# tidyflow: A simplified workflow for doing machine learning with tidymodels

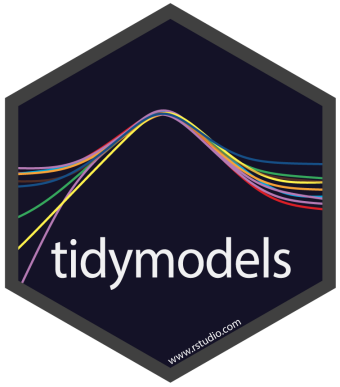
cimentadaj.github.io/tidyflow

@cimentadaj

24th of September, 2020

# tidymodels

- https://www.tidymodels.org/
- Rewrite of `caret`
- Maturing (started circa 2017)
- Created with a `'tidy'` philosophy in mind
- Decouples caret into many packages:
  - `recipes`
  - `parsnip`
  - `rsample`
  - `yardstick`
  - ...




I tried it out but it was too difficult for me

https://bit.ly/303EVuh

# tidymodels

```
library(AmesHousing)
library(tidymodels)

ames <- make_ames()

ames_split <- initial_split(ames, prop = .7)
ames_train <- training(ames_split)
ames_test <- testing(ames_split)
ames_cv <- vfold_cv(ames_train)

mod_rec <-
  recipe(Sale_Price ~ Longitude + Latitude + Neighborhood, data = ames_train) %>%
  step_log(Sale_Price, base = 10) %>%
  step_other(Neighborhood, threshold = 0.05) %>%
  step_dummy(all_nominal())

lm_mod <- linear_reg(penalty = tune(), mixture = tune()) %>% set_engine("glmnet")

ml_wflow <-
  workflow() %>%
  add_recipe(mod_rec) %>%
  add_model(lm_mod)

res <-
  ml_wflow %>%
  tune_grid(
    resamples = ames_cv,
    grid = 10,
    metrics = metric_set(rmse)
  )

best_params <- select_best(res, metric = "rmse", maximize = FALSE)

reg_res <-
  ml_wflow %>%
  finalize_workflow(best_params) %>%
  fit(data = ames_train)

reg_res %>%
  predict(new_data =  bake(mod_rec, ames_test)) %>%
  bind_cols(ames_test, .) %>%
  mutate(Sale_Price = log10(Sale_Price)) %>%
  select(Sale_Price, .pred) %>%
  rmse(Sale_Price, .pred)
```

- Data is repeated many times
- Different fit functions (`tune_*`, `fit`, `fit_resamples`, etc..)
- Non-linear workflow (workflow is defined after data, resampling, etc..)
- Too many objects to remember (predict by mistake on the test set, which `fit` function to use, etc...)

# tidyflow

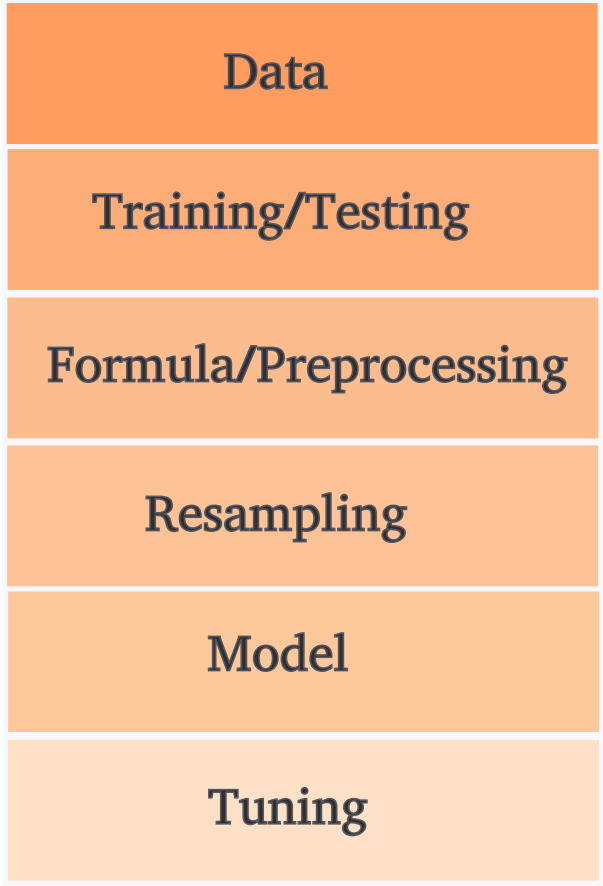
- https://cimentadaj.github.io/tidyflow/
- `tidyflow` is a fork of `workflows` (`tidymodels`)
- It aims to create a higher level extension to `tidymodels`
- Bundles your data, splitting, resampling, preprocessing, modeling, and grid search in a single object.

