Reproducible workflows at scale with drake

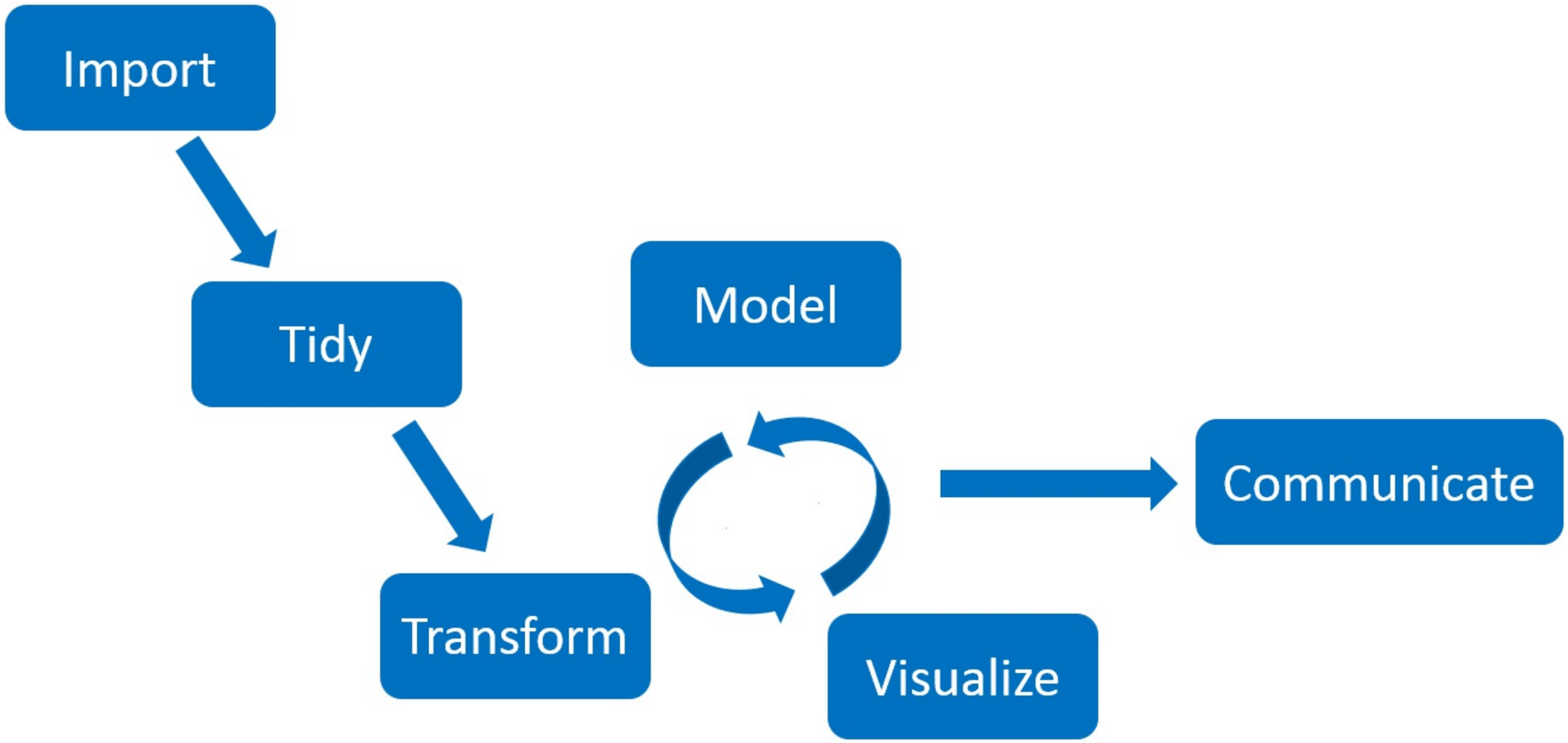
Will Landau

Large data science projects

