ozone, mean(Ozone, na.rm = TRUE))

}

fit\_model <- function(data) { biglm(Ozone ~ Wind + Temp, data)

}

create\_plot <- function(data) { ggplot(data) +

geom\_histogram(aes(x = Ozone), bins = 12) + theme\_gray(24)

}

# List of targets. list(

# airquality dataset in base R:

tar\_target(raw\_data\_file, "raw\_data.csv", format = "file"), tar\_target(data, read\_clean(raw\_data\_file)), tar\_target(fit, fit\_model(data)),

tar\_target(hist, create\_plot(data))

)

targets inspects your code and constructs a dependency graph.

# R console tar\_visnetwork()

tar\_make() runs the correct targets in the correct order.

# R console tar\_make()

#> ● run target raw\_data\_file #> ● run target data

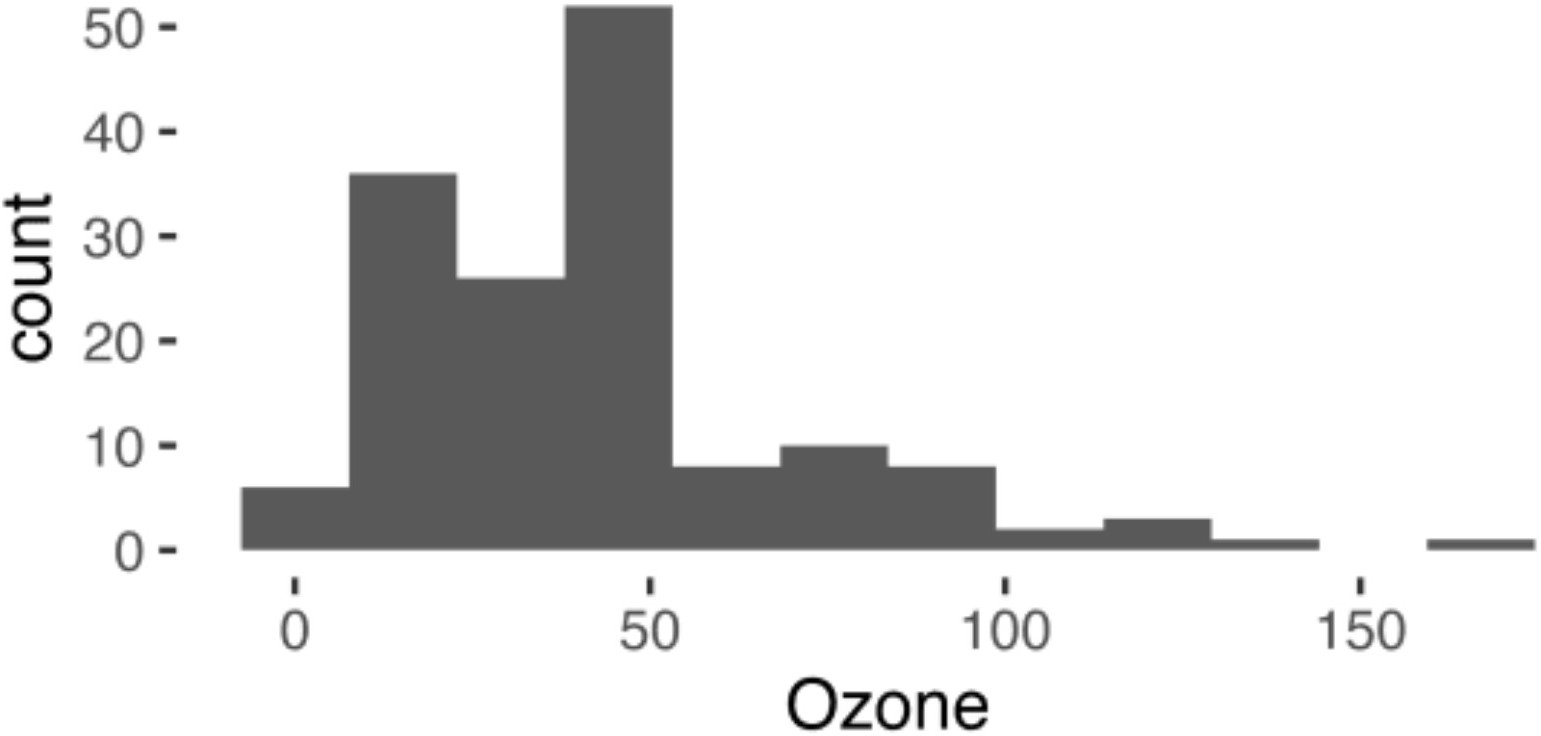
#> ● run target fit #> ● run target hist #> ● end pipeline

Alternatives tar\_make\_clustermq() and tar\_make\_future() leverage clustermq6 and future7, respectively, to distribute targets on traditional schedulers such as SLURM8. It is only a matter of time before these backends become capable of sending jobs to the cloud9.

