Reproducible computation at scale in R

Will Landau

Purpose: manage data analyses with long runtimes

Scale up the
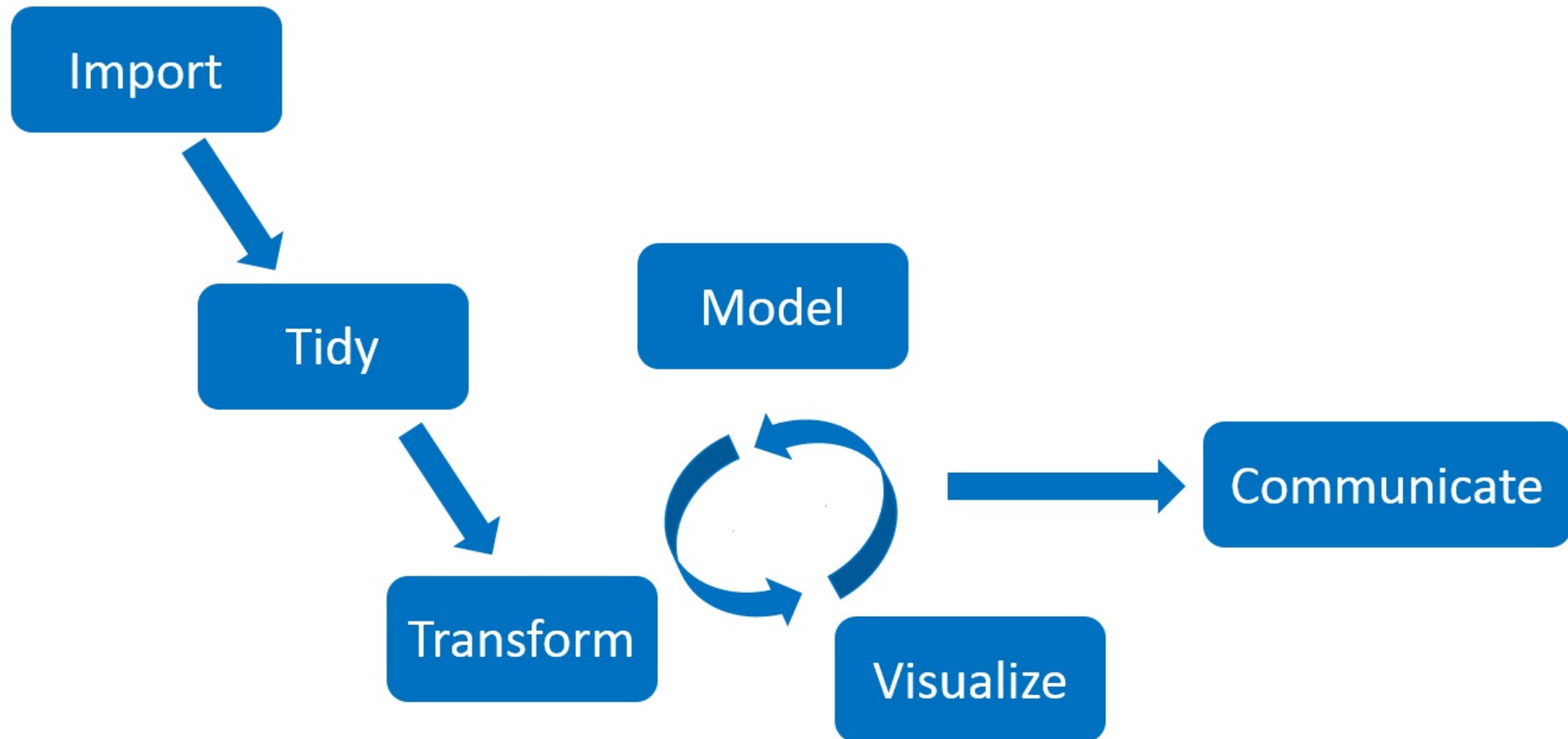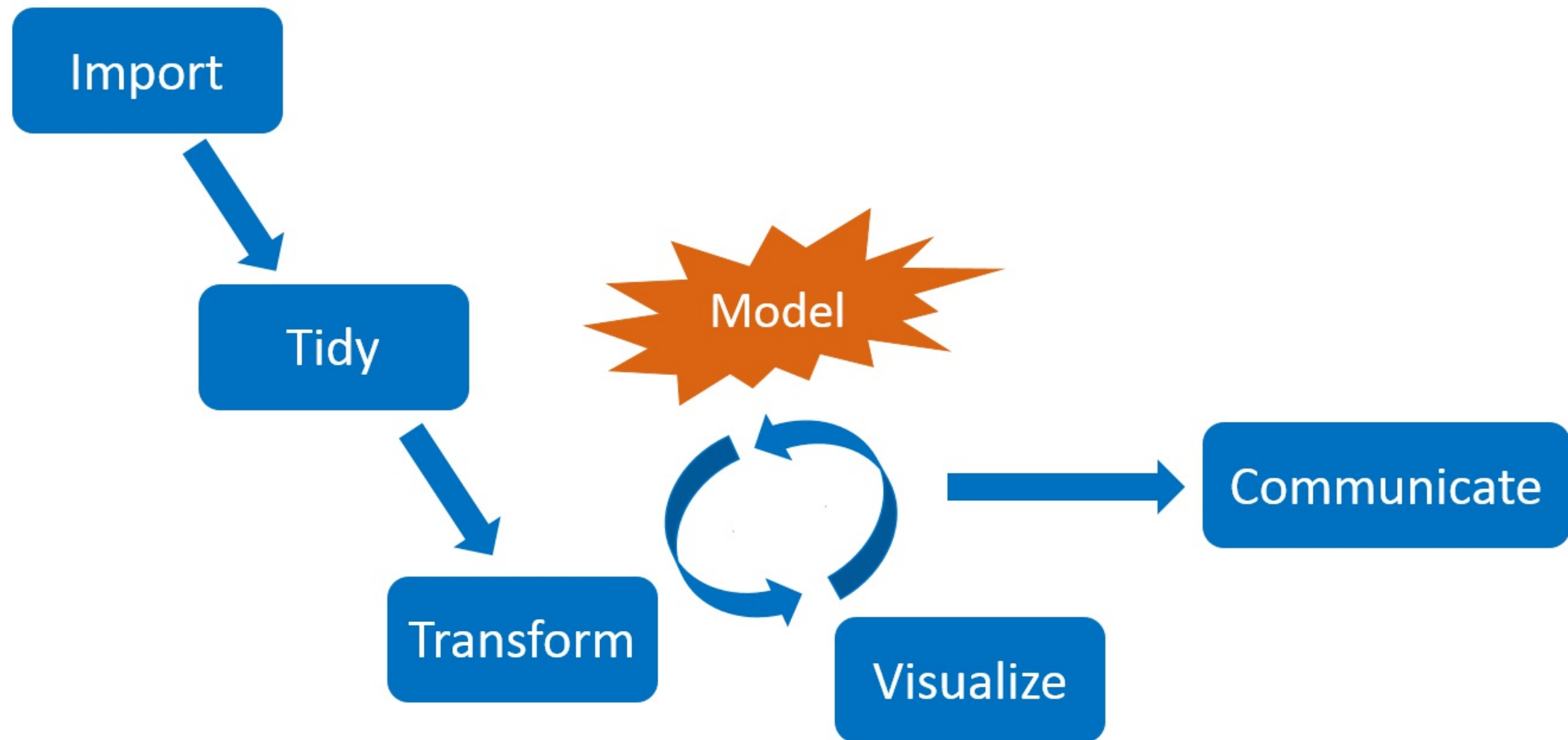work you need.

