Reproducible computation at scale in R

Will Landau

Copyright Eli Lilly and Company

Large statistical computation

- Bayesian data analysis
- Bayesian network meta-analysis
- Graph-based multiple comparison procedures
- Subgroup identification
- Predictive modeling
- Deep neural networks
- PK/PD modeling
- Clinical trial simulation
- Target identification

Copyright Eli Lilly and Company

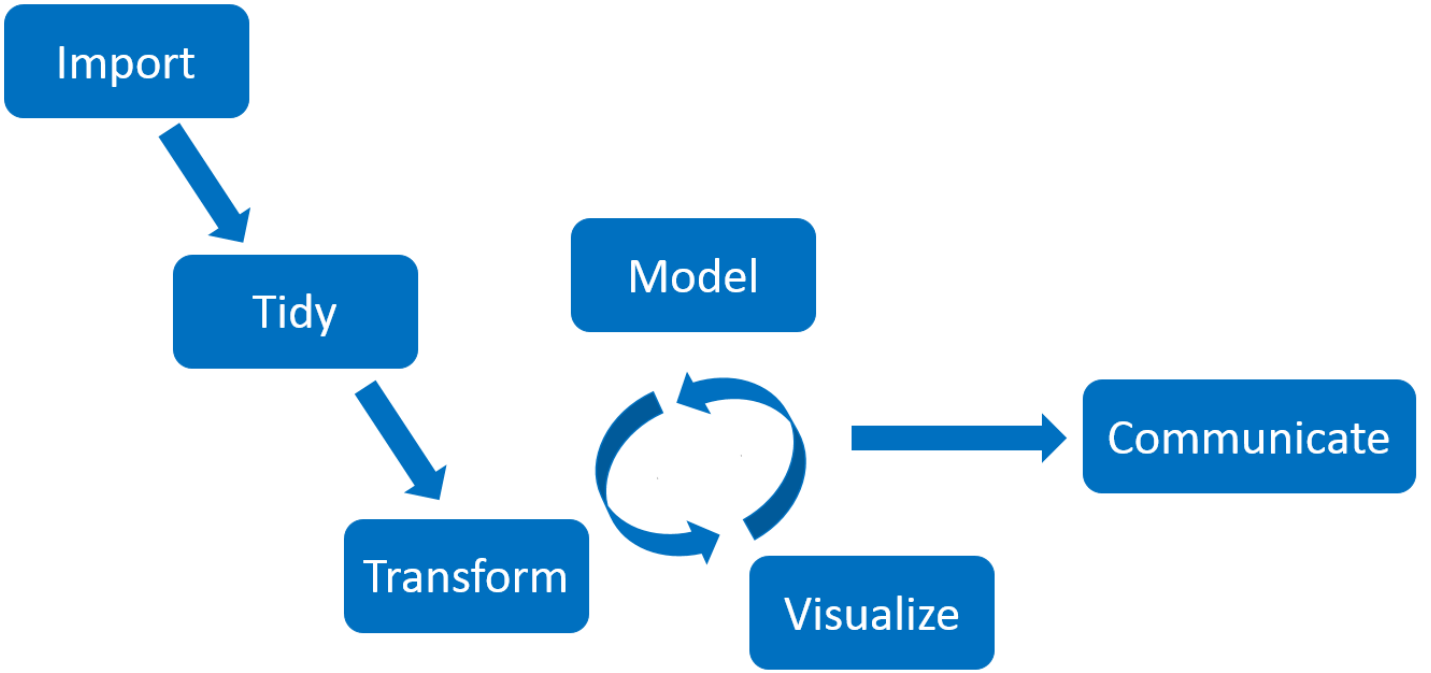
Common features

1. Heavy use of the R language.
2. Long runtimes.
3. Multiple sub-tasks.
4. Frequent changes to code and data.

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

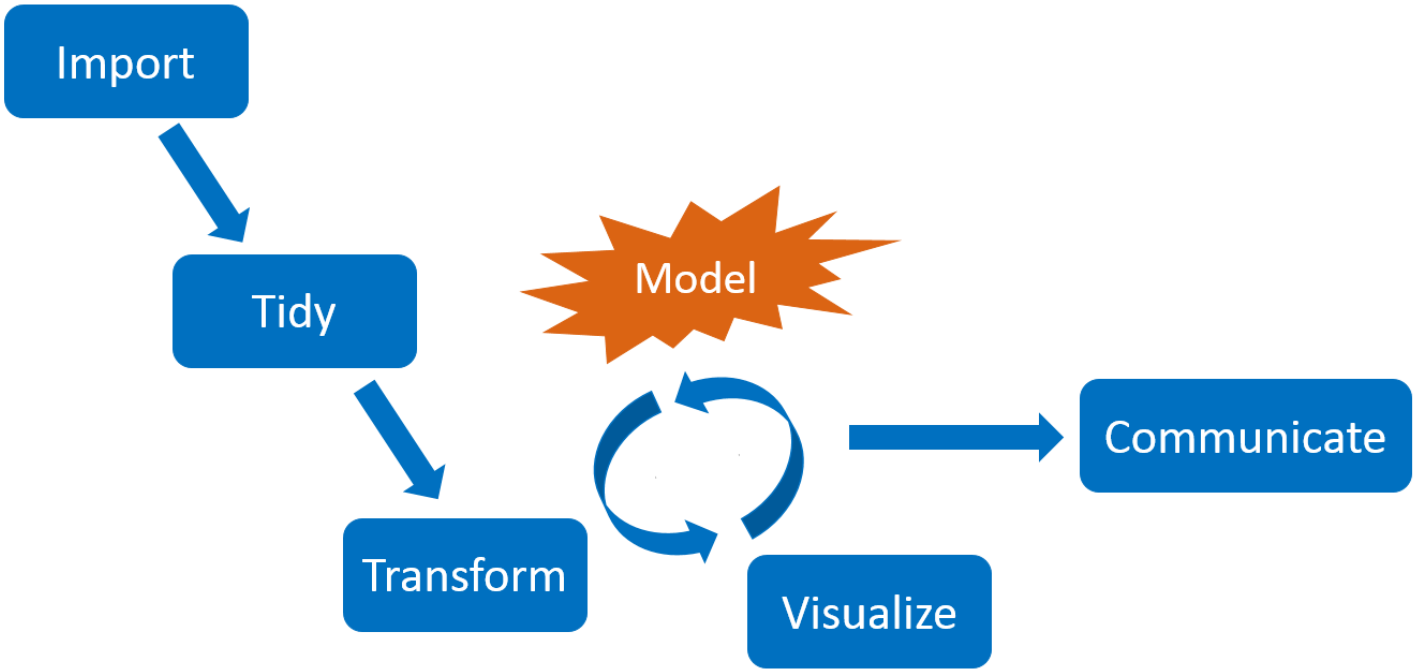
Copyright Eli Lilly and Company

Interconnected tasks



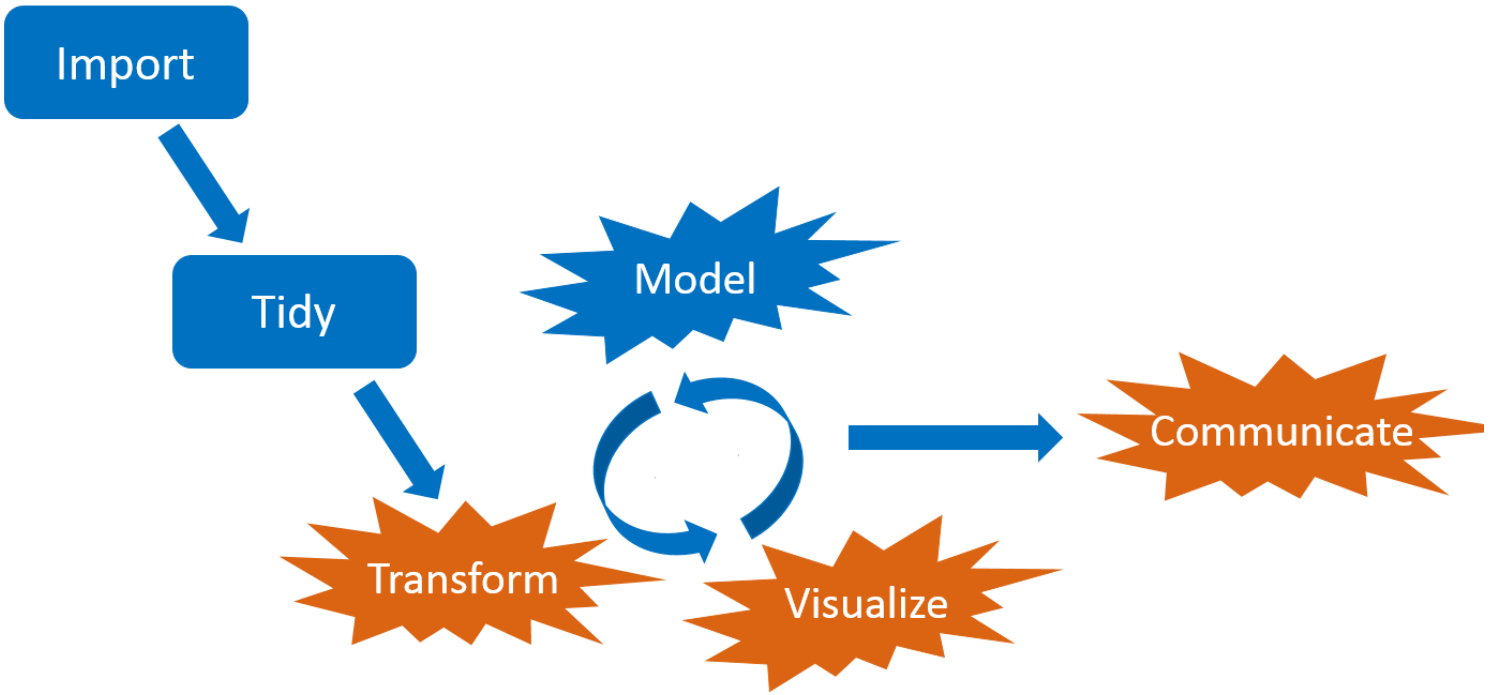
Copyright Eli Lilly and Company

Changes



Copyright Eli Lilly and Company

Consequences



Copyright Eli Lilly and Company

Pipeline tools and workflow managers

