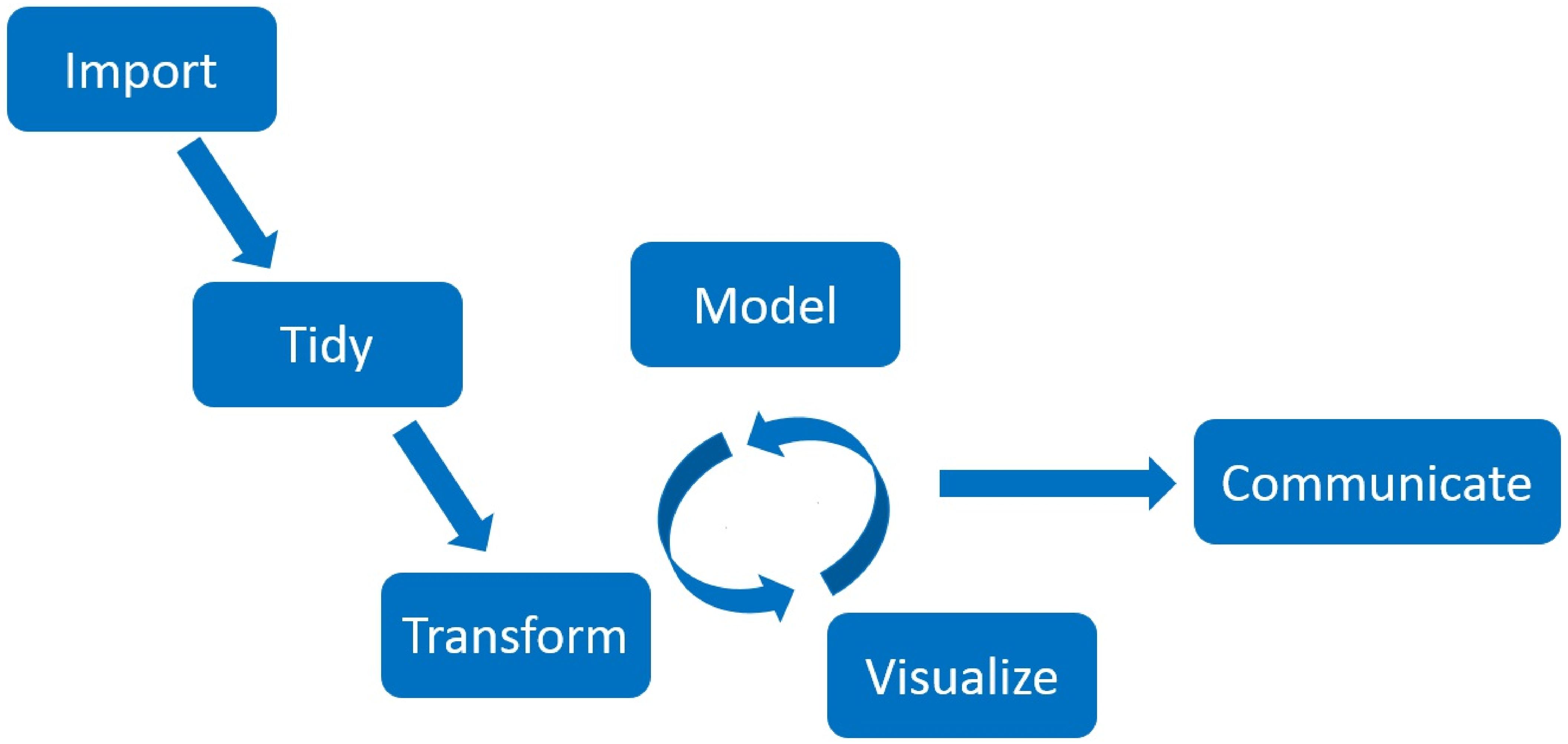
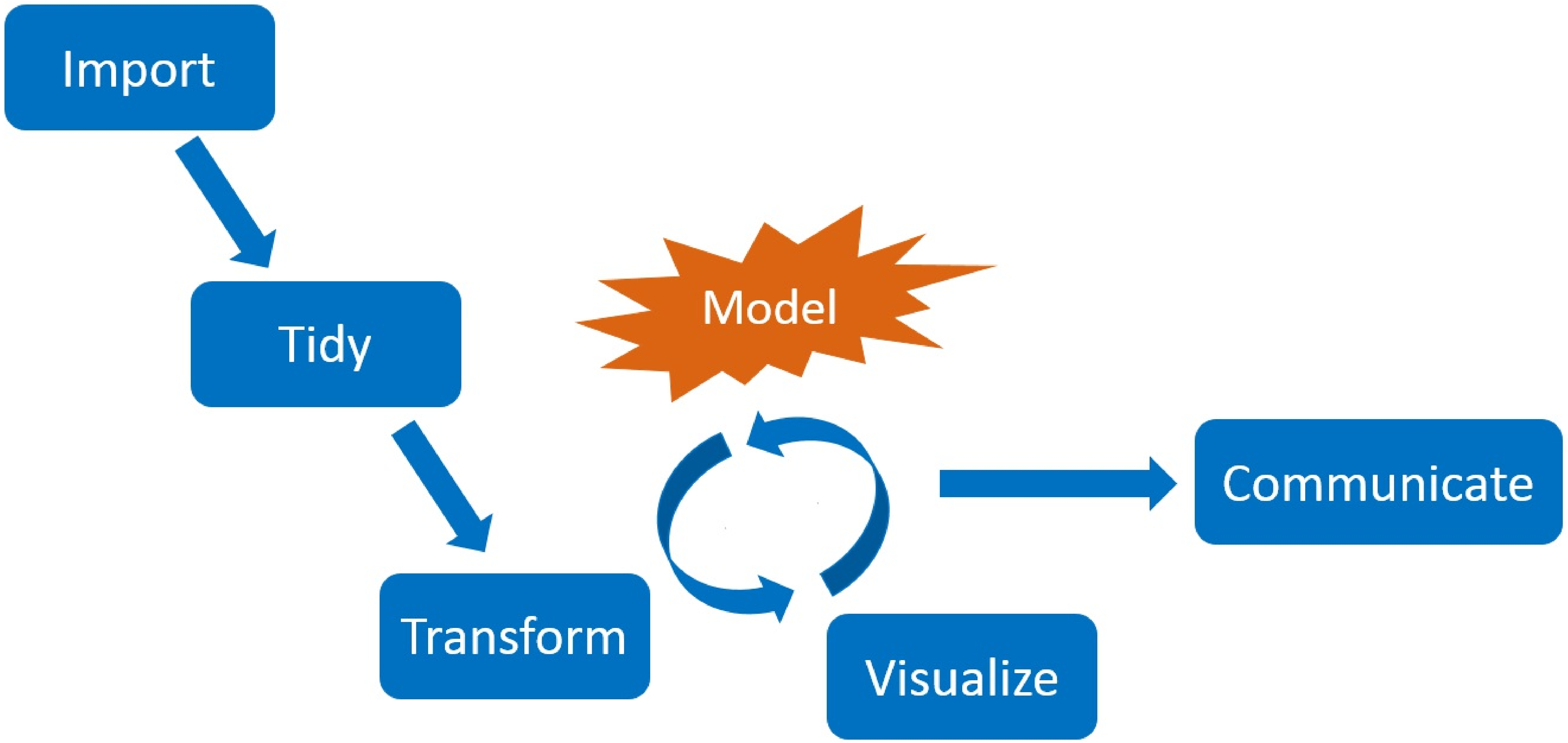
Reproducible workflows at scale with drake

