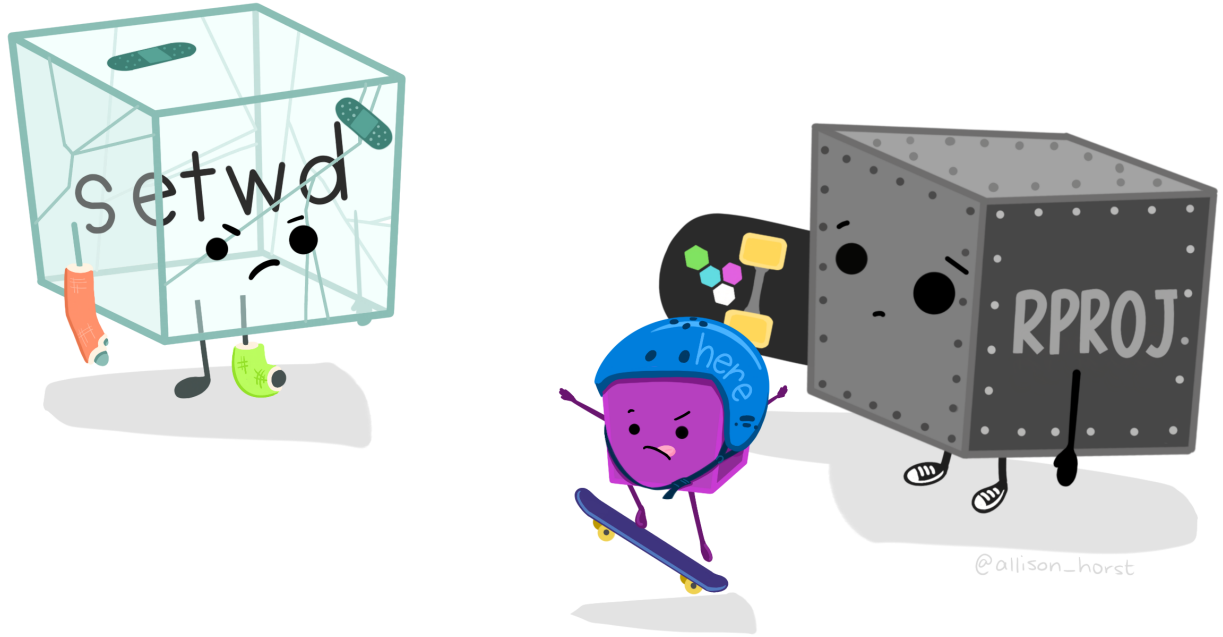
Code pipelines in R

Managing code with targets

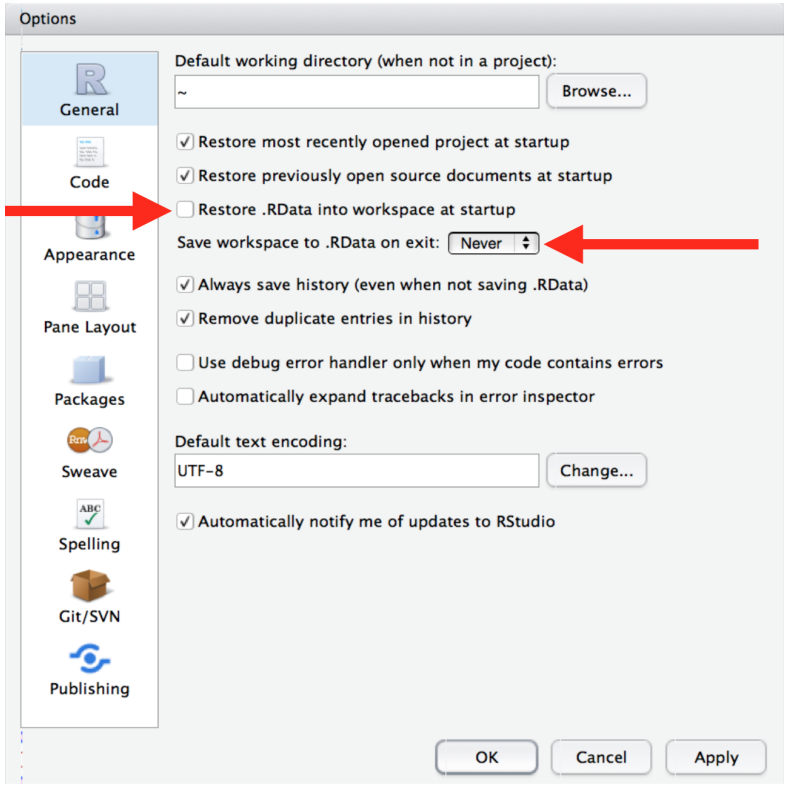
2021-06-04

Strategies for reproduciblity



Project-Oriented Workflows

Strategies for reproduciblity



Use a clean slate; Restart early & often

Strategies for reproduciblity

R Markdown for reproducible documents

renv for package version management

git/GitHub for version control

Script-oriented workflows

```
01-read-data.R
02-clean-data.R
03-decriptive-stats.R
...
n-output.R
report.Rmd
```

Doesn't scale well—in terms of both time and scope

Script-oriented workflows

```
01-read-data.R
02-clean-data.R
03-decriptive-stats.R
...
n-output.R
report.Rmd
```

Doesn't scale well—in terms of both time and scope

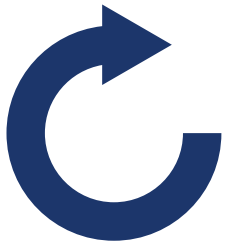
Not clear what we can skip

Function-oriented workflows

```
source("functions.R")
data <- read_data("data.csv") %>%
  clean_data()
table1 <- create_table(data)
...
ggsave("figure3.png")
rmarkdown::render("report.Rmd")
```

Scale the work you need.

