Max Kuhn (RStudio)

- Example Data
- Regularized Linear Models
- Multivariate Adaptive Regression Splines
- Ensembles of MARS Models
- Model Comparison via Bayesian Analysis


tidymodels

The data that are used here are an extended version of the ubiquitous data set.

was used to obtain fuel efficiency data on cars from 2015-2019.

Over this time range, duplicate ratings were eliminated; these occur when the same car is sold for several years in a row. As a result, there are 4209 cars that are listed in the data. The predictors include the automaker and addition information about the cars (e.g. intake valves per cycle, aspiration method, etc).

In our analysis, the data from 2015-2018 are used for training to see if we can predict the 613 cars that were new in 2018 or 2019.

Non-combustion engines have their MPG estimated even though there are no real gallons. Let's get rid of these for simplicity.

As before, let's take 10 minutes to get familiar with the training set using numerical summaries and plots.

More information about the data can be found on the government website and in this repo.

is very helpful here to discover the nature of relationships between the outcome ( ) and the potential predictors.

: If you had two models with competing RMSE values (in mpg), how big of a difference would be big enough to be ?

We'll start by fitting linear regression models to these data.

As a reminder, the "linear" part means that the model is linear in the ; we can add nonlinear terms to the model (e.g. or) without causing issues.

We could start by using and the formula method using what we've learned so far:

If one of these low occurrence car makes is only in the assessment set, it may cause issues (or errors) in some models.

A basic recipe will be created to account for this that collapses these data into an "other" category and will create dummy variables.

With recipes, variables can have different roles, such as case-weights, cluster, censoring indicator, etc.

We should keep in the data so that we can use it to diagnose issues but we don't want it as a predictor.

The role of this variable will be changed to .

We'll look at the car data and examine a few different models to illustrate some more complex models and approaches to optimizing them. We'll start with linear models.

However, some potential issues with linear methods:

- They do not automatically do and including irrelevant predictors may degrade performance.
- Linear models are sensitive to situations where the predictors are (aka collinearity). This isn't too big of an issue for these data though.

To mitigate these two scenarios, will be used. This approach adds a penalty to the regression parameters.

- In order to have a large slope in the model, the predictor will need to have a large impact on the model.

There are different types of regularization methods.

As an example of collinearity, our data set has two predictors that have a correlation above 0.95: and .

What happens when we fit models with both predictors versus one-at-a-time?

2 Door Lugg Vol	-0.173	---	0.125	14.1
2 Door Pass Vol	---	-0.023	-0.038	14.1

The coefficients can drastically change depending on what is in the model and their corresponding variances can also be artificially large.

Now suppose we want to see if the regression coefficients will result in better fits.

The model can be used to build a linear model using L_1 or L_2 regularization (or a mixture of the two).

- The general formulation minimizes: .
- An L_1 penalty (penalty is) can have the effect of setting coefficients to zero.
- L_2 regularization ( ) is basically ridge regression where the magnitude of the coefficients are dampened to avoid overfitting.

For a model, we need to determine the total amount regularization (called) and the mixture of L_1 and L_2 (called).

- = 1 is a while = 0 is (aka weight decay).

Predictors require centering/scaling before being used in a , lasso, or ridge regression model.

Technical bits can be found in Statistical Learning with Sparsity.

We have two tuning parameters now (and). We can extend our previous grid search approach by creating a 2D grid of parameters to test.

- must be between zero and one. A small grid is used for this parameter.
- is not as clear-cut. We consider values on the $\log_{10}$ scale. Usually values less than one are sufficient but this is not always true.

We can create combinations of these parameters and store them in a data frame:

Instead of using to tune the model, the function in the package will be introduced.

was developed to:

- create a unified interface for modeling and prediction to 238 models
- streamline model tuning using resampling
- provide a variety of "helper" functions and classes for day-to-day model building tasks
- increase computational efficiency using parallel processing
- enable several feature selection frameworks

It was originally developed in 2005 and is still very active.

There is an extensive page and an article in JSS.

can take a formula method, a recipe object, or a non-formula approach (/) to specify the model.

Another argument, , is used to specify the type of model to fit.

This is named after the fitting function.