You can install the package from Github (and also `tidymodels`):

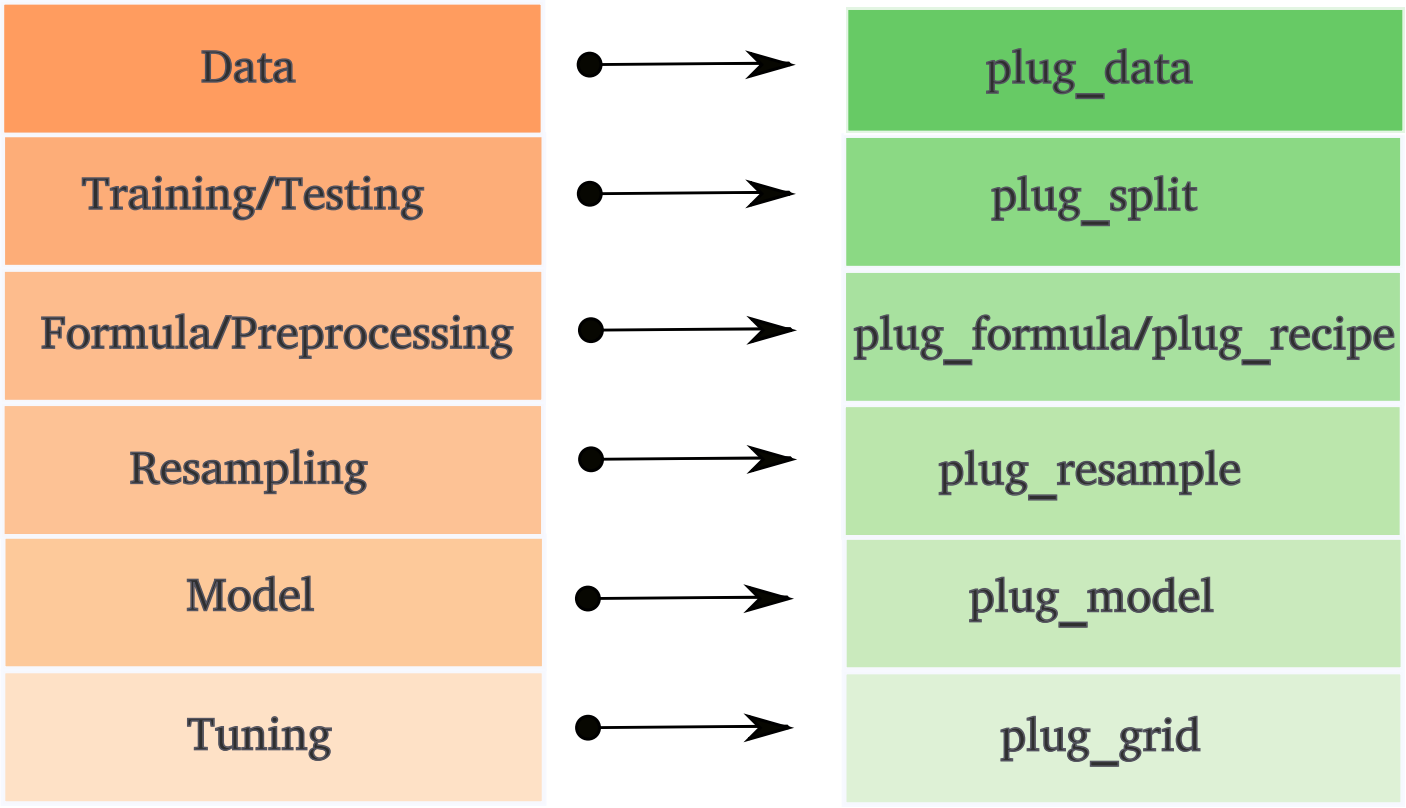
```
install.packages("tidymodels")

devtools::install_github("cimentadaj/tidyflow")
```