Skip the work you don't.



See evidence of reproducibility.

Setting up a pipeline

targets requires a _targets.R file in the root directory of your project

Setting up a pipeline

targets requires a _targets.R **file in the root directory of your project**

Create it with tar_script()**; open it with** tar_edit()

_targets.R

```
library(targets)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = "tidyverse")
source("R/functions.R")

list(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())),
  tar_target(plot, create_line_plot(gapminder))
)
```

_targets.R

```
library(targets)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = "tidyverse")
source("R/functions.R")

list(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())),
  tar_target(plot, create_line_plot(gapminder))
)
```

_targets.R

```
library(targets)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = "tidyverse")
source("R/functions.R")

list(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())),
  tar_target(plot, create_line_plot(gapminder))
)
```

_targets.R

```
library(targets)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = "tidyverse")
source("R/functions.R")

list(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())
  tar_target(plot, create_line_plot(gapminder))
)
```

functions.R **is common for small projects, but targets doesn't care how or from where you source functions.**

R/functions.R

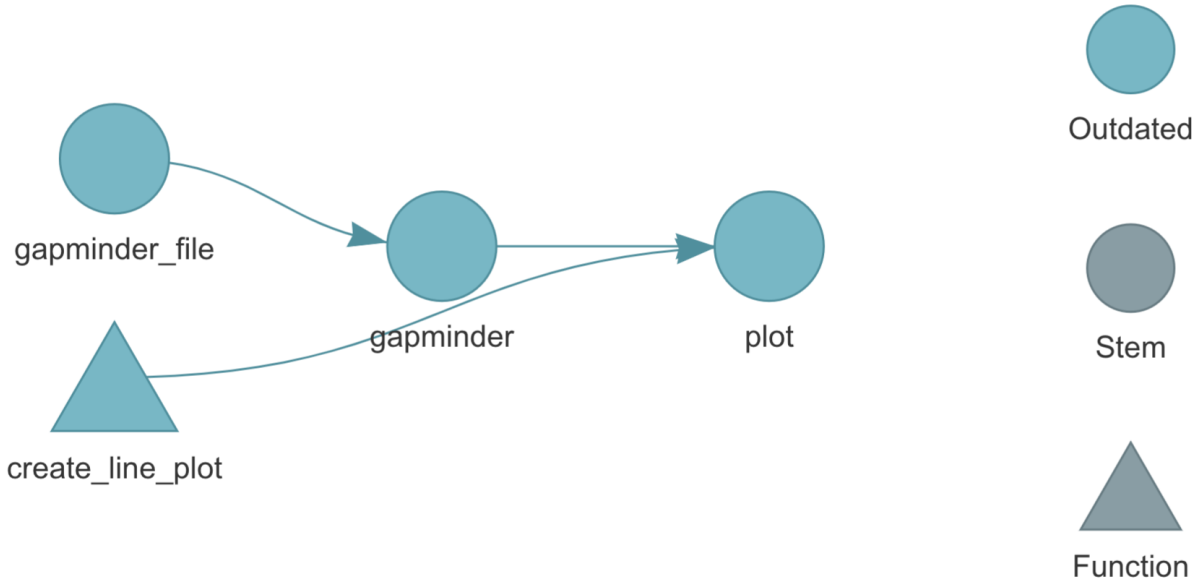
```r
create_line_plot <- function(gapminder) {
  gapminder %>%
    filter(continent != "Oceania") %>%
    ggplot(aes(x = year, y = lifeExp, group = country, color = count
    geom_line(lwd = 1, show.legend = FALSE) +
    facet_wrap(~ continent) +
    theme_minimal(14) +
    theme(strip.text = element_text(size = rel(1.1)))
}
```

_targets.R

```
library(targets)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = "tidyverse")
source("R/functions.R")

list(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())),
  tar_target(plot, create_line_plot(gapminder))
)
```

_targets.R **must** end in a list of targets

tar_visnetwork()



Run this in the console! Don't put it in _targets.R or other scripts.