We will need to use for that model. has a list of all possibilities.

of listing the submodels that should be evaluated is to used the parameter:

Alternatively, the argument will let determine a grid sequence for each parameter.

How much (and how) should we resample these data to create the model?

Previously, cross-validation was discussed. If 10-fold CV was used, the assessment set would consist of about 352 cars (on average).

That seems like an acceptable amount of data to determine the RMSE for each submodel.

has a control function that can be used to define parameters related to the numerical aspects of the search:

Let's add some nonlinearity and centering/scaling to the preprocessing and run the search:

Our grid contained 5 values for and 20 values of .

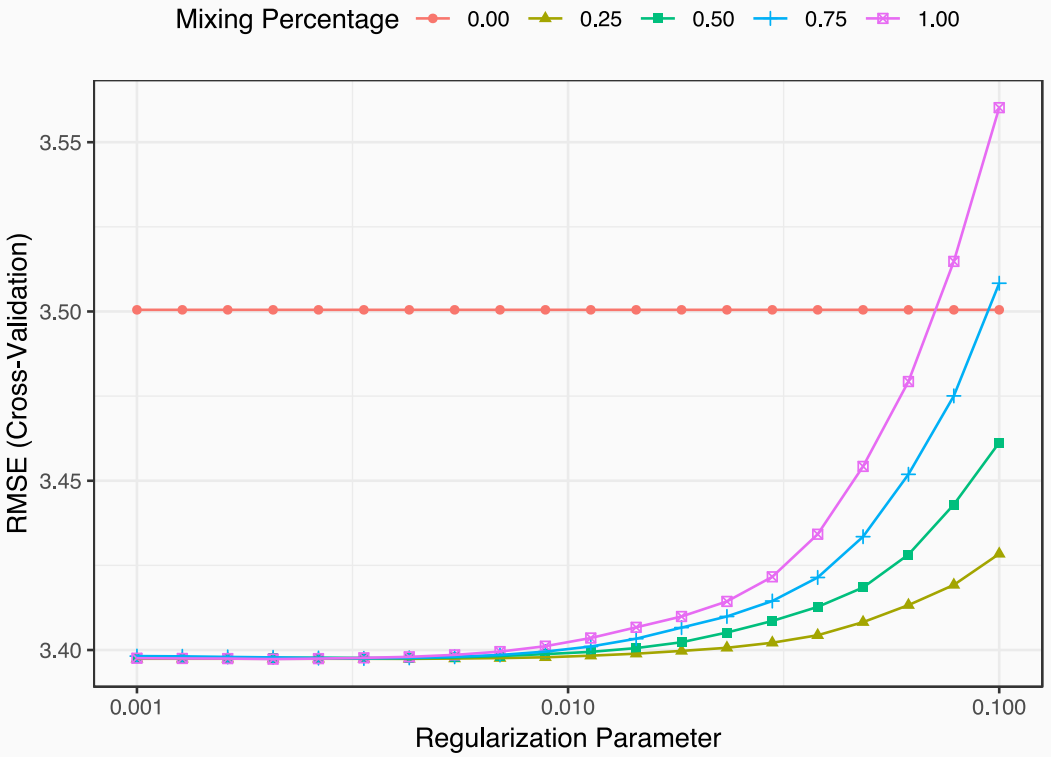
Although it might seem like we are fitting 100 models resample, we are not.

For many models, including , there are some computational shortcuts that can be used.