Skip the
work you don't.

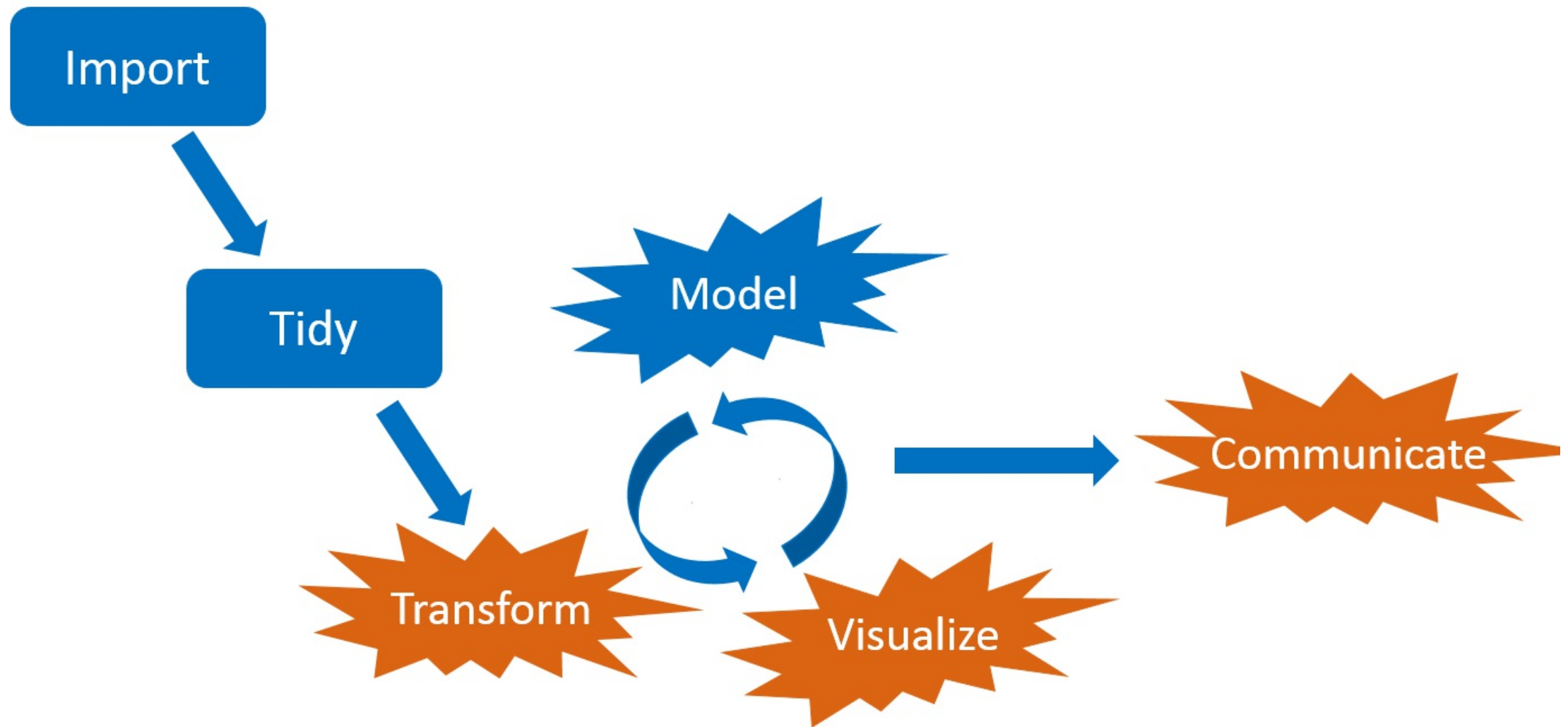**See** evidence of
reproducibility.

Workflows have interconnected steps.

When you change something...

...the downstream steps are no longer valid.

Do you rerun everything from scratch?

- Takes too long.
- Too frustrating.

https://openclipart.org/detail/275842/sisyphus-overcoming-silhouette

Do you pick and choose pieces to update?

- Messy and prone to human error.
- Not reproducible.

https://openclipart.org/detail/216179/messy-desk

Pipeline toolkits solve this problem.