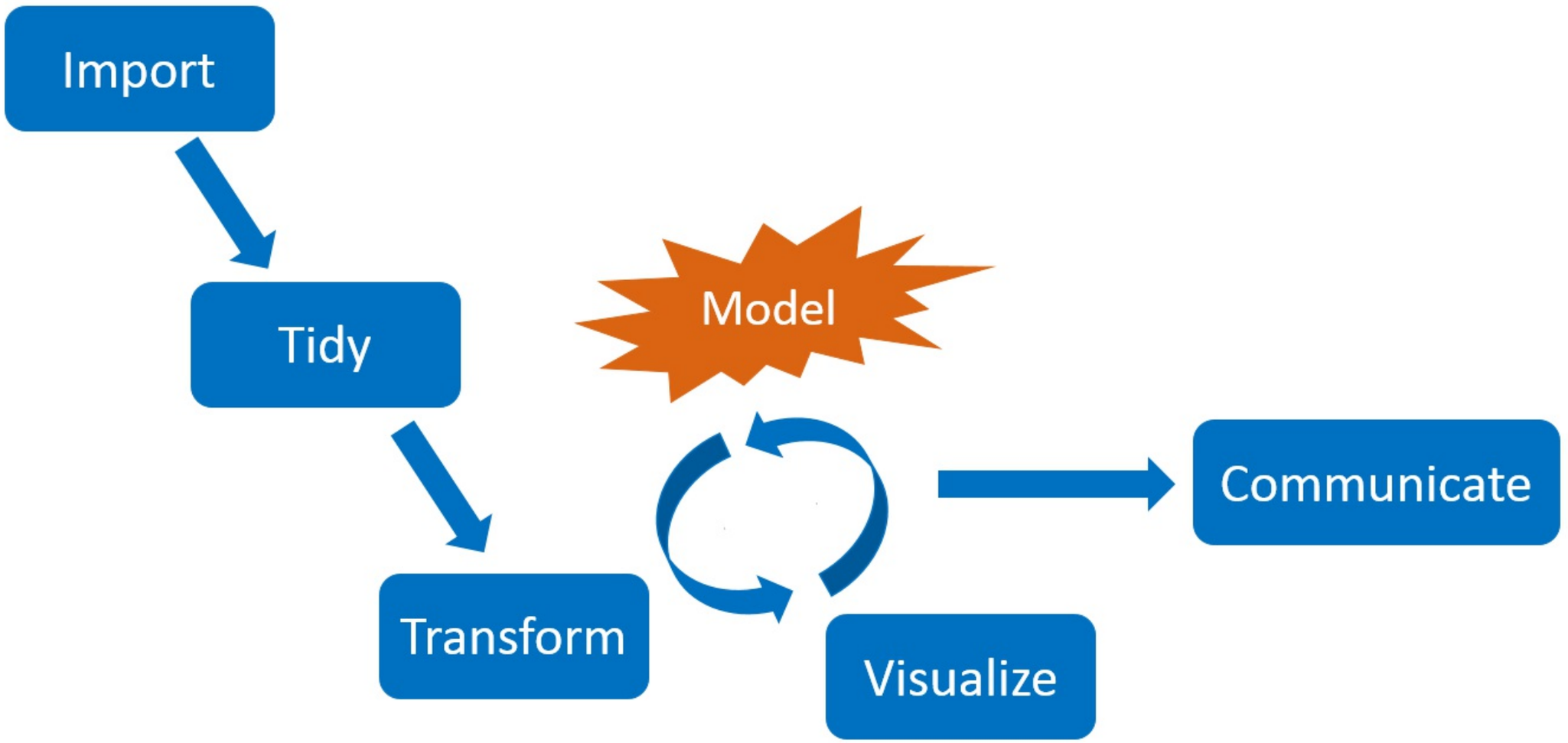
1. Long runtimes.
2. Many steps.
3. Steps are interconnected.