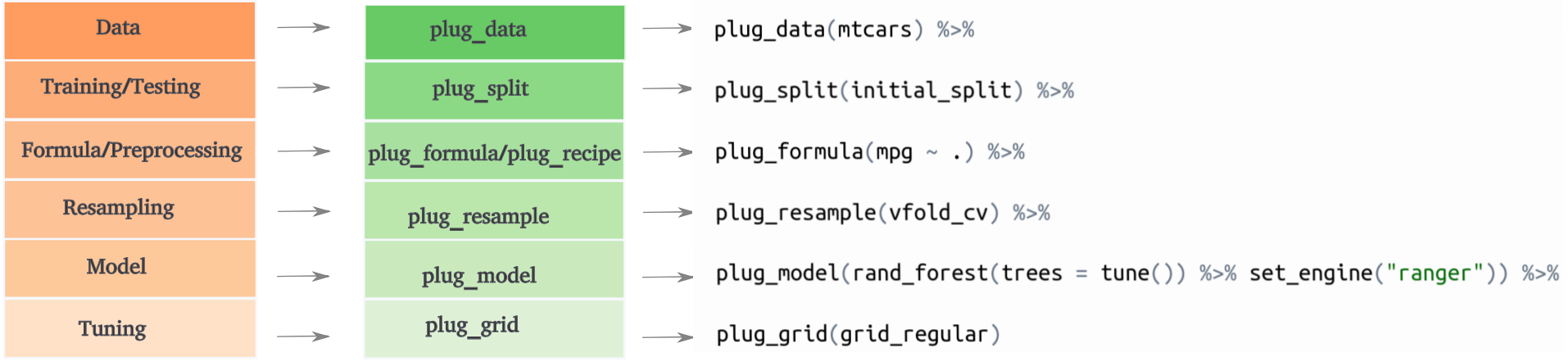
# tidyflow


Data
Training/Testing
Formula/Preprocessing
Resampling
Model
Tuning

# tidyflow


Data
Training/Testing
Formula/Preprocessing
Resampling
Model
Tuning
plug_data
plug_split
plug_formula/plug_recipe
plug_resample
plug_model
plug_grid

# tidyflow


Data
Training/Testing
Formula/Preprocessing
Resampling
Model
Tuning
plug_data
plug_split
plug_formula/plug_recipe
plug_resample
plug_model
plug_grid
tidyflow(seed = 51231) %>%
plug_data(mtcars) %>%
plug_split(initial_split) %>%
plug_formula(mpg ~ .) %>%
plug_resample(vfold_cv) %>%
plug_model(rand_forest(trees = tune()) %>% set_engine("ranger")) %>%
plug_grid(grid_regular)

# tidyflow

```
library(tidymodels)
library(tidyflow)

tflow <-
  mtcars %>%
  tidyflow(seed = 5213) %>%
  plug_split(initial_split) %>%
  plug_formula(mpg ~ .) %>%
  plug_model(linear_reg() %>% set_engine("lm"))

tflow
```

```
## ══ Tidyflow ════════════════════════════════════════════════════════════════
## Data: 32 rows x 11 columns
## Split: initial_split w/ default args
## Formula: mpg ~ .
## Resample: None
## Grid: None
## Model:
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

# tidyflow