- Sophisticated, vibrant, active space of tools: github.com/pditommaso/awesome-pipeline.
- Most are language-agnostic or designed for other languages.
- **drake** is uniquely devoted to R.
  - A focus on ordinary R functions and variables rather than cumbersome files.
  - Heavy use of the data frame, even as a substitute for the traditional Makefile.
  - Native tidy evaluation support.
  - A domain-specific language (DSL) for creating large workflows.

Example uses in the pharmaceutical industry

- Clinical trial modeling and simulation
- Subgroup identification
- Bayesian network meta analysis
- Graph-based multiple comparison procedures
- Bayesian networks in genomics
- PK/PD modeling (e.g. `mrgsolve`)
- **Deep learning**

Example deep learning workflow

- Goal: predict customers who cancel their subscriptions with a telecom company.
- Data: IBM Watson Telco Customer Churn dataset.
- Workflow principles generalize to pharma, e.g. business analytics and genomics problems.

https://openclipart.org/detail/90739/newplus, https://github.com/rstudio/keras

File structure

```
make.R
R/
├── packages.R
├── functions.R
└── plan.R
data/
└── customer_churn.csv
```

packages.R

```r
library(drake)
library(keras)
library(recipes)
library(rsample)
library(tidyverse)
library(yardstick)
```

functions.R

```r
prepare_recipe <- function(data) {
  # ...
}

define_model <- function(rec) {
  # ...
}

train_model <- function(data, rec, units1, units2, act1, act2, act3) {
  # ...
}

confusion_matrix <- function(data, rec, serialized_model) {
  # ...
}

compare_models <- function(...) {
  # ...
}
```
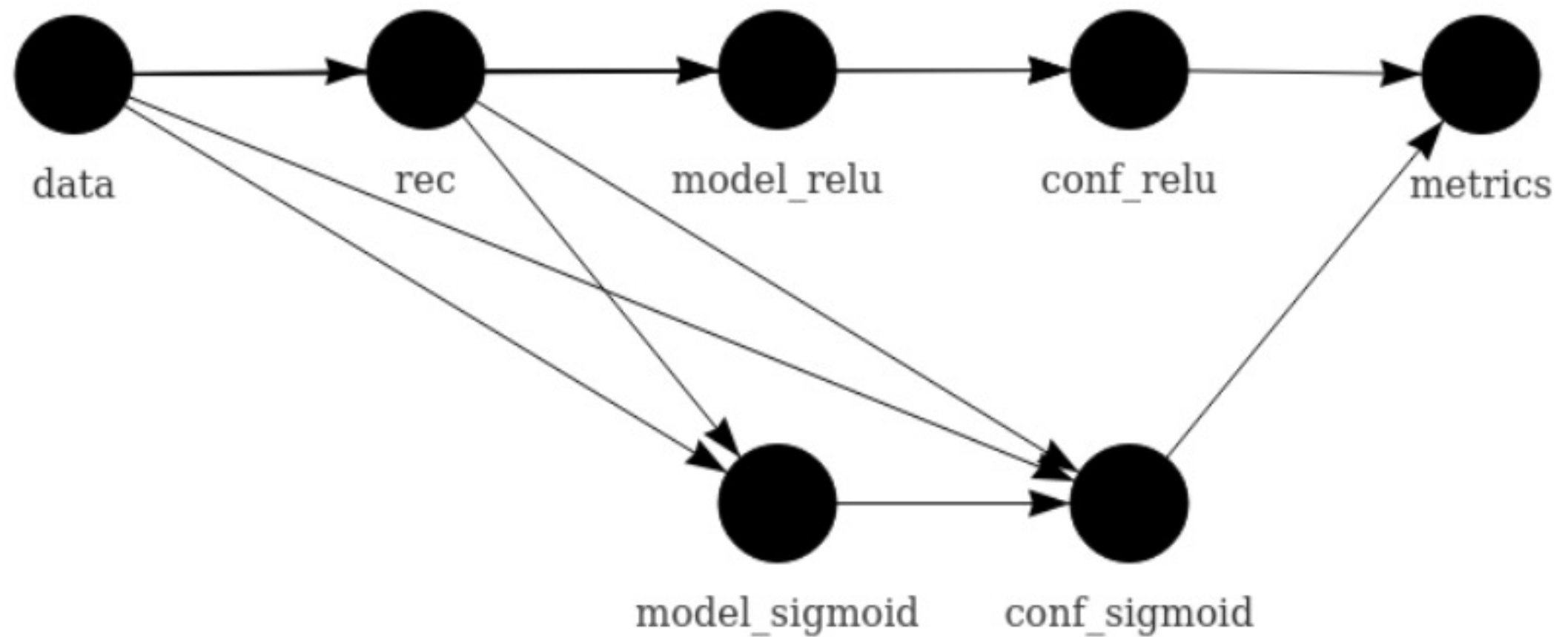
plan.R

```r
activations <- c("relu", "sigmoid")

plan <- drake_plan(
  data = read_csv(file_in("data/customer_churn.csv"), col_types = cols()) %>%
    initial_split(prop = 0.3),
  rec = prepare_recipe(data),
  model = target(
    train_model(data, rec, act1 = act),
    transform = map(act = !!activations)
  ),
  conf = target(
    confusion_matrix(data, rec, model),
    transform = map(model, .id = act)
  ),
  metrics = target(
    compare_models(conf),
    transform = combine(conf)
  )
)
```