- Deep learning
- Classical machine learning.
- Bayesian computation, e.g. Markov chain Monte Carlo.
- Spatial data analysis.
- Clinical trial modeling and simulation.
- Subgroup identification.
- Graph-based multiple comparison procedures.
- Genomics pipelines.
- PK/PD modeling (e.g. `mrgsolve`)
- ...

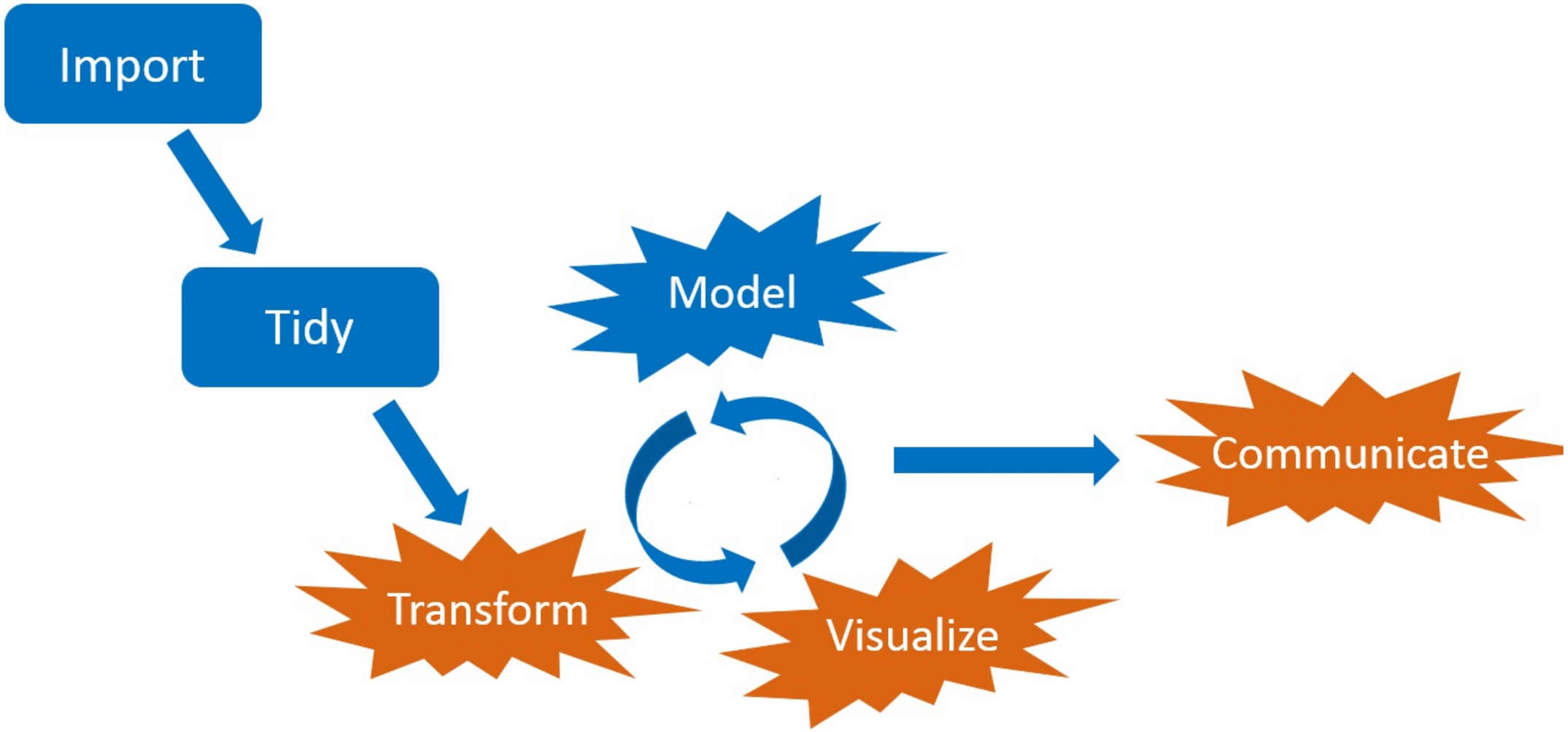
Workflows have interconnected steps.