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.

Copyright Eli Lilly and Company

What distinguishes `drake`?

- Respects the way R works.
- Better code, i.e. functions.
- Time savings allow for an incremental development strategy:
    1. Change a couple things.
    2. Run the workflow.
    3. Inspect results.
    4. *Repeat often*.

Copyright Eli Lilly and Company

Example drake workflow

- Find the model that best predicts which customers will cancel their telecom subscriptions.
- IBM Watson Telco Customer Churn dataset.
- Workflow principles generalize to the life sciences.

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

Copyright Eli Lilly and Company

❌ Say goodbye to numbered imperative scripts!

```
run_everything.R
R/
├── 01-data.R
├── 02-munge.R
├── 03-model.R
├── 04-results.R
└── 05-plot.R
data/
└── customer_churn.csv
```

Copyright Eli Lilly and Company

✓ drake wants you to write **functions**.

> - Everything that exists is an object.
> - Everything that happens is a function call.
>
> John Chambers

```
add_things <- function(argument1, argument2) {
  argument1 + argument2
}

add_things(1, 2)
#> [1] 3

add_things(c(3, 4), c(5, 6))
#> [1]  8 10
```

Copyright Eli Lilly and Company

Functions in a drake workflow

```
split_data <- function(churn_file) {
  read_csv(churn_file, col_types = cols()) %>%
    initial_split(prop = 0.3)
}

prepare_recipe <- function(churn_data) {
  churn_data %>%
    training() %>%
    recipe(Churn ~ .) %>%
    step_rm(customerID) %>%
    step_naomit(all_outcomes(), all_predictors()) %>%
    step_discretize(tenure, options = list(cuts = 6)) %>%
    step_log(TotalCharges) %>%
    step_mutate(Churn = ifelse(Churn == "Yes", 1, 0)) %>%
    step_dummy(all_nominal(), -all_outcomes()) %>%
    step_center(all_predictors(), -all_outcomes()) %>%
    step_scale(all_predictors(), -all_outcomes()) %>%
    prep()
}
```