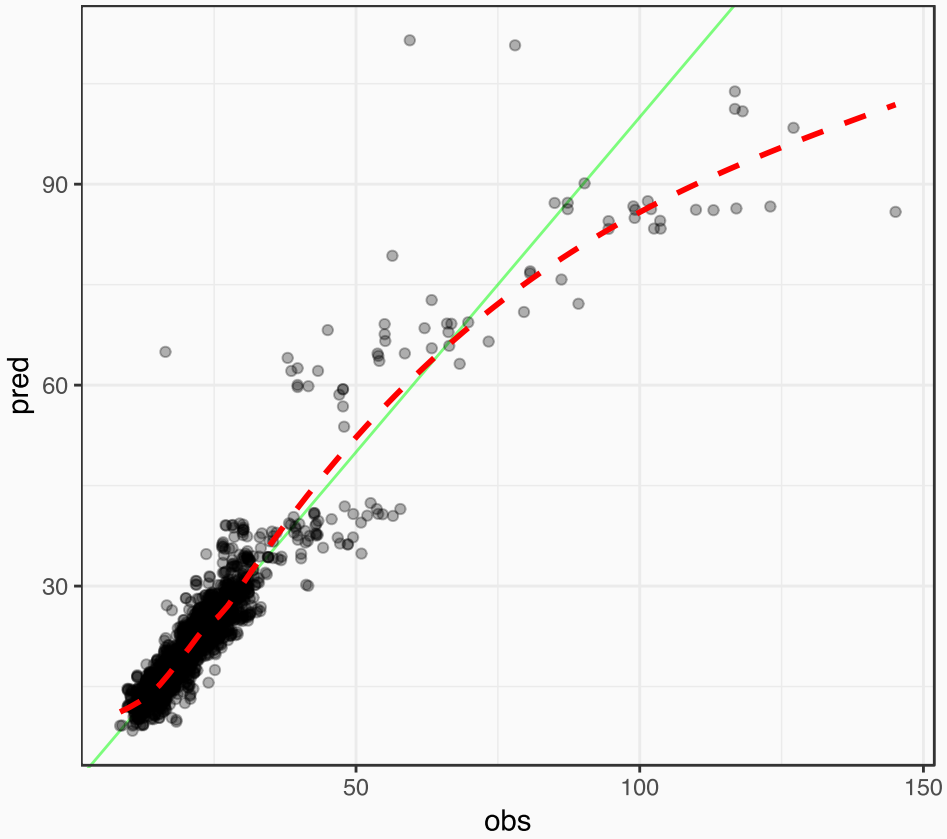
In this case, for a fixed value of , the model computes the results for all possible values of . Predictions from any of these models can be obtained from the same object.

This means that we only need to fit 5 models per resample.

Trees and other models can often exploit this and automatically does this whenever possible.


Mixing Percentage
0.00
0.25
0.50
0.75
1.00
RMSE (Cross-Validation)
3.55
3.50
3.45
3.40
0.001
0.010
0.100
Regularization Parameter

Since we used , the predictions on the assessment sets are contained in the sub-object . This can be used to plot the data:



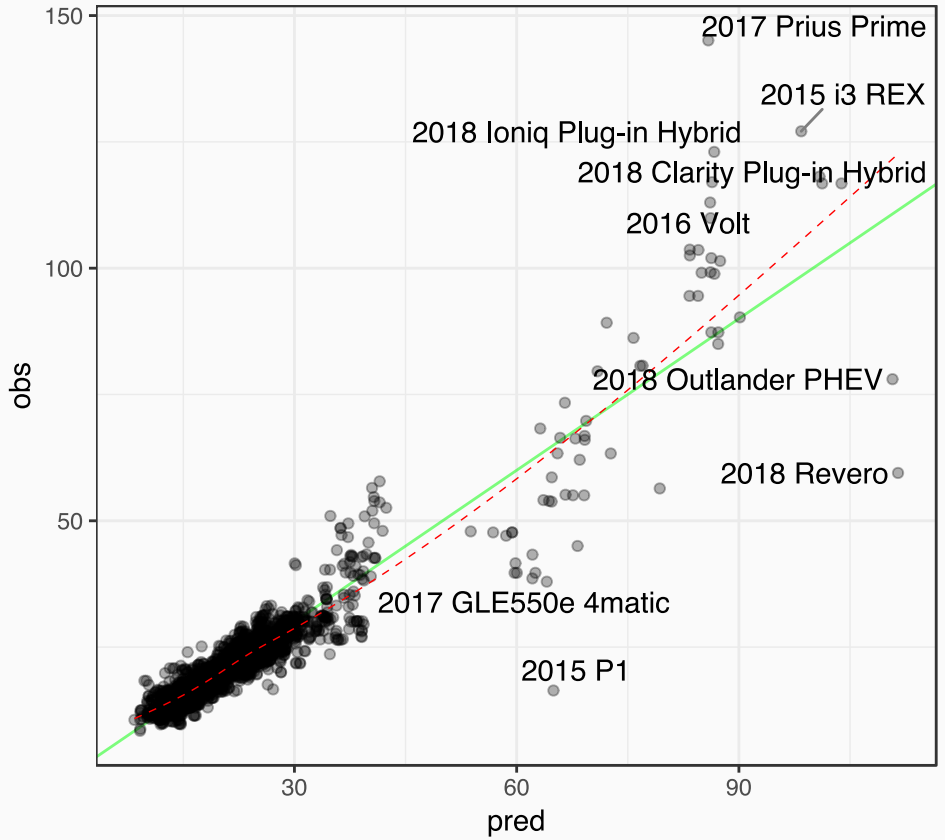
The object doesn't contain any information about the cars. Let's add some columns and also use the package to identify locations of poor fit.