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...the downstream steps are no longer valid.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?

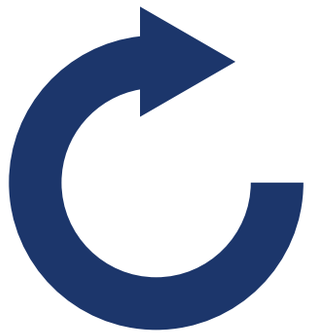
- Messy.
- Prone to human error.
- Not reproducible.

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

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.
  - Increased speed and reduced memory usage.
  - More reproducibility safeguards.
  - History and provenance.
  - Reproducible data recovery and renaming (experimental).

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

```
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, model) {
  # ...
}

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

plan.R

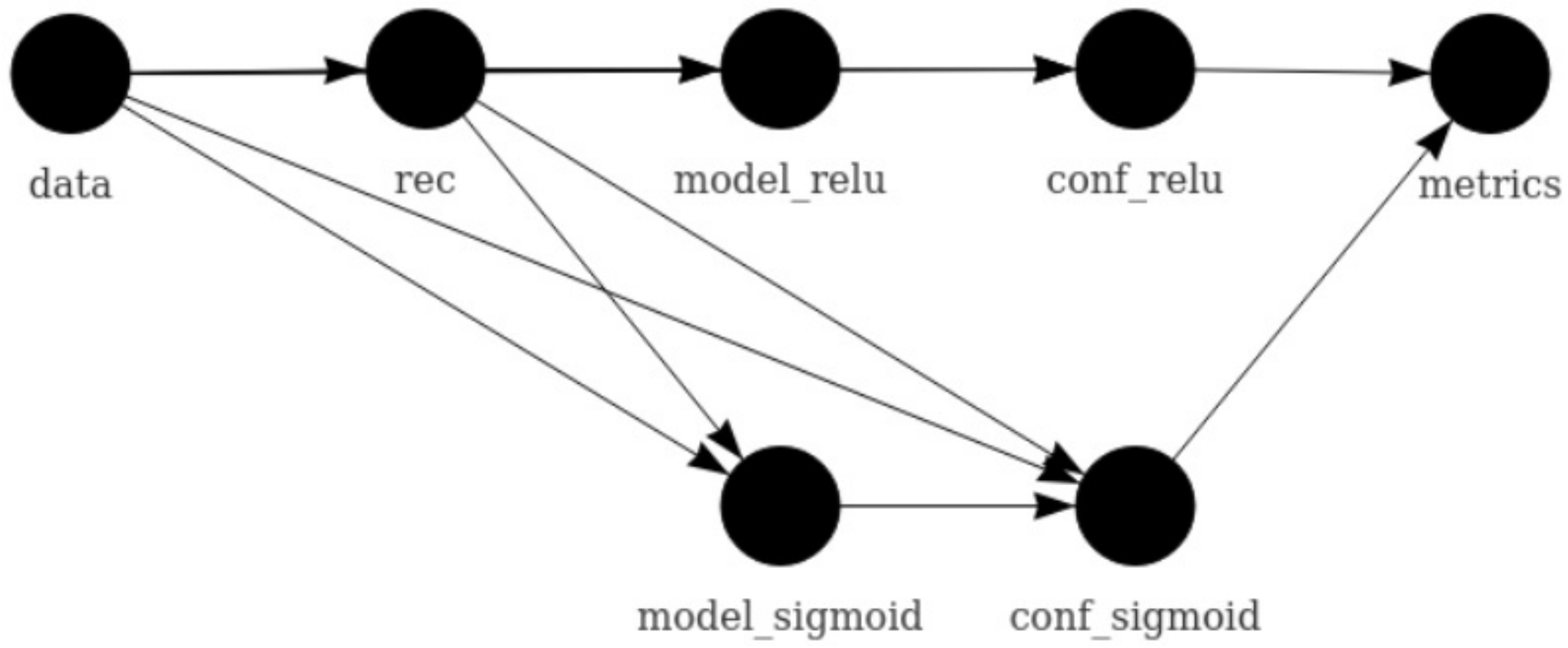
```
activations <- c("relu", "sigmoid")