Copyright Eli Lilly and Company

Functions in a drake workflow

```
define_model <- function(churn_recipe, units1, units2, act1, act2, ac
  # ...
}

train_model <- function(churn_recipe, units1, units2, act1, act2, act
  # ...
}

test_accuracy <- function(churn_data, churn_recipe, model) {
  # ...
}

test_model <- function(churn_data, churn_recipe, units1, units2, acti
  # ...
}

train_best_model <- function(best_run, churn_recipe) {
  # ...
}
```

Copyright Eli Lilly and Company

Typical project structure

- There are *many* variations on this theme.

```
make.R # Top-level script
R/
├── packages.R # Calls to library(drake), library(keras), etc.
├── functions.R
└── plan.R
data/
└── customer_churn.csv
.drake/ # drake's cache
└──     # Output automatically appears here.
```

Copyright Eli Lilly and Company

Build up your workflow in a **drake plan**.

- Usually goes in the `R/plan.R` script.

```
# R/plan.R
plan <- drake_plan(
  churn_recipe = prepare_recipe(churn_data),
  churn_data = split_data(file_in("data/customer_churn.csv"))
)
```

Copyright Eli Lilly and Company

The plan is a data frame of skippable *targets*.

```
plan
#> # A tibble: 2 x 2
#>   target       command
#>   <chr>        <expr>
#> 1 churn_recipe prepare_recipe(churn_data)
#> 2 churn_data   split_data(file_in("data/customer_churn.csv"))
```

Copyright Eli Lilly and Company

drake understands code and data dependencies.

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

```
vis_drake_graph(plan)
```

Copyright Eli Lilly and Company

Build your first targets.

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

```
make(plan)
#> ▶ target churn_data
#> ▶ target churn_recipe
```

Copyright Eli Lilly and Company

Check the targets for problems

- `loadd()` and `readd()` get targets from the cache.

```
ncol(training(readd(churn_data)))
#> [1] 21

loadd(churn_recipe)
ncol(juice(churn_recipe))
#> [1] 36
```

Copyright Eli Lilly and Company

Build up the plan *gradually*.

1. Add a couple targets.
2. Build targets with `make(plan)`.
3. Inspect targets with `loadd()` and `readd()`.
4. *Repeat often*. Not as time-consuming as you might think!

Copyright Eli Lilly and Company

Add some model runs.

```
# R/plan.R
plan <- drake_plan(
  churn_recipe = prepare_recipe(churn_data),
  churn_data = split_data(file_in("data/customer_churn.csv")),
  run_relu = test_model(act1 = "relu", churn_data, churn_recipe),
  run_sigmoid = test_model(act1 = "sigmoid", churn_data, churn_recipe
)
```

Copyright Eli Lilly and Company

Previous work is still up to date.

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

```
outdated(plan)
#> [1] "run_relu"    "run_sigmoid"
```

Copyright Eli Lilly and Company

Previous work is still up to date.

```
vis_drake_graph(plan)
```

Copyright Eli Lilly and Company

drake skips up-to-date targets.

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

```
make(plan)
#> ▶ target run_relu
#> ▶ target run_sigmoid
```

Copyright Eli Lilly and Company

Inspect the newest targets.

```
readd(run_relu)
#> # A tibble: 1 x 6
#>   accuracy units1 units2 act1  act2  act3
#>      <dbl>  <dbl>  <dbl> <chr> <chr> <chr>
#> 1    0.803     16     16 relu  relu  sigmoid

readd(run_sigmoid)
#> # A tibble: 1 x 6
#>   accuracy units1 units2 act1    act2  act3
#>      <dbl>  <dbl>  <dbl> <chr>   <chr> <chr>
#> 1    0.799     16     16 sigmoid relu  sigmoid
```