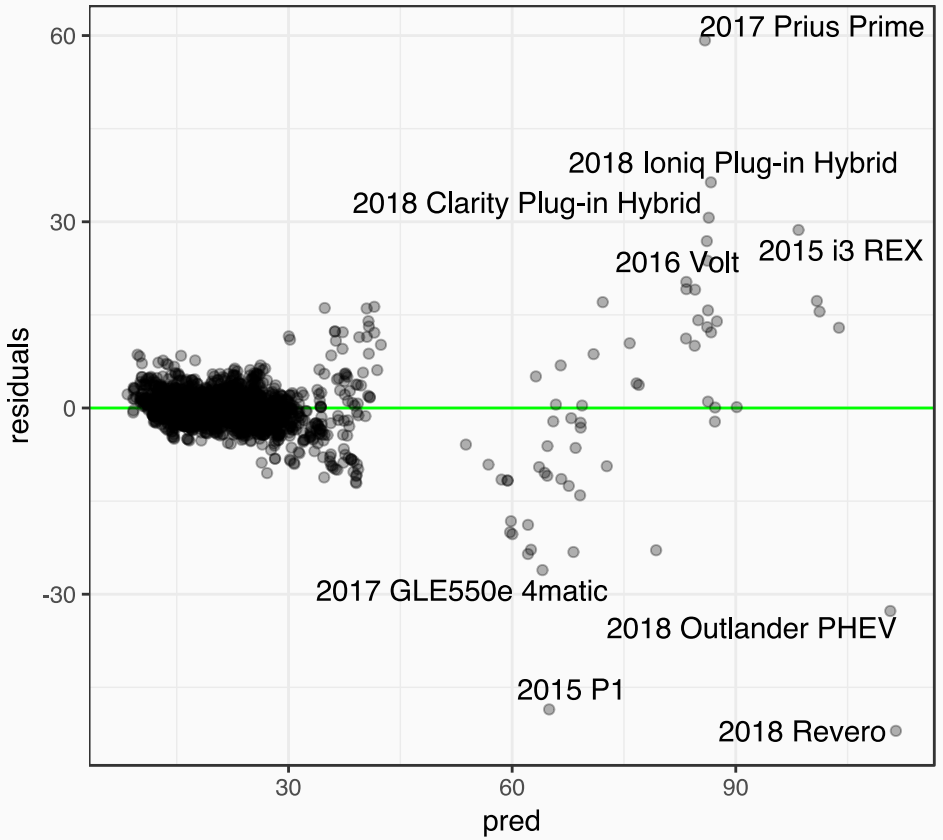
The function will add columns to the data in the object.

can be used with a subset of the data to locate large residuals.

The and packages are also good options for interactivity.


dplyr
ggplot2
ggrepel


2017 Prius Prime
2015 i3 REX
2018 Ioniq Plug-in Hybrid
2018 Clarity Plug-in Hybrid
2016 Volt
2018 Outlander PHEV
2018 Revero
2017 GLE550e 4matic
2015 P1
obs
pred
150
100
50
30
60
90