Data frame of workflow steps

```
plan
## # A tibble: 7 x 2
##   target       command
##   <chr>        <expr>
## 1 data         read_csv(file_in("data/customer_churn.csv"), col_types = c
## 2 rec          prepare_recipe(data)
## 3 model_relu   train_model(data, rec, act1 = "relu")
## 4 model_sigmo… train_model(data, rec, act1 = "sigmoid")
## 5 conf_relu    confusion_matrix(data, rec, model_relu)
## 6 conf_sigmoid confusion_matrix(data, rec, model_sigmoid)
## 7 metrics      compare_models(conf_relu, conf_sigmoid)
```
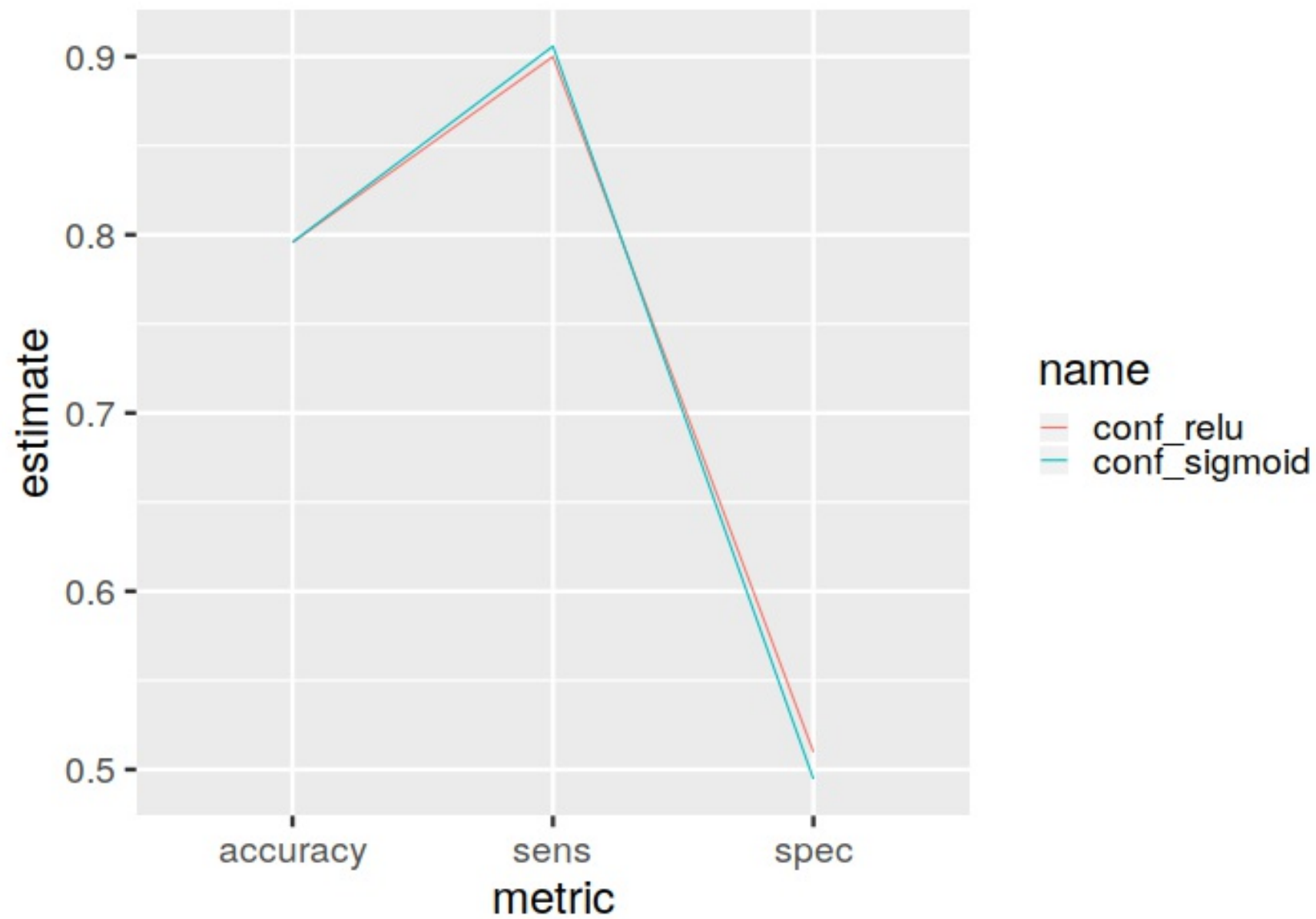
The workflow

