targets

```
install.packages("targets")
```

The `targets`[2] 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 `drake`[3], which in turn succeeded Rich FitzJohn's groundbreaking `remake`[4] 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 reports[5]. 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.

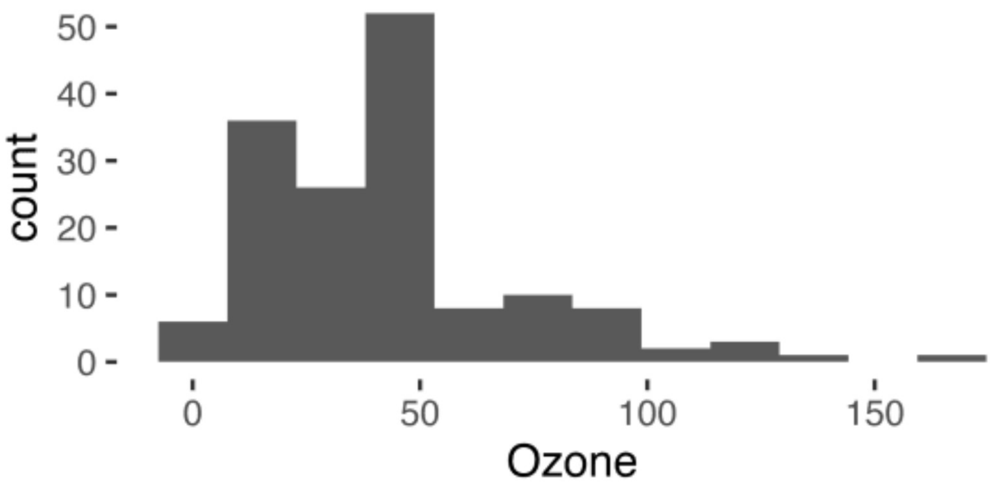
```
# R console
tar_make()

#> ● 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 `clustermq`[6] and `future`[7], respectively, to distribute targets on traditional schedulers such as SLURM[8]. It is only a matter of time before these backends become capable of sending jobs to the cloud[9].

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.
>
> — John Chambers

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
>
> I tried building a little prototype equivalent in Julia and I think it's possible, but above my skill level
>
> — Dr. David Neuzerling (@mdneuzerling) December 17, 2020

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 Targetopia[11] 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()
```

`stantargets`[12] abstracts away most of the targets and functions required for a solid Bayesian data analysis with Stan[13]. 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 `tarchetypes`[16] 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