Will Landau

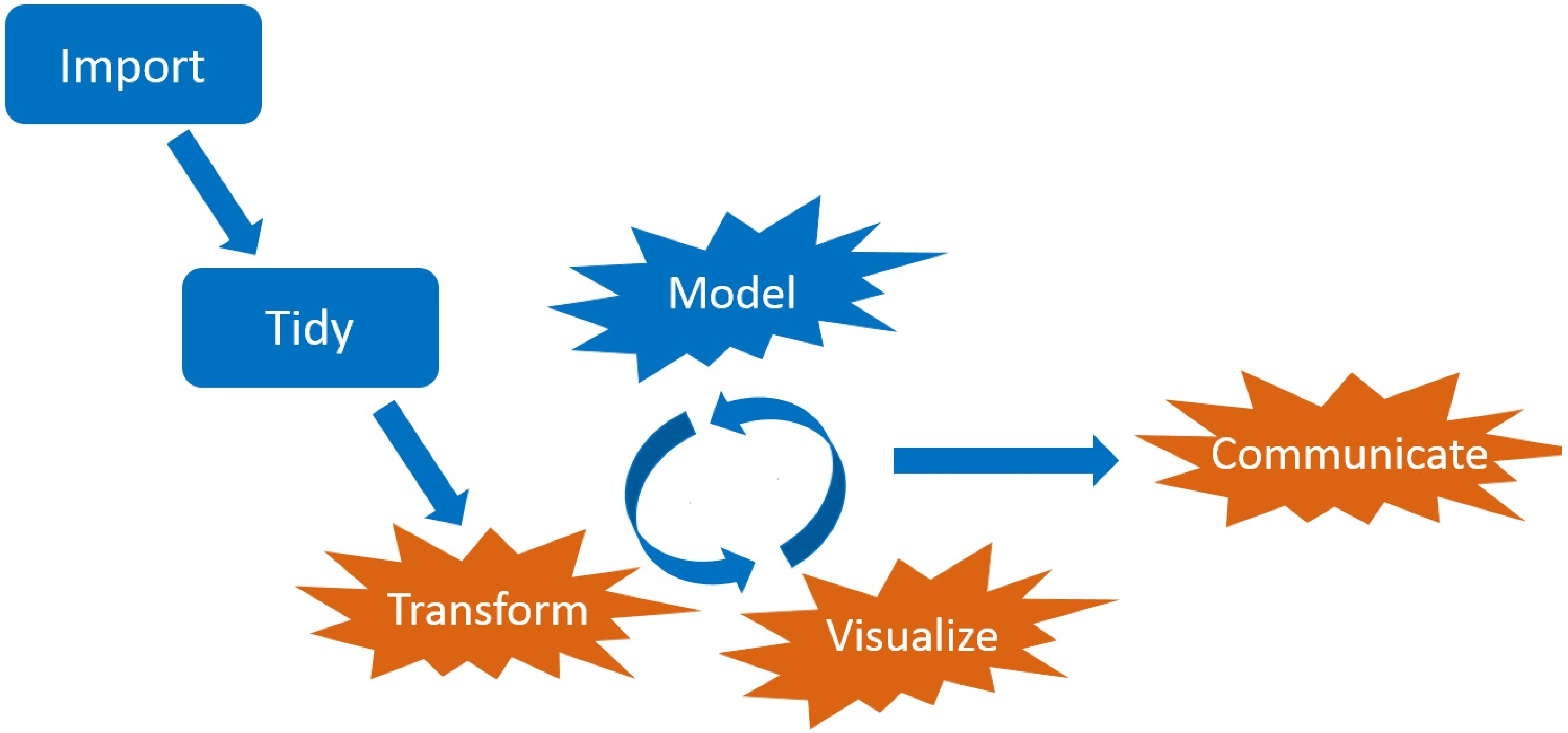
Workflows have interconnected steps.



When you change something...



...the downstream steps are no longer valid.



Do you rerun everything from scratch?

- Not if you deal with long runtimes!

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

Do you pick and choose what to update?

- Messy.
- Prone to human error.
- Not reproducible.

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

When do we face these issues?

Long computation!

- **Machine learning**
- Bayesian data analysis
- Social network analysis
- Spatial statistics
- Econometrics
- Bayesian causal networks
- Clinical trial simulation
- PK/PD modeling (e.g. `mrgsolve`)
- ...

Solution: pipeline tools

Scale up the work you need.

Skip the work you don't.

See evidence of reproducibility.

- Tons exist already: github.com/pditommaso/awesome-pipeline.
- Most are language-agnostic or designed for Python or the shell.

What makes `drake` different?

- Aggressively designed for R.
  - Think **functions**, not script files.
  - Think **variables**, not output files.
  - Think **data frames**, not `Makefiles`.
- Major improvements in late 2018 and early 2019:
  - A domain-specific language for workflows.
  - Massive improvements in speed and memory usage.
  - Special functions and checks to safeguard reproducibility.

Example: a 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 other industries.