Run the project in make.R.

```
source("R/packages.R")
source("R/functions.R")
source("R/plan.R")

make(plan)
## target data
## target rec
## target model_relu
## target model_sigmoid
## target conf_relu
## target conf_sigmoid
## target metrics
```

Compare models.

```
readd(metrics) # See also loadd()
```

Try another activation function.

```
activations <- c("relu", "sigmoid", "softmax")

plan <- drake_plan(
  data = read_csv(file_in("data/customer_churn.csv"), col_types = cols()) %>%
    initial_split(prop = 0.3),
  rec = prepare_recipe(data),
  model = target(
    train_model(data, rec, act1 = act),
    transform = map(act = !!activations)
  ),
  conf = target(
    confusion_matrix(data, rec, model),
    transform = map(model, .id = act)
  ),
  metrics = target(
    compare_models(conf),
    transform = combine(conf)
  )
)
```
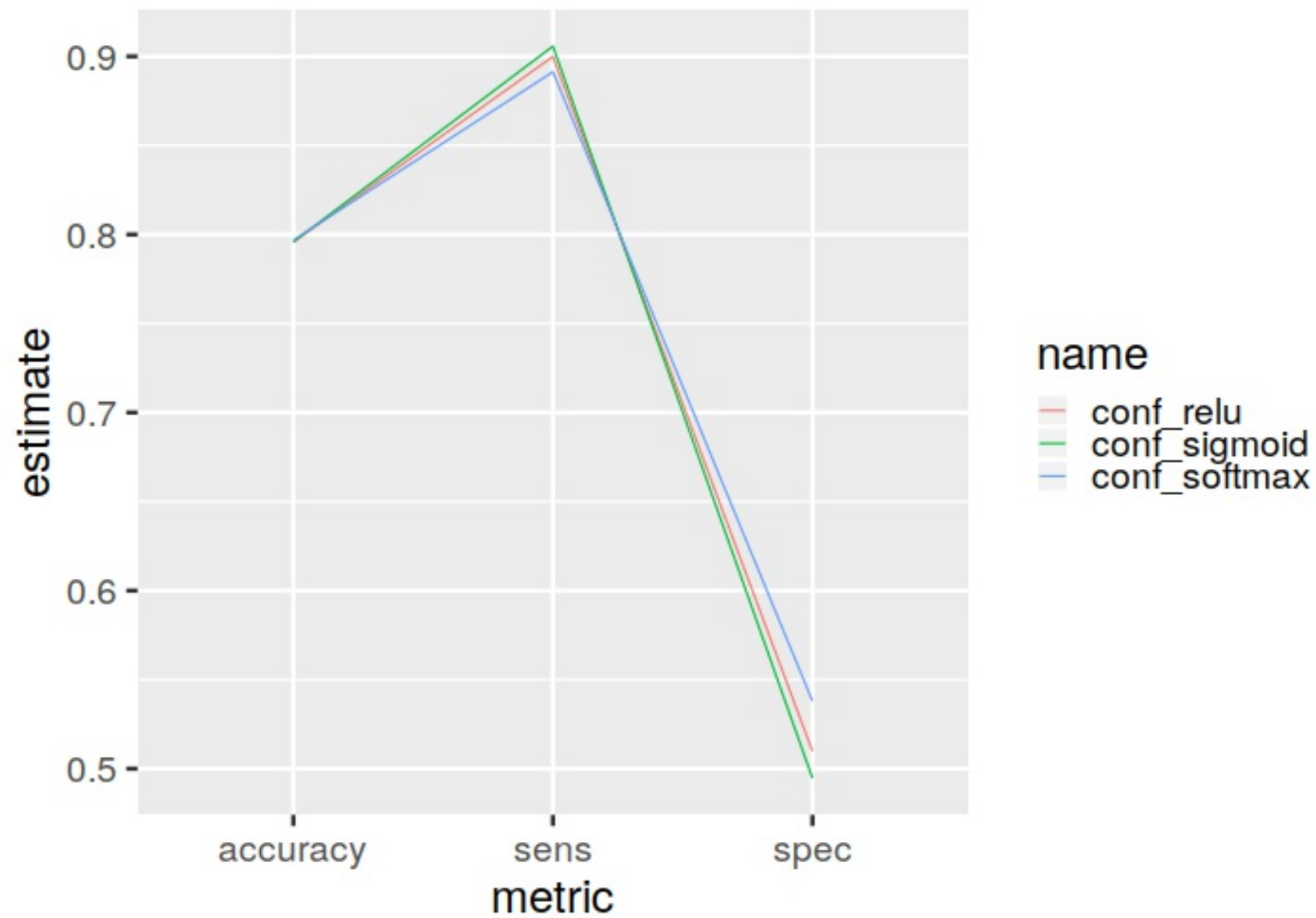
vis_drake_graph()
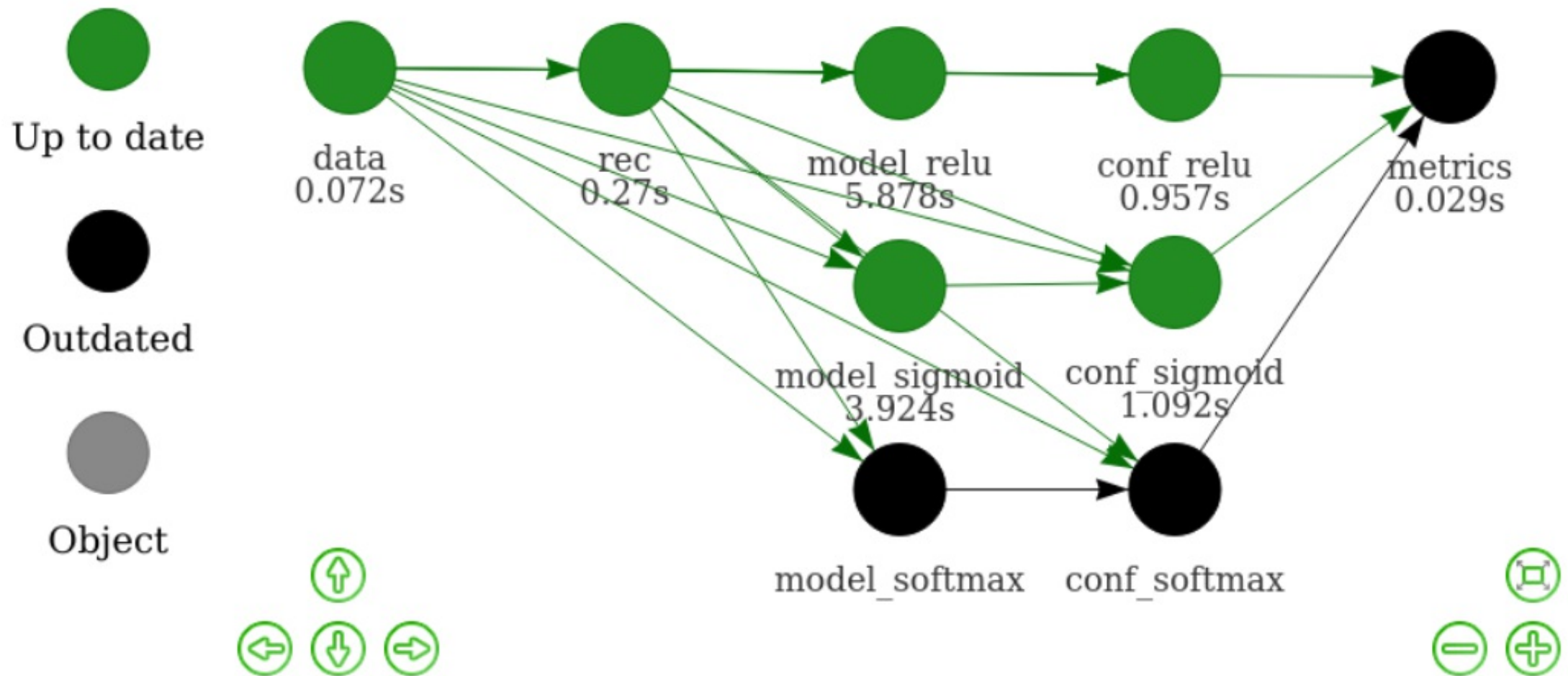
Refresh the results in make.R.

```r
source("R/packages.R")
source("R/functions.R")
source("R/plan.R") # modified

make(plan)
## target model_softmax
## target conf_softmax
## target metrics
```