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),
    format = "keras",
    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 3
##   target      command                                                  forma
##   <chr>       <expr>                                                   <chr
## 1 data        read_csv(file_in("data/customer_churn.csv"), col_type… <NA>
## 2 rec         prepare_recipe(data)                                   … <NA>
## 3 model_relu  train_model(data, rec, act1 = "relu")                   … keras
## 4 model_sigm… train_model(data, rec, act1 = "sigmoid")                … keras
## 5 conf_relu   confusion_matrix(data, rec, model_relu)                 … <NA>
## 6 conf_sigmo… confusion_matrix(data, rec, model_sigmoid)              … <NA>
## 7 metrics     compare_models(conf_relu, conf_sigmoid)                 … <NA>
```

The workflow



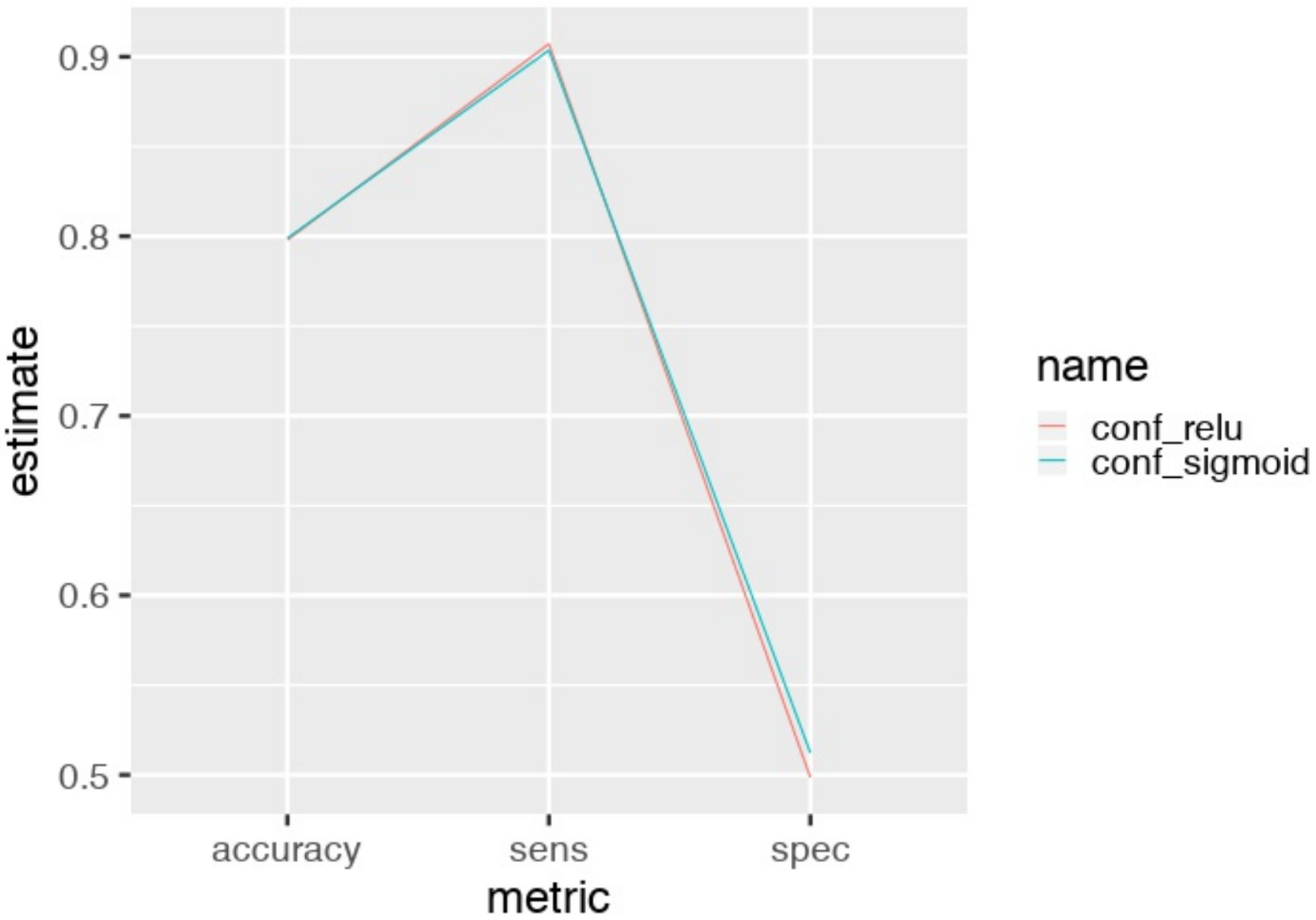
Run the project in make.R.