_targets.R

```
library(targets)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = "tidyverse")
source("R/functions.R")

list(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())),
  tar_target(plot, create_line_plot(gapminder))
)
```

_targets.R

```
library(targets)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = "tidyverse")
source("R/functions.R")

list(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())),
  tar_target(plot, create_line_plot(gapminder))
)
```

_targets.R

```
library(targets)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = "tidyverse")
source("R/functions.R")

list(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())),
  tar_target(plot, create_line_plot(gapminder))
)
```

Building the pipeline

```
tar_make()
```

- start target gapminder_file
- built target gapminder_file
- start target gapminder
- built target gapminder
- start target plot
- built target plot
- end pipeline

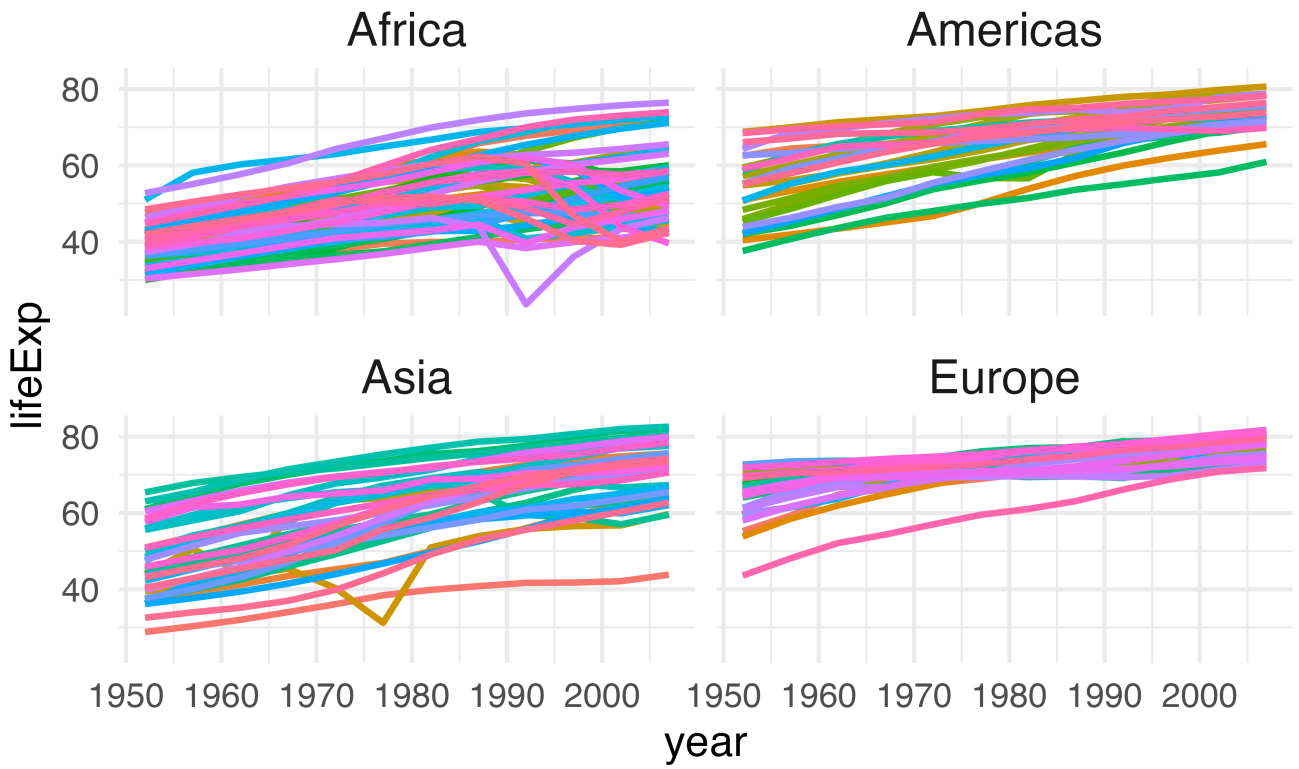
Run this in the console! Don't put it in _targets.R or other scripts.

Where's my output?