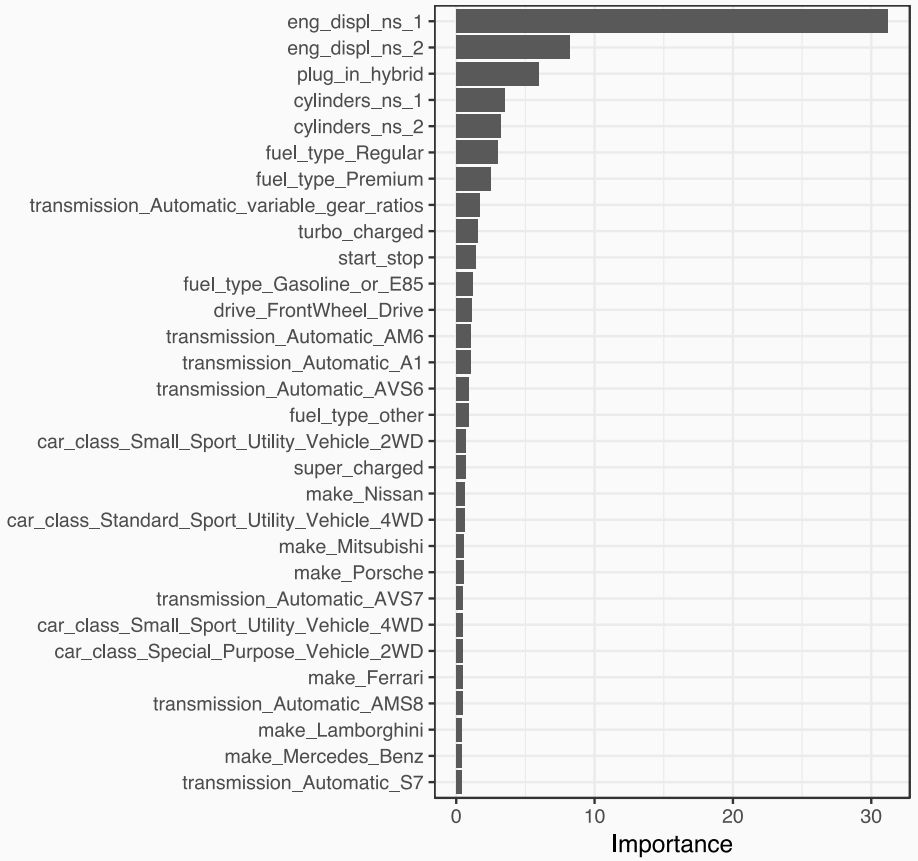

2017 Prius Prime
2018 Ioniq Plug-in Hybrid
2018 Clarity Plug-in Hybrid
2016 Volt
2015 i3 REX
2017 GLE550e 4matic
2018 Outlander PHEV
2015 P1
2018 Revero
residuals
pred
60
30
0
-30
30
60
90

For a linear model such as , we can directly inspect and interpret the model coefficients to understand what is going on.

A more general approach of computing "variable importance" scores can be useful for assessing which predictors are driving the model. These are model-specific.

For this model, we can plot the absolute values of the coefficients.

This is another good reason to center and scale the predictors before the model.



- Setting the seed just before calling will ensure that the same resamples are used between models. There is also a help page on reproducibility.
- calls the underlying model (e.g.) and arguments can be passed to the lower level functions via the .
- You can write your own model code (or examine what uses) with .
- If the formula method is used, dummy variables will be generated for the model.
- If you don't like the settings that are chosen, can be used to change them without repeating all of the resampling.