```
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()
```

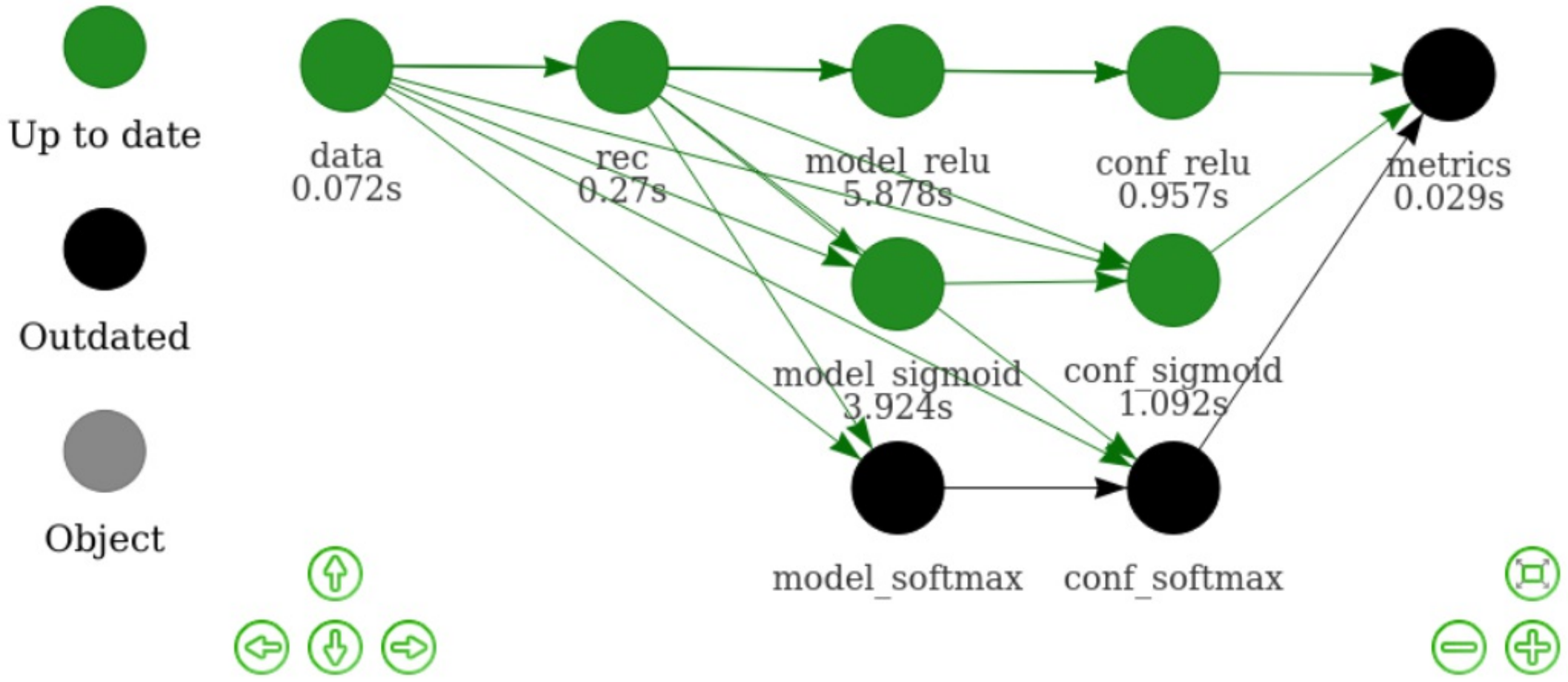


Try another activation function.