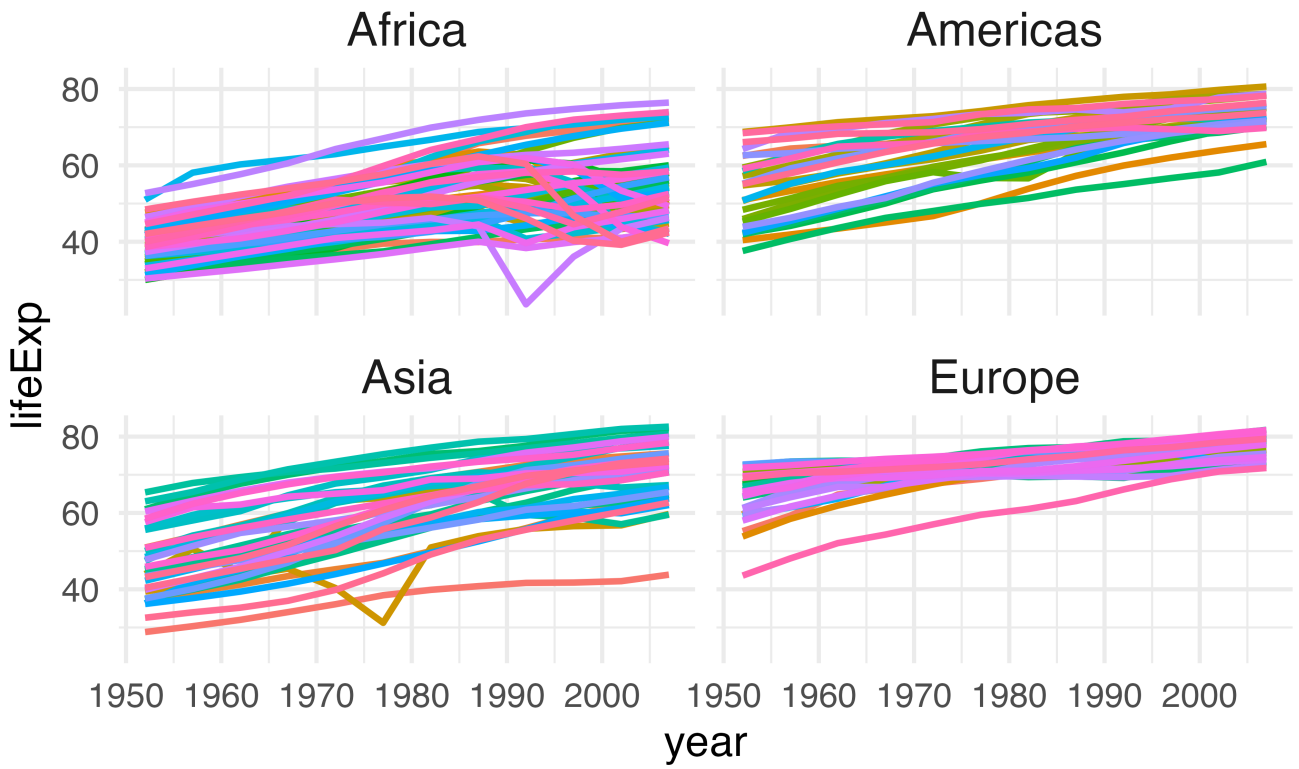
`tar_read(target_name)`: return target results

`tar_load(target_name)`: load target_name into global environment

```
tar_read(plot)
```

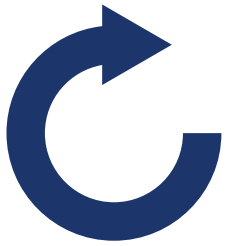


```
tar_load(plot)

plot
```



Scale the work you need.

Skip the work you don't.



See evidence of reproducibility.

Your Turn 1

Use tar_script() **in the console, then open** _targets.R **(you can also use** tar_edit() **in the console to open it for you). Read the resulting script.**

Predict how many targets there are, what they are called, and any other dependencies in your code. Run tar_visnetwork() **in the console to check if you were right.**

Run tar_make() **in the console, then run** tar_visnetwork() **again. What's different? Try running** tar_make() **again.**

Building our mental model

```
tar_target(plot, create_line_plot(gapminder))
```

```
plot <- create_line_plot(gapminder)
```

```
list(
  plot = create_line_plot(gapminder)
)
```

tarchetypes

Collection of target and pipeline archetypes for targets

Express complicated pipelines with concise syntax

Need to include library(tarchetypes) **in** _targets.R

Plans

```
list(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())
  tar_target(plot, create_line_plot(gapminder))
)
```

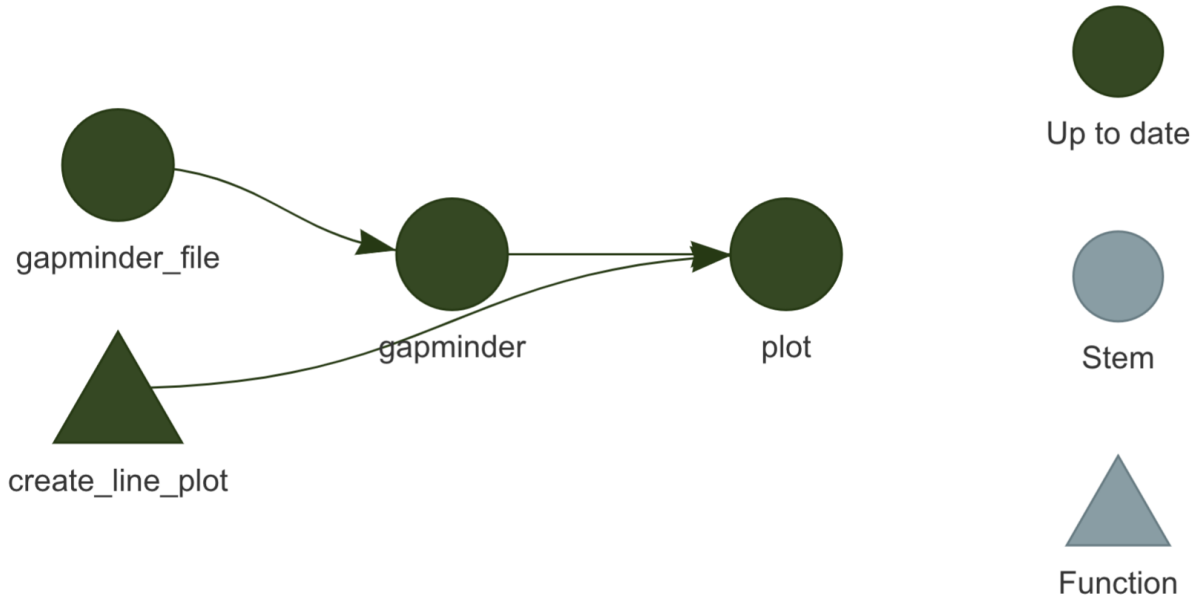
Plans

```
tar_plan(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  tar_target(gapminder, read_csv(gapminder_file, col_types = cols())),
  tar_target(plot, create_line_plot(gapminder))
)
```