```
res <- fit(tflow)
res
```

```
## == Tidyflow [trained] ======================================================
## Data: 32 rows x 11 columns
## Split: initial_split w/ default args
## Formula: mpg ~ .
## Resample: None
## Grid: None
## Model:
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
##
## == Results =================================================================
##
##
## Fitted model:
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
##
```

# tidyflow

`tidyflow` leverages the power of `tidymodels` so we can continue to use the same infrastructure:

- `plug_split`: a function to be applied to the data such as `initial_split`, etc...
- `plug_formula`: the formula of the model definition. A tidyflow needs to specify either a formula or a recipe, but not both.
- `plug_recipe`: a formula containing a recipe that will be applied to the training data.
- `plug_resample`: a function to be applied to the preprocessed data such as `vfold_cv`, etc...
- `plug_grid`: a function to be applied to the tuning placeholders in the recipe or the data such as `grid_regular`, etc...
- `plug_model`: a model object such as `rand_forest`, etc...

# tidyflow

Let's work out a more complicated example based on the initial `tidymodels` example. Let's begin a reproducible tidyflow with the data, the split and the seed:

```
library(AmesHousing)
library(tidymodels)
library(tidyflow)

ames <- make_ames()
tflow <- ames %>% tidyflow(seed = 52131) %>% plug_split(initial_split)
tflow
```

```
## ══ Tidyflow ════════════════════════════════════════════════════════════════
## Data: 2.93K rows x 81 columns
## Split: initial_split w/ default args
## Recipe/Formula: None
## Resample: None
## Grid: None
## Model: None
```

The `tidyflow` is currently only holding the data and the split.

# tidyflow

The `tidyflow` currently knows that it has a data frame and it will work solely once the training data set. Let's add a few transformations to the data with a recipe:

```
mod_rec <-
  ~ recipe(Sale_Price ~ Longitude + Latitude + Neighborhood, data = .x) %>%
    step_other(Neighborhood, threshold = 0.05) %>%
    step_dummy(all_nominal())

tflow <- tflow %>% plug_recipe(mod_rec)
tflow
```

```
## ══ Tidyflow ════════════════════════════════════════════════════════════════
## Data: 2.93K rows x 81 columns
## Split: initial_split w/ default args
## Recipe: available
## Resample: None
## Grid: None
## Model: None
```

# tidyflow

Let's run a regularized regression where we grid search through the `penalty` and `mixture` hyper-parameters:

```
reg_mod <- linear_reg(penalty = tune(), mixture = tune()) %>% set_engine("glmnet")

tflow <- tflow %>%
  plug_resample(vfold_cv) %>%
  plug_model(reg_mod) %>%
  plug_grid(grid_regular, levels = 5)

tflow
```

```
## ══ Tidyflow ════════════════════════════════════════════════════════════════
## Data: 2.93K rows x 81 columns
## Split: initial_split w/ default args
## Recipe: available
## Resample: vfold_cv w/ default args
## Grid: grid_regular w/ levels = ~5
## Model:
## Linear Regression Model Specification (regression)
##
## Main Arguments:
##   penalty = tune()
##   mixture = tune()
##
## Computational engine: glmnet
```

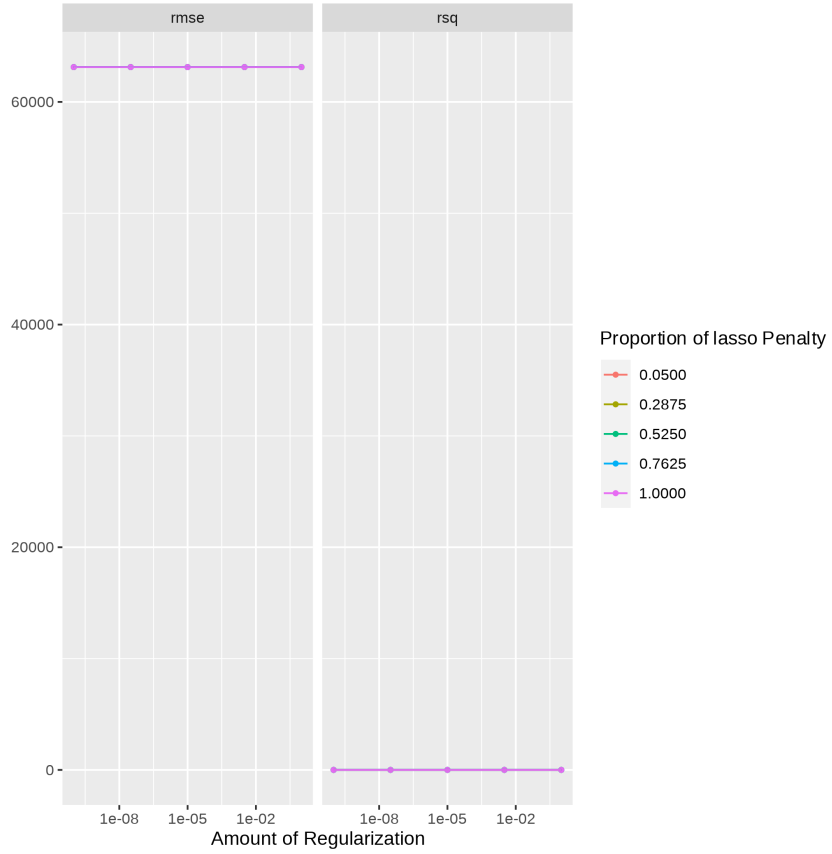
# tidyflow