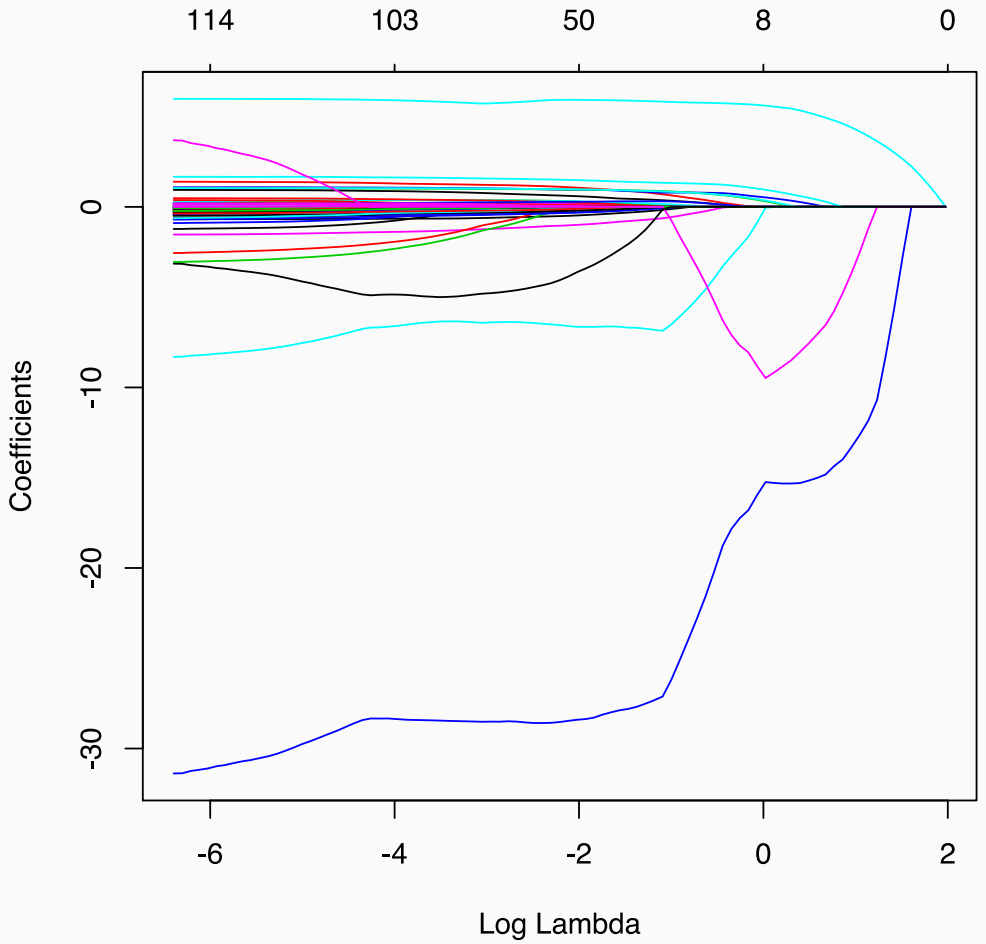
The object saves the optimized model that was fit to the entire training set in the slot .

This can be used as it normally would.

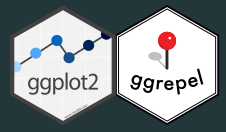
The plot on the right is creating using

However, !

Use the method on the object that is produced by .




broom
dplyr
ggplot2
ggrepel


ggplot2
ggrepel

MARS is a nonlinear machine learning model that develops sequential sets of artificial features that are used in linear models (similar to the previous spline discussion).

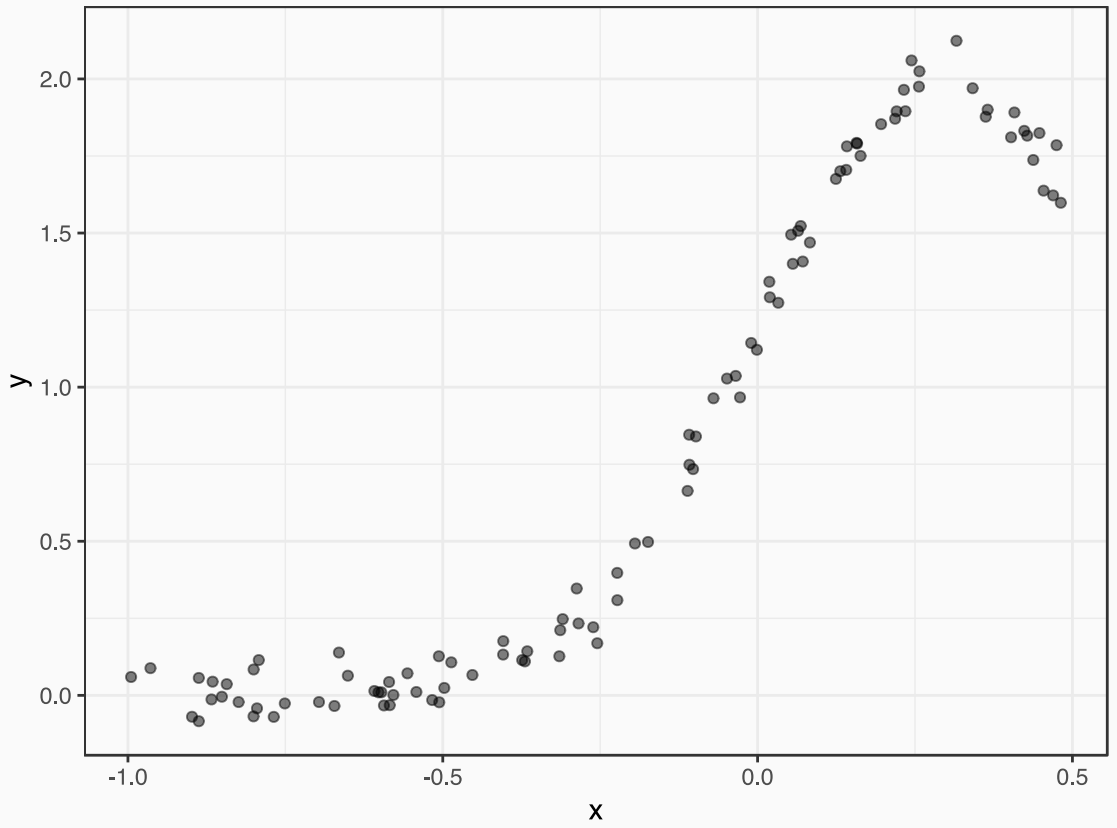
The features are "hinge functions" or single knot splines that use the function:

The MARS model does a fast search through every predictor and every value of each predictor to find a suitable "split" point for the predictor that results in the best features.

Suppose a value is found. The MARS model creates two model terms and that are added to the intercept column. This creates a type of .

These terms are the same as deep learning rectified linear units (ReLU).

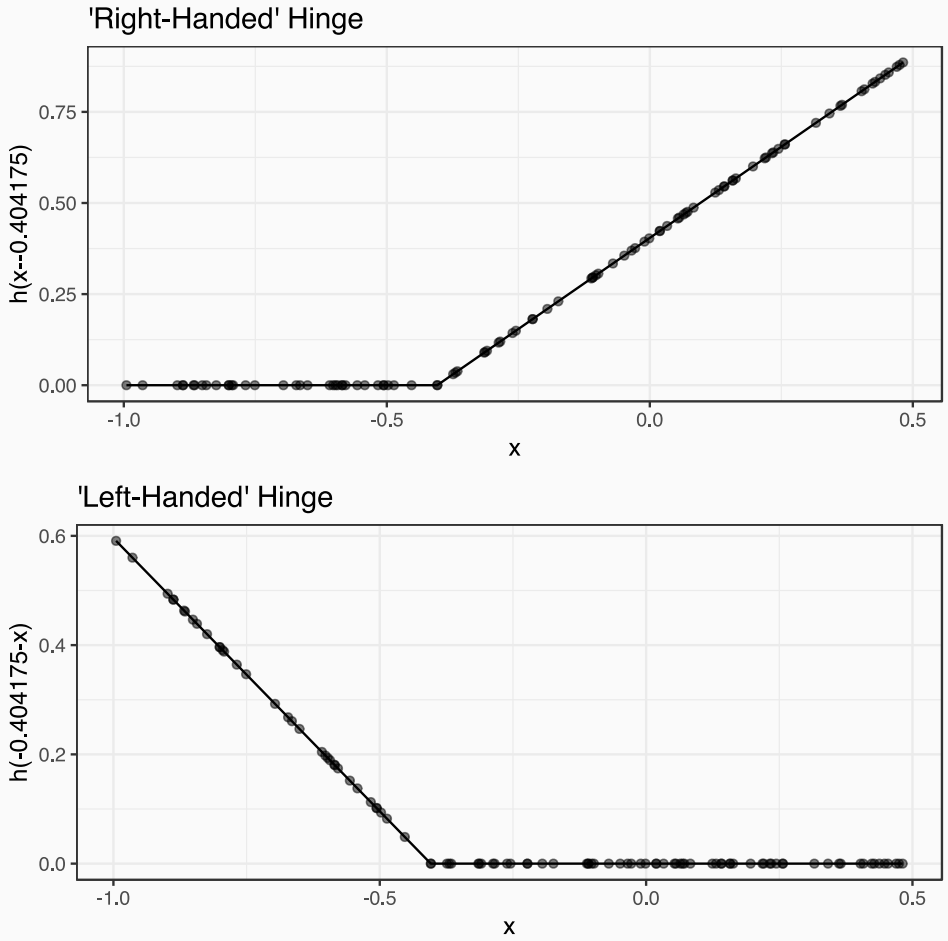
Let's look at some example data...