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, units1, units2, act1, act2, act3) {
  # ...
}

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

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

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

plan.R

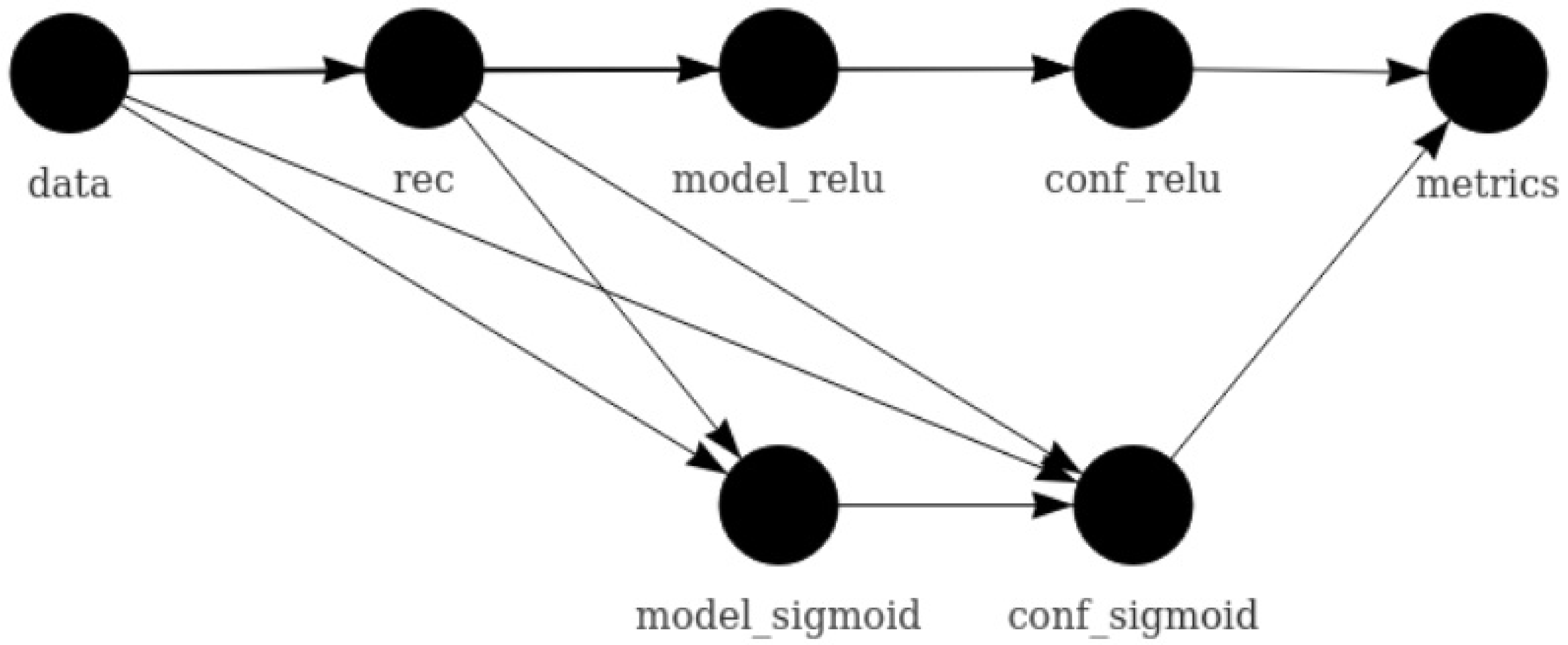
```r
activations <- c("relu", "sigmoid")

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



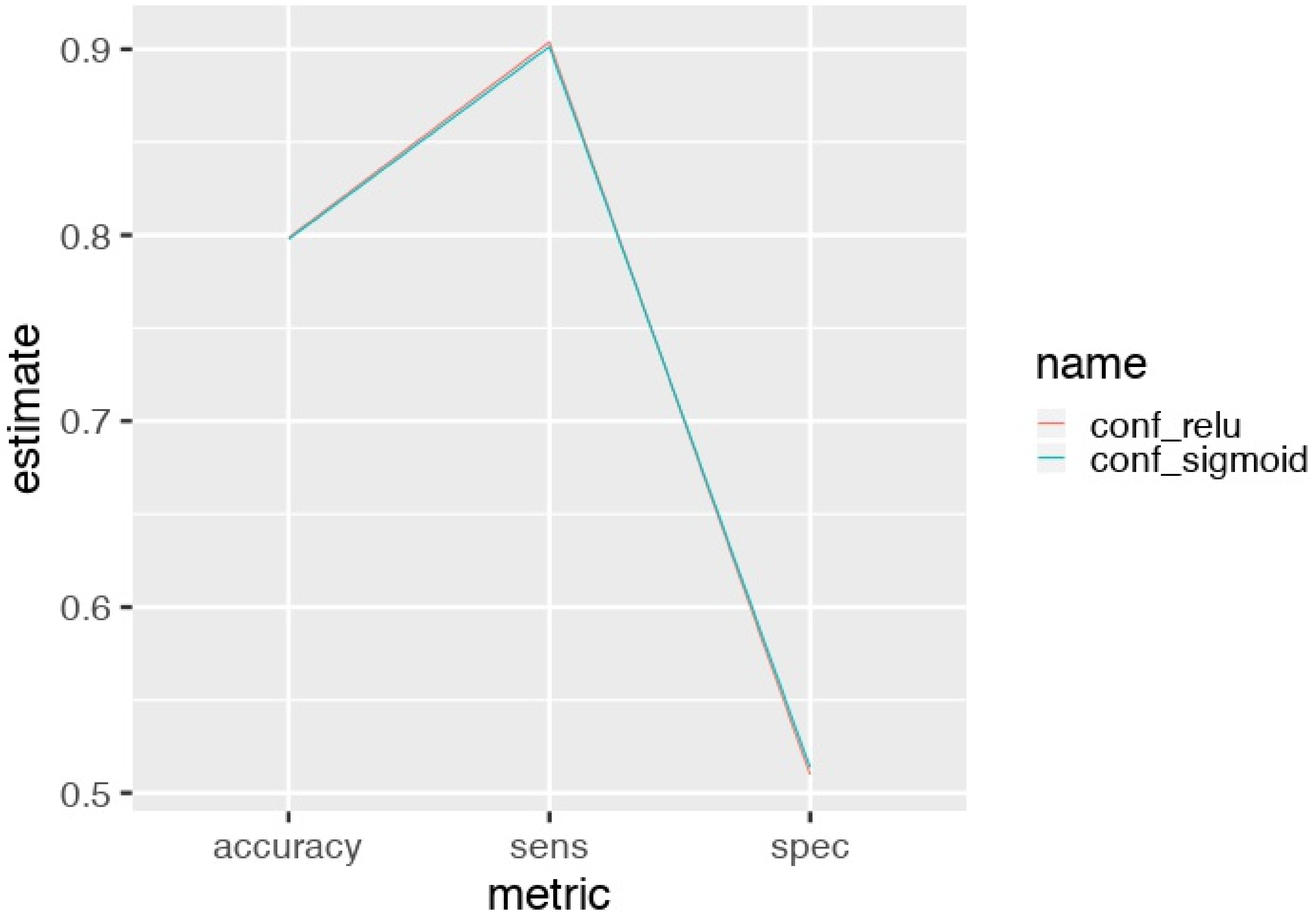
Run the project in make.R.