Copyright Eli Lilly and Company

Find the best model

```
plan <- drake_plan(
  churn_recipe = prepare_recipe(churn_data),
  churn_data = split_data(file_in("data/customer_churn.csv")),
  run_relu = test_model(act1 = "relu", churn_data, churn_recipe),
  run_sigmoid = test_model(act1 = "sigmoid", churn_data, churn_recipe
  best_run = bind_rows(run_relu, run_sigmoid) %>%
    top_n(1, accuracy) %>%
    head(1),
  best_model = target(
    train_best_model(best_run, churn_recipe),
    format = "keras"
  )
)
```

Copyright Eli Lilly and Company

Find the best model

```
make(plan)
#> ▶ target best_run
#> ▶ target best_model
```

Copyright Eli Lilly and Company

Find the best model

```
readd(best_model)
#> Model
#> _________________________________________________________________
#> Layer (type)                 Output Shape              
#> =================================================================
#> dense_6 (Dense)              (None, 16)                
#> _________________________________________________________________
#> dropout_4 (Dropout)          (None, 16)                
#> _________________________________________________________________
#> dense_7 (Dense)              (None, 16)                
#> _________________________________________________________________
#> dropout_5 (Dropout)          (None, 16)                
#> _________________________________________________________________
#> dense_8 (Dense)              (None, 1)                 
#> =================================================================
#> Total params: 865
#> Trainable params: 865
#> Non-trainable params: 0
#> _________________________________________________________________
```

Copyright Eli Lilly and Company

Try another model.

```
# R/plan.R
plan <- drake_plan(
  churn_recipe = prepare_recipe(churn_data),
  churn_data = split_data(file_in("data/customer_churn.csv")),
  run_relu = test_model(act1 = "relu", churn_data, churn_recipe),
  run_sigmoid = test_model(act1 = "sigmoid", churn_data, churn_recipe
  run_softmax = test_model(act1 = "softmax", churn_data, churn_recipe
  best_run = bind_rows(run_relu, run_sigmoid, run_softmax) %>%
    top_n(1, accuracy) %>%
    head(1),
  best_model = target(
    train_best_model(best_run, churn_recipe),
    format = "keras"
  )
)
```

Copyright Eli Lilly and Company

What gets done stays done.

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

```
outdated(plan)
#> [1] "best_model"  "best_run"    "run_softmax"
```

Copyright Eli Lilly and Company

What gets done stays done.

```
vis_drake_graph(plan)
```

Copyright Eli Lilly and Company

New best model?

- Only if the new run beats the old runs.
- Otherwise, `drake` does not bother to retrain the best model.

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

```
make(plan)
#> target run_softmax
#> target best_run
```

Copyright Eli Lilly and Company

What if we need to change a function?

```
define_model <- function(churn_recipe, units1, units2, act1, act2, ac
  input_shape <- ncol(
    juice(churn_recipe, all_predictors(), composition = "matrix")
  )
  keras_model_sequential() %>%
    layer_dense(
      units = units1,
      kernel_initializer = "uniform",
      activation = act1,
      input_shape = input_shape
    ) %>%
    layer_dropout(rate = 0.2) %>% # previously 0.1
    layer_dense(
      units = units2,
      kernel_initializer = "uniform",
      activation = act2
    ) %>%
    layer_dropout(rate = 0.1) %>%
    layer_dense(
      units = 1,
      kernel_initializer = "uniform",
      activation = act3
    )
```

Copyright Eli Lilly and Company

drake knows what to do.

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

Copyright Eli Lilly and Company

drake knows what to do.