```
res <- fit(tflow)
```

```
## # Tuning results
## # 10-fold cross-validation
## # A tibble: 10 x 4
##    splits           id     .metrics          .notes
##    <list>           <chr>  <list>            <list>
##  1 <split [2K/220]> Fold01 <tibble [50 × 6]> <tibble [0 × 1]>
##  2 <split [2K/220]> Fold02 <tibble [50 × 6]> <tibble [0 × 1]>
##  3 <split [2K/220]> Fold03 <tibble [50 × 6]> <tibble [0 × 1]>
##  4 <split [2K/220]> Fold04 <tibble [50 × 6]> <tibble [0 × 1]>
##  5 <split [2K/220]> Fold05 <tibble [50 × 6]> <tibble [0 × 1]>
##  6 <split [2K/220]> Fold06 <tibble [50 × 6]> <tibble [0 × 1]>
##  7 <split [2K/220]> Fold07 <tibble [50 × 6]> <tibble [0 × 1]>
##  8 <split [2K/220]> Fold08 <tibble [50 × 6]> <tibble [0 × 1]>
##  9 <split [2K/219]> Fold09 <tibble [50 × 6]> <tibble [0 × 1]>
## 10 <split [2K/219]> Fold10 <tibble [50 × 6]> <tibble [0 × 1]>
```

# tidyflow

```
res %>% pull_tflow_fit_tuning() %>% autoplot() + facet_wrap(~ .metric, ncol = 2)
```



# tidyflow

We can allow `tidyflow` to find the best combination of parameters and quickly look at our metric of interest.

```
final_mod <- res %>% complete_tflow(metric = "rmse")
multi_metric <- metric_set(rsq, rmse)

final_mod %>%
  predict_testing() %>%
  multi_metric(Sale_Price, .pred)
```

```
## # A tibble: 2 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.318
## 2 rmse    standard   65599.
```

# tidyflow

Once you've fitted your `tidyflow`, you can begin extracting many of the separate parts:

- `pull_tflow_rawdata()`
- `pull_tflow_split()`
- `pull_tflow_training()`
- `pull_tflow_testing()`
- `pull_tflow_resample()`
- `pull_tflow_spec()`
- `pull_tflow_fit()`
- `pull_tflow_fit_tuning()`
- ...

# tidyflow

Resources:

- Several vignettes showcasing detailed functionalities: https://cimentadaj.github.io/tidyflow/
- My course notes 'Machine Learning for Social Scientists': https://cimentadaj.github.io/ml_socsci/
- Source code. Looking for collaborations, features, bugs or new ideas: https://github.com/cimentadaj/tidyflow

Road map:

- Grid of models
- `plug_metric` for evaluating custom metrics
- Including custom options for additional `tune_*` executions (`tune_bayes`, etc...)

> Thanks to RStudio for supporting open source work and the `tidymodels` team for such a fresh infrastructure for doing tidy machine learning in R.