```
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()
```

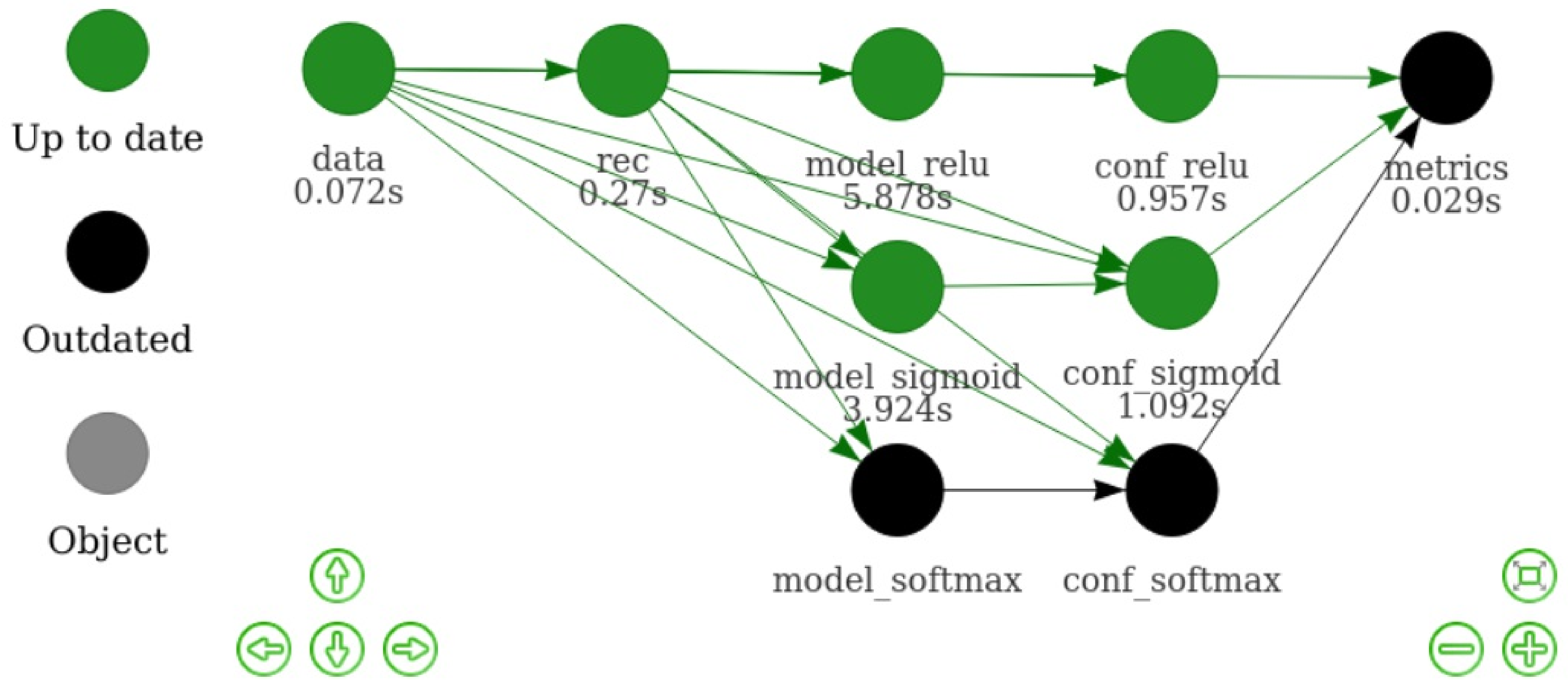


Try another activation function.

