Modeling in the Tidyverse

Max Kuhn (RStudio)

Goals of Tidy Modeling

The tidy modeling packages are a set of coordinated packages that:

- Promote tenets of the tidyverse (manifesto here):

    1. Reuse existing data structures.
    2. Compose simple functions with the pipe.
    3. Embrace functional programming.
    4. Design for humans.

- Encourage empirical validation and good methodology

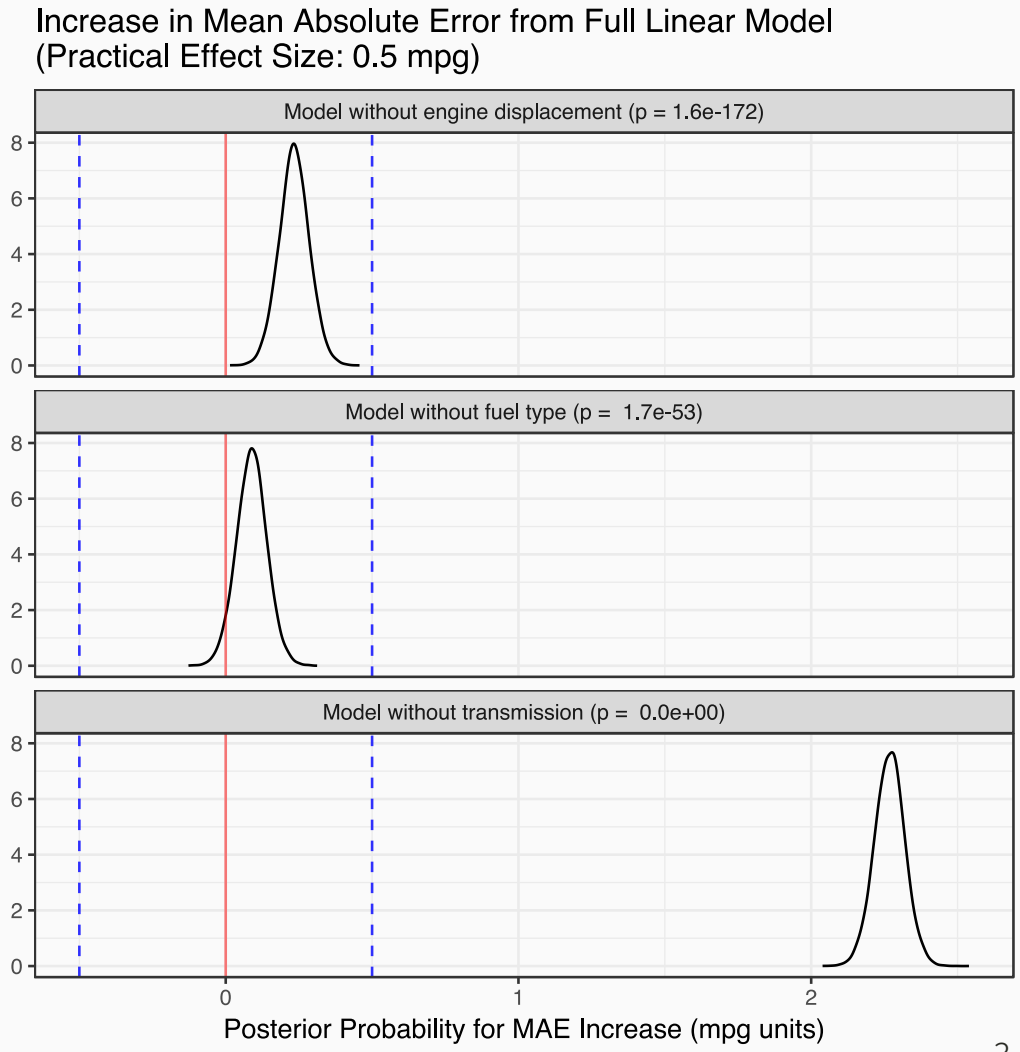
- Smooth out diverse interfaces

- Enable a wider variety of methodologies (esp. for feature engineering)

Empirical Validation and Good Methodology

For example:

- Embrace resampling to protect against poor methodology (e.g. classical stepwise, enhanced interrogation of data)
- Use loss functions that are relevant (e.g. expected return on investment vs accuracy)
- Don't solely rely on p-values to compare and characterize models
- Utilize Bayesian ROPE estimates to assess *practical differences* (example on right is based on an updated version of `mtcars` modeled using ordinary `lm`)



Smooth Out Diverse Interfaces

For example, to produce class probabilities:

Function	Package	Code
`lda`	`MASS`	`predict(obj)`
`glm`	`stats`	`predict(obj, type = "response")`
`gbm`	`gbm`	`predict(obj, type = "response", n.trees)`
`mda`	`mda`	`predict(obj, type = "posterior")`
`rpart`	`rpart`	`predict(obj, type = "prob")`
`Weka`	`RWeka`	`predict(obj, type = "probability")`

- `caret` does this for classification and regression
- Extend this to nearly all type of models
- Exploit delayed evaluation of expressions to produce a cleaner interface
- View R as the *primary* computational engine but offer other options

For example: `rand_forest` as an interface to `randomForest`, `ranger`, `sparklyr::ml_random_forest`, etc.

Possible Syntax

A *pipeline* consists of a set of actions such as :

- generic model specification (`parsnip` package)
- declaration of variables (formulas, `recipes`)

Optionally, things like:

- pre-processing methods (`recipes`)
- simple univariate filters (package TBA)
- calibration/post-fit adjustments (package TBA)

Aspects of these components *do not* have to be immediately defined. For example:

```
# Define the model matrix
vars_and_preproc <- recipe(response ~ ., data = dat) %>%
  step_knnimpute(all_predictors(), K = varying())

# Choose a model such as random forest...
model_spec <- rand_forest(
  trees = 1000,
  min_n = varying(),
  mtry = varying()
)

# ... or another types of model
model_spec <- surv_reg(distribution = varying())

# Optionally layer in some pre-model feature selection
filter <- feature_filter(all_predictors())

# Combine them together
model_spec <- pipeline() %>%
  add(vars_and_preproc) %>%
  add(model_spec) %>%
  add(filter)

# `pipeline` detects what what arguments are varying (if any)
```

Possible Syntax

At some point though, the pipeline needs to be finalize so that it can be estimated:

```
model_fit <- fit(data = train_dat, model_spec, engine = "R") # or stan or spark etc.
```

However, if there are still placeholders for parameters, there will be methods for tuning these values:

```
grid <- random_grid(model_spec, size = 20,
                    data = train_dat)

folds <- rsample::vfold_cv(train_dat)

grid_results <-
  grid_search(
    model_spec,
    values = grid,
    sampling = folds
  )
```

```
final_param <- grid_results %>% pick_best()
# or
final_param <- genetic_opt(model_spec, iter = 20,
                           sampling = folds)

fitted_model <- model_spec %>%
  update(final_param) %>%
  fit(data = train_dat)
```

Future Plans

Once the model interface is finalized, a set of packages will quickly follow.

Components will be released in packages that are relatively small in scope.

The plan is to offer both high- and low-level APIs for these tasks.

- `caret` is popular partly because it can make a lot of decisions for you. Obviously this is good and bad.
- For example, it isn't too difficult to do simple grid search using `rsample`, `recipes`, and `purrr` (see the workshop notes).

Thanks

- Hadley and our tidyverse team for the support and help
- David Robinson for `broom`
- All the contributors to the current set of packages