Plans

```
tar_plan(
  tar_target(gapminder_file, "gapminder.csv", format = "file"),
  gapminder = read_csv(gapminder_file, col_types = cols()),
  plot = create_line_plot(gapminder)
)
```

tar_visnetwork()



Files

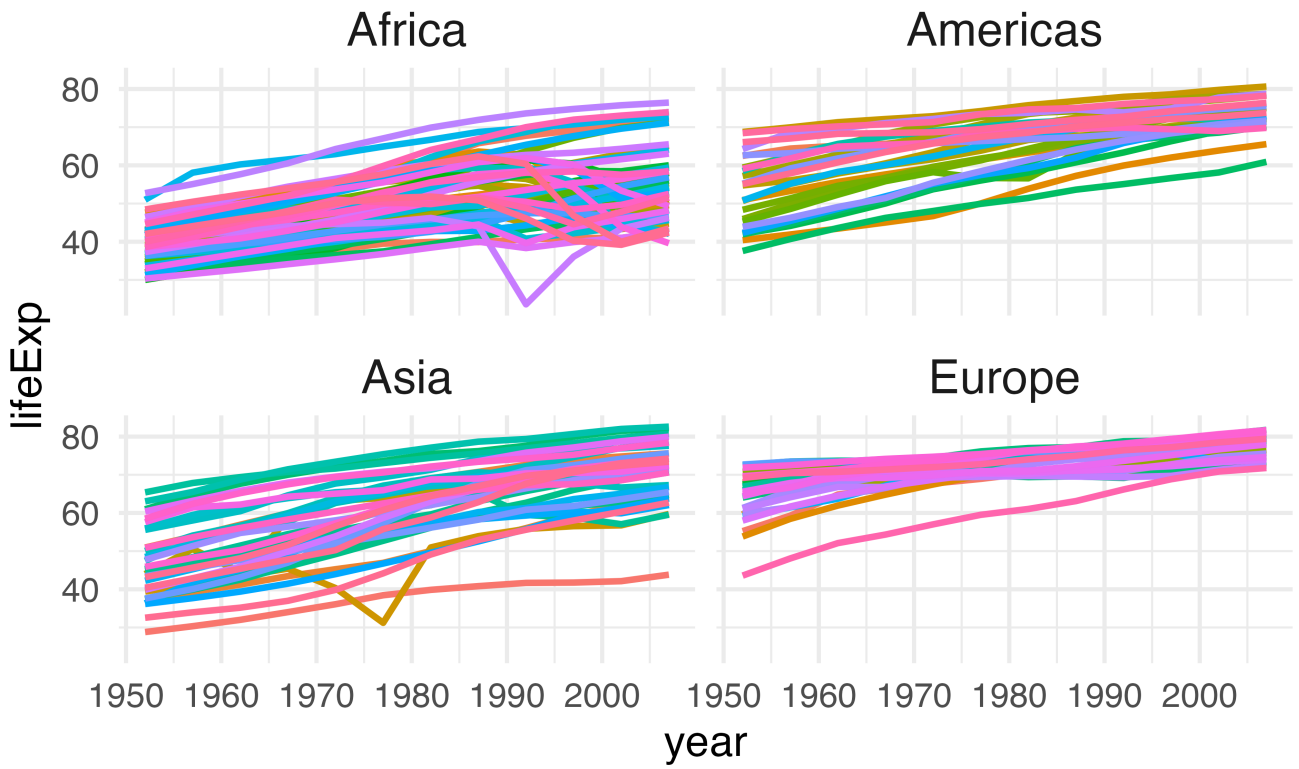
```
tar_plan(
  tar_file(gapminder_file, "gapminder.csv"),
  gapminder = read_csv(gapminder_file, col_types = cols()),
  plot = create_line_plot(gapminder)
)
```