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

```
make(plan)
#> ▶ target run_relu
#> ▶ target run_softmax
#> ▶ target run_sigmoid
#> ▶ target best_run
#> ▶ target best_model
```

Copyright Eli Lilly and Company

Undo the change.

```
define_model <- function(churn_recipe, units1, units2, act1, act2, ac
  input_shape <- ncol(
    juice(churn_recipe, all_predictors(), composition = "matrix")
  )
  keras_model_sequential() %>%
    layer_dense(
      units = units1,
      kernel_initializer = "uniform",
      activation = act1,
      input_shape = input_shape
    ) %>%
    layer_dropout(rate = 0.1) %>% # Changed back to 0.1.
    layer_dense(
      units = units2,
      kernel_initializer = "uniform",
      activation = act2
    ) %>%
    layer_dropout(rate = 0.1) %>%
    layer_dense(
      units = 1,
      kernel_initializer = "uniform",
      activation = act3
    )
```

Copyright Eli Lilly and Company

Undo the change.

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

- Recover old targets instead of building new ones.

```
make(plan, recover = TRUE)
#> ✓ recover run_relu
#> ✓ recover run_softmax
#> ✓ recover run_sigmoid
#> ✓ recover best_run
#> ✓ recover best_model
```

Copyright Eli Lilly and Company

Similar story if the data file changes.

```
source("R/packages.R")
source("R/functions.R")
source("R/plan.R") # Requires file_in() in the plan.
```

```
make(plan)
#> target churn_data
#> target churn_recipe
#> target run_relu
#> target run_sigmoid
#> target run_softmax
#> target best_run
#> target best_model
```

Copyright Eli Lilly and Company

Evidence of reproducibility

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

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

outdated(plan)
#> character(0)
```

Copyright Eli Lilly and Company

Evidence of reproducibility

```
vis_drake_graph(plan)
```

Copyright Eli Lilly and Company

History of past model runs

```
history <- drake_history() %>%
  select(target, act1, exists, hash) %>%
  filter(!is.na(act1)) %>%
  print()
#> # A tibble: 6 x 4
#>   target      act1    exists hash
#>   <chr>       <chr>   <lgl>  <chr>
#> 1 run_relu    relu    TRUE   693f38d0cca6df48
#> 2 run_relu    relu    TRUE   d18e61338781810b
#> 3 run_sigmoid sigmoid TRUE   bfcea2797b1845c8
#> 4 run_sigmoid sigmoid TRUE   2b28e18f2a2a88bc
#> 5 run_softmax softmax TRUE   0fba209a1b82f8a1
#> 6 run_softmax softmax TRUE   907f0a76c2d5da5b
```

Copyright Eli Lilly and Company

Recover an old model run.

```
drake_cache()$get_value(history$hash[1])
#> # A tibble: 1 x 6
#>   accuracy units1 units2 act1  act2  act3
#>      <dbl>  <dbl>  <dbl> <chr> <chr> <chr>
#> 1    0.803     16     16 relu  relu  sigmoid
```

Copyright Eli Lilly and Company

More highlights

- High-performance computing
- Efficient data formats
- Dynamic branching
- Dynamic files
- Memory management

```
# Run targets in parallel on a cluster:
options(clustermq.scheduler = "slurm")
make(plan, parallelism = "clustermq", jobs = 64)
```

Copyright Eli Lilly and Company

Resources

- Get `drake`:

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

- Example code from these slides:

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

- Workshop materials:

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

Copyright Eli Lilly and Company

Links

- Development repository: https://github.com/ropensci/drake
- Full user manual https://books.ropensci.org/drake/
- Reference website: https://docs.ropensci.org/drake
- Hands-on workshop: https://github.com/wlandau/learndrake
- 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.

Copyright Eli Lilly and Company

Thanks

- Edgar Ruiz
- example code

- Matt Dancho
- blog post

Copyright Eli Lilly and Company

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

Copyright Eli Lilly and Company

The workshop

1. Sign up for a free account at https://rstudio.cloud.
2. Log into https://rstudio.cloud/project/627076.
3. Work through the R notebooks in order.

Topic	Notebook
Custom functions	`1-functions/1-functions.Rmd`
`drake` plans	`2-plans/2-plans.Rmd`
Changing workflows	`3-changes/3-changes.Rmd`
Static branching	`4-static/4-static.Rmd`
Dynamic branching	`5-dynamic/5-dynamic.Rmd`
Files and R Markdown	`6-files/6-files.Rmd`

Copyright Eli Lilly and Company