Compare models.

```
readd(metrics)
```

Evidence of reproducibility

```r
source("R/packages.R")
source("R/functions.R")
source("R/plan.R")

make(plan)
## All targets are already up to date.
```

- See also outdated().

High-performance computing

```r
# template file with configuration
drake_hpc_template_file("slurm_clustermq.tmpl")

# Use SLURM resource manager with the template.
options(
  clustermq.scheduler = "slurm",
  clustermq.template = "slurm_clustermq.tmpl"
)

# make() is the basically the same.
make(plan, jobs = 2, parallelism = "clustermq")
```

High-performance computing

Resources

```
install.packages("drake")                      # release
devtools::install_github("ropensci/drake") # development
```

- Download today's code: `drake_example("customer-churn-simple")`
- Faster model storage: `drake_example("customer-churn-fast")`
- Reference website
- Full user manual
- Example workflows
- File an issue.
- Contribute code.
- Discuss at rOpenSci.org.

Thanks

- Edgar Ruiz
- example code

- Matt Dancho
- blog post

Thanks

- Maëlle Salmon
- Ben Marwick
- Julia Lowndes
- Peter Slaughter
- Jenny Bryan
- Rich FitzJohn
- Stefanie Butland

- Jarad Niemi
- Kirill Müller
- Henrik Bengtsson
- Michael Schubert
- Kendon Bell
- Miles McBain
- Patrick Schratz
- Alex Axthelm
- Jasper Clarkberg
- Tiernan Martin
- Ben Listyg
- TJ Mahr
- Ben Bond-Lamberty
- Tim Mastny
- Bill Denney
- Amanda Dobbyn
- Daniel Falster
- Rainer Krug
- Brianna McHorse
- Chan-Yub Park