See also tar_url(), tar_parquet(), and other file formats

Your Turn 2

Work through Your Turn 2 in exercises.Rmd

```
tar_read(plot)
```

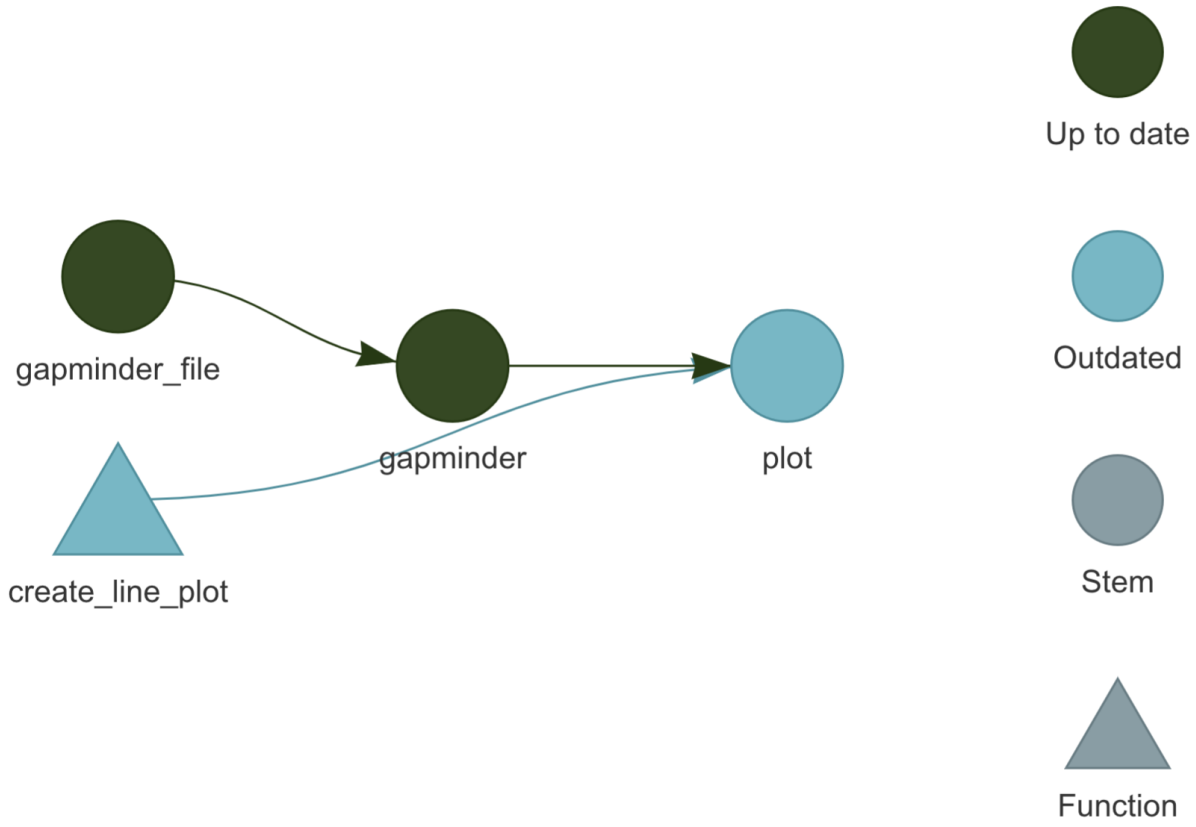


```r
create_line_plot <- function(gapminder) {
  gapminder %>%
    filter(continent != "Oceania") %>%
    ggplot(aes(
      x = year,
      y = lifeExp,
      group = country,
      color = country
    )) +
    geom_line(lwd = 1, show.legend = FALSE) +
    facet_wrap(~ continent) +
    scale_color_manual(values = country_colors) +
    theme_minimal(14) +
    theme(strip.text = element_text(size = rel(1.1)))
}
```

_targets.R

```
library(targets)
library(tarchetypes)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = c("tidyverse", "gapminder"))
source("R/functions.R")

tar_plan(
  tar_file(gapminder_file, "gapminder.csv"),
  gapminder = read_csv(gapminder_file, col_types = cols()),
  plot = create_line_plot(gapminder)
)
```

tar_visnetwork()