```
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),
    format = "keras",
    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()



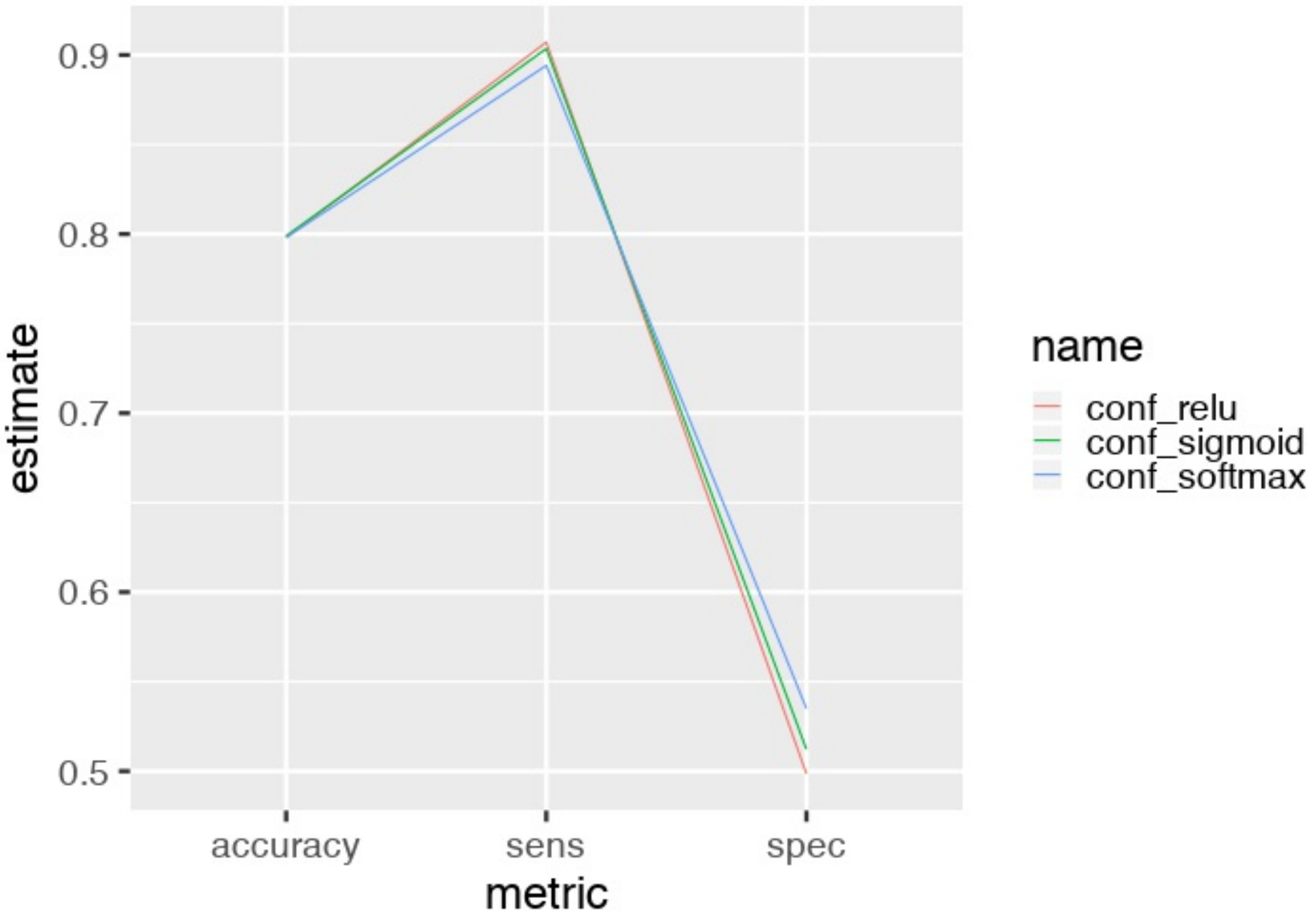
Refresh the results in make.R.

```
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

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

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

- See also `outdated()`.

Specialized data formats

- Increased speed and reduced memory consumption