Your can store the results in the \_targets/ folder (default) or Amazon S3 buckets. Either way, loading data back into R is the same.

# R console

tar\_read(hist) # see also tar\_load()



Up-to-date targets do not rerun, which saves countless hours in computationally intense fields like machine learning, Bayesian statistics, and statistical genomics.

# R console tar\_make()

#> ✓ skip target raw\_data\_file #> ✓ skip target data

#> ✓ skip target fit #> ✓ skip target hist #> ✓ skip pipeline

# The next challenge

To help workflows scale, targets adopts the classical, pedantic, function-oriented perspective of the R language.10

Nearly everything that happens in R results from a function call. Therefore, basic programming centers on creating and refining functions.

The more often you write your own functions, the nicer your experience becomes.

I’m thinking about why this exists only in R and it may be because:

1. R’s functional approach makes it easier to detect dependencies, and
2. R’s uses lazy evaluation

But if your mind is on the domain knowledge, or if you feel pressure to work fast, then it can be hard to write functions for everything.

# Target factories

The best way to write fewer functions is to write less code. To write less code, we need abstraction and automation. Target factories are package functions that return lists of pre-configured target objects, and they make specialized pipelines reusable.

# script inside example.package #' @export

read\_clean <- function(path) { path %>%

read\_csv(col\_types = cols()) %>%

mutate(Ozone = replace\_na(Ozone, mean(Ozone, na.rm = TRUE))

}

#' @export

fit\_model <- function(data) { biglm(Ozone ~ Wind + Temp, data)

}

#' @export

create\_plot <- function(data) { ggplot(data) +

geom\_histogram(aes(x = Ozone), bins = 12) + theme\_gray(24)

}

#' @title Example target factory.

#' @description Concise shorthand to express our example pipeline. #' @details

#' Target factories should use `tar\_target\_raw()`.

#' `tar\_target()` is for users, and `tar\_target\_raw()` is for developers. #' The former quotes its arguments, while the latter evaluates them.

#' @export

biglm\_factory <- function(file) { list(

tar\_target\_raw("raw\_data\_file", as.expression(file), format = "file"), tar\_target\_raw("data", quote(example.package::read\_clean(raw\_data\_file))), tar\_target\_raw("fit", quote(example.package::fit\_model(data))), tar\_target\_raw("hist", quote(example.package::create\_plot(data)))

)

}

With the factory above, our long \_targets.R file suddenly collapses down to three lines.

# \_targets.R file library(targets) library(example.package)

biglm\_factory("raw\_data.csv")

And you still have complete freedom to add more targets to the list.

# \_targets.R file library(targets) library(example.package)

run\_model2 <- function(data) {...}

list( # Target lists can be arbitrarily nested. biglm\_factory("raw\_data.csv"), tar\_target(model2, run\_model2(data))

)

# The R Targetopia



The R Targetopia11 is an emerging ecosystem of packages to bring target factories to specific domains of Statistics and data science.

## stantargets

library(remotes) install\_github("wlandau/stantargets") library(cmdstanr)

install\_cmdstan()

stantargets12 abstracts away most of the targets and functions required for a solid Bayesian data analysis with Stan13. With a single target factory and a single function to generate data, stantargets can give you an entire sensitivity analysis or an entire simulation-based calibration study.14 15

# \_targets.R for simulation-based calibration to validate a Stan model. library(targets)

library(stantargets) generate\_data <- function() {

true\_beta <- stats::rnorm(n = 1, mean = 0, sd = 1) x <- seq(from = -1, to = 1, length.out = n)

y <- stats::rnorm(n, x \* true\_beta, 1)

list(n = n, x = x, y = y, true\_beta = true\_beta)

}

list( tar\_stan\_mcmc\_rep\_summary( model

"model.stan", # We assume you already have a Stan model file. generate\_data(), # Runs once per rep.

batches = 25, # Batching reduces per-target overhead. reps = 40, # Number of simulation reps per batch. data\_copy = "true\_beta",

variables = "beta", summaries = list(

~posterior::quantile2(.x, probs = c(0.025, 0.975))

)

)

)

# R console tar\_visnetwork()

## tarchetypes



install.packages("tarchetypes")

The tarchetypes16 R Targetopia package is far more general than stantargets. Its target factories include tar\_rep() for arbitrary simulation studies, tar\_render() for dependency-aware literate programming, and tar\_render\_rep() for parameterized R Markdown. tar\_plan() is a drake\_plan()-like target factory to help drake users transition to targets.

# \_targets.R file library(targets) library(tarchetypes) tar\_plan(

tar\_target(raw\_data\_file, "raw\_data.csv", format = "file"), data = read\_clean(raw\_data\_file),

fit = fit\_model(data), hist = create\_plot(data)

)

## You can help!

The R Targetopia has exciting potential for tidymodels