Outdated targets

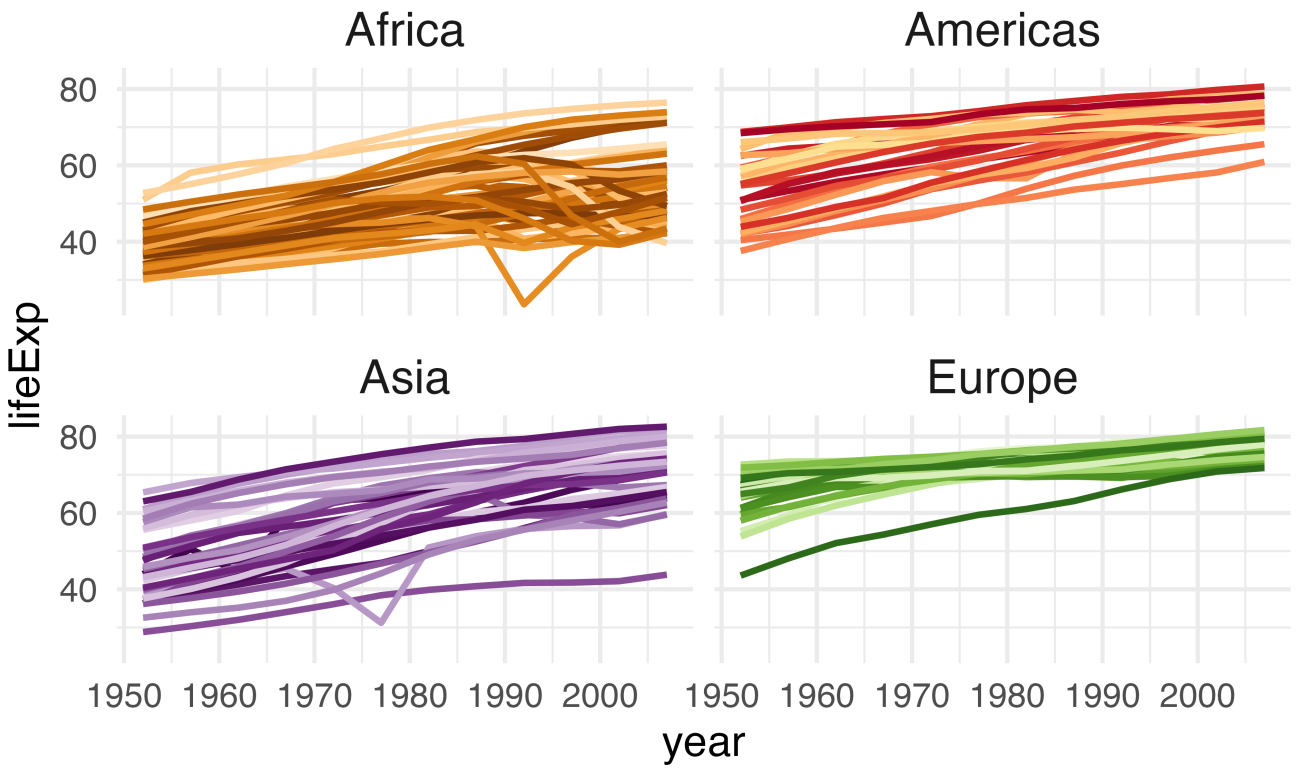
```
tar_outdated()
```

```
## [1] "plot"
```

tar_make()

```
✓ skip target gapminder_file
✓ skip target gapminder
• start target plot
• built target plot
• end pipeline
```

```
tar_read(plot)
```



Your Turn 3

Change diabetes_file **to use** diabetes.csv **instead**

Run tar_outdated() **in the console. What's this telling you? Confirm with** tar_visnetwork()

Predict which targets are going to re-run, then run tar_make()**. Were you right?**

Confirm that diabetes **has changed by looking at it with** tar_read()**. The new dataset should have 403 rows and 20 columns.**

Building up your pipeline

Good targets are meaningful units of your analysis or important dependencies like files

Building up your pipeline

Good targets are meaningful units of your analysis or important dependencies like files

Add one or two targets at a time

Building up your pipeline

Good targets are meaningful units of your analysis or important dependencies like files

Add one or two targets at a time

Run tar_make() **and** tar_visnetwork() **often**

Building up your pipeline

Good targets are meaningful units of your analysis or important dependencies like files

Add one or two targets at a time

Run tar_make() **and** tar_visnetwork() **often**

Load all targets: tar_load(everything())

Organizing your functions

functions.R **only works well for small pipelines, but targets doesn't have a preference for you organize them**

Organizing your functions

functions.R **only works well for small pipelines, but targets doesn't have a preference for you organize them**

> While you're free to arrange functions into files as you wish, [avoid] the two extremes... don't put all functions into one file and don't put each function into its own separate file. —R Packages, 2nd Ed.

What I like to do

1. **Create an** R/ **folder**
2. **Create and open new files in** R/ **with** use_r("file_name") **from the usethis package**
3. **In** _targets.R**:** purrr::walk(fs::dir_ls("R"/), source)

Remember, you need to source() **each file in** _targets.R

Your Turn 4

Add source("R/functions.R") **to** _targets.R

Add "gtsummary" to the packages **argument of** tar_option_set()

Create a new target called table_one **using** create_table_one(diabetes)

Run tar_visnetwork() **and** tar_make()

Take a look at the target you just created

Your Turn 5

Open R/functions.R **and modify** create_table_one()**: Add the argument** missing_text = "(Missing)" **to** tbl_summary()**. Make sure to save your file after you've made the change.**

Run tar_outdated() **in the console, then look at** tar_visnetwork()

Predict which targets are going to re-run, then run tar_make()**. Were you right?**

Including R Markdown files as targets

1. **Create an R Markdown file**
2. **Use** `tar_read()` **or** `tar_load()` **in the Rmd file to access targets**
3. **Include** `tar_render(target_name, "file_name.Rmd")` **in your list of targets**

report.Rmd

````
```{r setup, include = FALSE}
knitr::opts_chunk$set(echo = FALSE)
library(targets)

tar_load(gapminder)
```


These data have `r nrow(gapminder)`
observations.