```r
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()



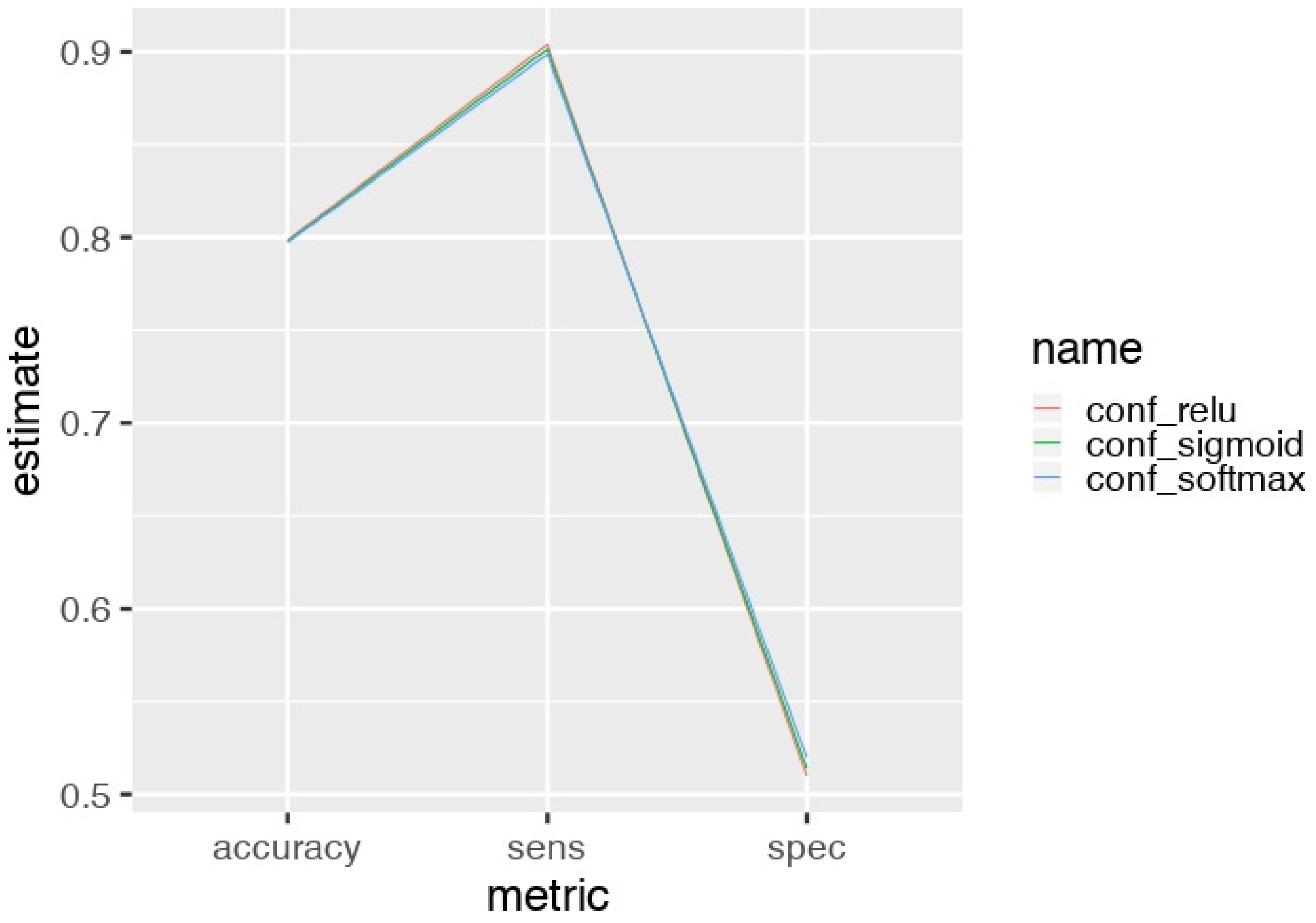
Refresh the results in make.R.

```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()`.

History and provenance

```
history <- drake_history()
history
## # A tibble: 10 x 9
##    target  time       hash   exists command         runtime latest  prop act
##    <chr>   <chr>      <chr>  <lgl>  <chr>             <dbl> <lgl>  <dbl> <ch
##  1 conf_r… 2019-07-… 85315… TRUE   confusion_ma…     1.63  TRUE     NA   <NA
##  2 conf_s… 2019-07-… e2212… TRUE   confusion_ma…     1.83  TRUE     NA   <NA
##  3 conf_s… 2019-07-… 934e8… TRUE   confusion_ma…     2.64  TRUE     NA   <NA
##  4 data    2019-07-… ca84b… TRUE   "read_csv(fi…     0.051 TRUE      0.3 <NA
##  5 metrics 2019-07-… 4f63d… TRUE   compare_mode…     0.024 FALSE    NA   <NA
##  6 metrics 2019-07-… ca8b2… TRUE   compare_mode…     0.02  TRUE     NA   <NA
##  7 model_… 2019-07-… 09fde… TRUE   "train_model…    11.2   TRUE     NA   rel
##  8 model_… 2019-07-… 46764… TRUE   "train_model…     7.36  TRUE     NA   sig
##  9 model_… 2019-07-… 55a46… TRUE   "train_model…     7.82  TRUE     NA   sof
## 10 rec     2019-07-… 40e50… TRUE   prepare_reci…     0.227 TRUE     NA   <NA
```

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

- Get `drake`:

```r
install.packages("drake")
```

- Workshop materials:

```r
remotes::install_github("wlandau/learndrake")
```

- Example code from these slides:

```r
drake::drake_example("customer-churn-simple")
```

Links

- Development repository: https://github.com/ropensci/drake
- Full user manual https://ropenscilabs.github.io/drake-manual
- Reference website: https://ropensci.github.io/drake
- Code examples: https://github.com/wlandau/drake-examples
- Discuss at rOpenSci.org: https://discuss.ropensci.org

rOpenSci use cases

- Use `drake`? Share your use case at https://ropensci.org/usecases.

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