```
library(drake)
n <- 1e8 # Each target is 1.6 GB in memory.
plan <- drake_plan(
  data_fst = target(
    data.frame(x = runif(n), y = runif(n)),
    format = "fst"
  ),
  data_old = data.frame(x = runif(n), y = runif(n))
)
make(plan)
#> target data_fst
#> target data_old
build_times(type = "build")
#> # A tibble: 2 x 4
#>   target   elapsed                user                   system
#>   <chr>    <Duration>             <Duration>             <Duration>
#> 1 data_fst 13.93s                 37.562s                7.954s
#> 2 data_old 184s (~3.07 minutes)   177s (~2.95 minutes)   4.157s
```

History and provenance

```
drake_history()
## # A tibble: 10 x 10
##    target  current built  exists hash  command     seed runtime  prop act
##    <chr>   <lgl>   <chr>  <lgl>  <chr> <chr>      <int>   <dbl> <dbl> <ch
##  1 conf_r… TRUE    2019-… TRUE   880e… confusio… 4.05e8  0.224   NA   <NA
##  2 conf_s… TRUE    2019-… TRUE   9211… confusio… 1.93e9  0.238   NA   <NA
##  3 conf_s… TRUE    2019-… TRUE   793c… confusio… 1.80e9  0.228   NA   <NA
##  4 data    TRUE    2019-… TRUE   1347… "read_cs… 1.29e9  0.0410   0.3 <NA
##  5 metrics FALSE   2019-… TRUE   3b36… compare_… 1.21e9  0.0290  NA   <NA
##  6 metrics TRUE    2019-… TRUE   fd9a… compare_… 1.21e9  0.0270  NA   <NA
##  7 model_… TRUE    2019-… TRUE   d623… "train_m… 1.47e9  5.06    NA   relu
##  8 model_… TRUE    2019-… TRUE   5228… "train_m… 1.26e9  3.21    NA   sigm
##  9 model_… TRUE    2019-… TRUE   283a… "train_m… 8.05e8  3.62    NA   soft
## 10 rec     TRUE    2019-… TRUE   1478… prepare_… 6.29e8  0.181   NA   <NA
```

Reproducible data recovery

```
clean() # Oops!

start <- proc.time()
make(plan, recover = TRUE)
## recover data
## recover rec
## recover model_relu
## recover model_sigmoid
## recover model_softmax
## recover conf_relu
## recover conf_sigmoid
## recover conf_softmax
## recover metrics

proc.time() - start
##    user  system elapsed
##   0.088   0.036   0.232
```

- Details + how to rename a target: https://ropenscilabs.github.io/drake-manual/walkthrough.html#automated-recovery-and-renaming

High-performance computing

```
# 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`:

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

- Workshop materials:

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

- Example code from these slides:

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

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