```{r figure-one, fig.cap = "Figure 1"}
tar_read(plot)
```
````

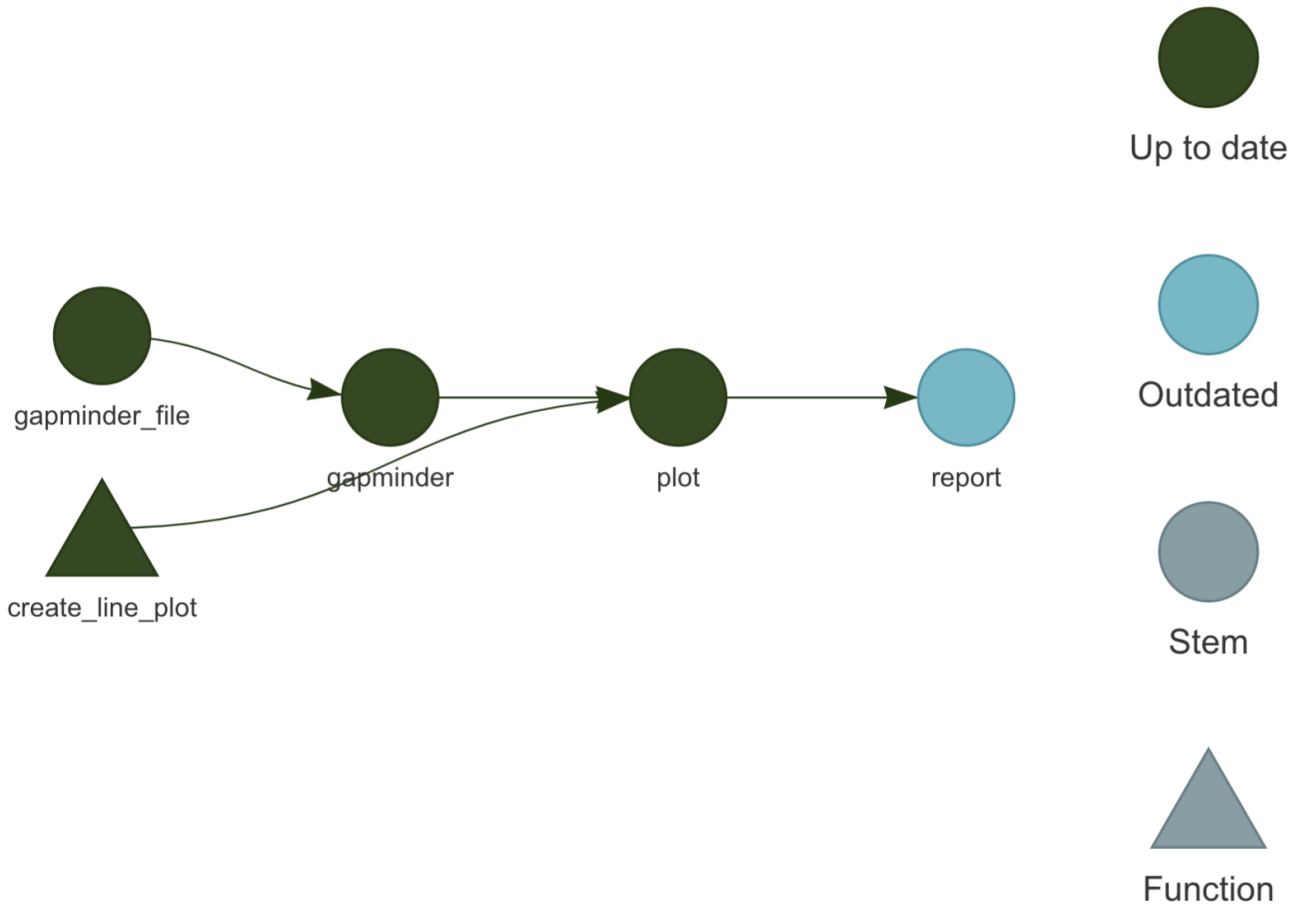
report.Rmd

````
```{r setup, include = FALSE}
knitr::opts_chunk$set(echo = FALSE)
library(targets)

tar_load(gapminder)
```

These data have `r nrow(gapminder)`
observations.

```{r figure-one, fig.cap = "Figure 1"}
tar_read(plot)
```
````

_targets.R

```
library(targets)
library(tarchetypes)
options(tidyverse.quiet = TRUE)
tar_option_set(packages = c("tidyverse", "gapminder"))
source("R/functions.R")

tar_plan(
  tar_file(gapminder_file, "gapminder.csv"),
  gapminder = read_csv(gapminder_file, col_types = cols()),
  plot = create_line_plot(gapminder),
  tar_render(report, "report.Rmd")
)
```

tar_visnetwork()



tar_make()

```
✓ skip target gapminder_file
✓ skip target gapminder
✓ skip target plot
• start target report
• built target report
• end pipeline
```

Your Turn 6

Work through Your Turn 6 in exercises.Rmd

The targets cache: _targets/

tar_destroy(): **Remove everything in the cache**

tar_prune(): **Remove targets that are no longer part of the pipeline**

Your Turn 7

Confirm that you can reproduce your entire pipeline from scratch. In the console:

Run tar_destroy()

Run tar_make()

Other features of targets

Automatic parallelization

Other features of targets

Automatic parallelization

Branching

Other features of targets

Automatic parallelization

Branching

Cloud integration

Resources

The targets User Manual: A comprehensive but friendly introduction to targets. Free online.

The Official targets Short Course: A free short course available on RStudio Cloud or locally

Talks and other targets resources