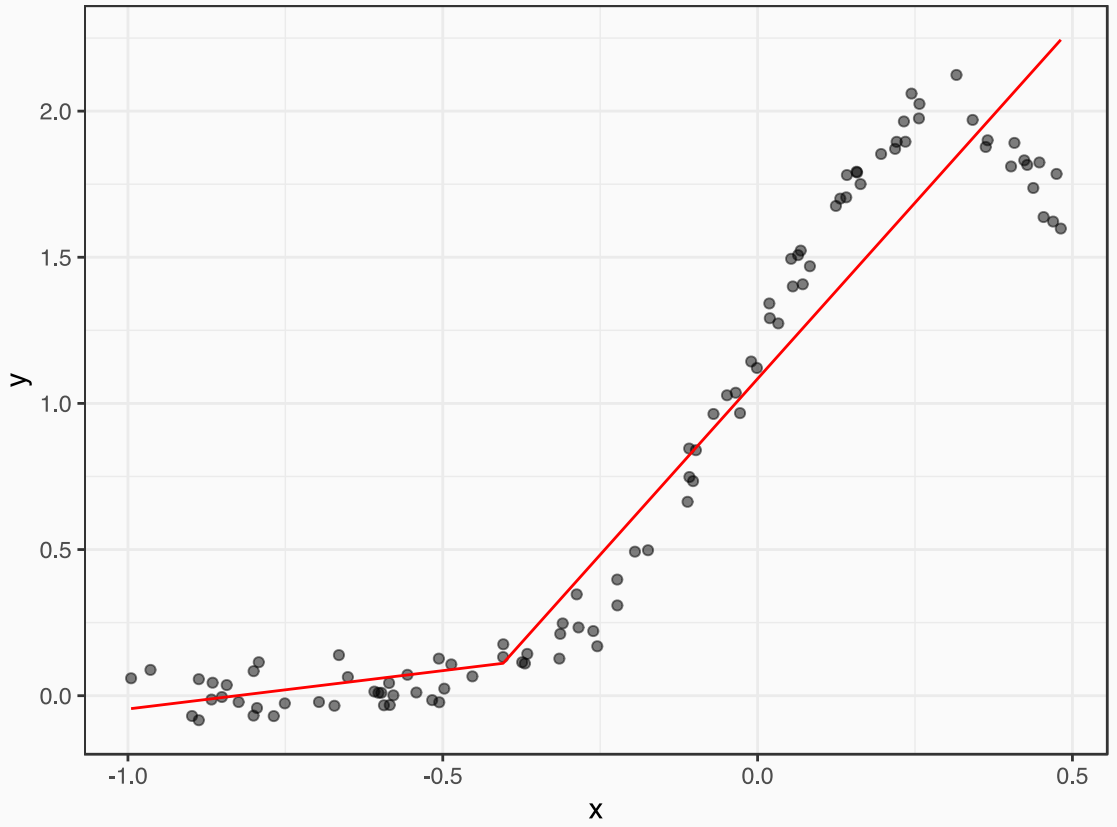

2.0
1.5
1.0
0.5
0.0
y
-1.0
-0.5
0.0
0.5
x

After searching through these data, the model evaluates all possible values of to find the best "cut" of the data. It finally chooses a value of -0.404.

To do this, it creates these two new predictors that isolate different regions of .

If we stop there, these two terms would be added into a linear regression model, yielding:




2.0
1.5
1.0
0.5
0.0
y
-1.0
-0.5
0.0
0.5
x

Similar to tree-based models, MARS starts off with a "growing" stage where it keeps adding new features until it reaches a pre-defined limit.

After the first pair is created, the next cut-point is found using another exhaustive search to see which split of a predictor is best .

Once all the features are created, a starts where model selection tools are used to eliminate terms that do not contribute meaningfully to the model.

Generalized cross-validation (GCV) is used to efficiently remove model terms while still providing some protection from overfitting.

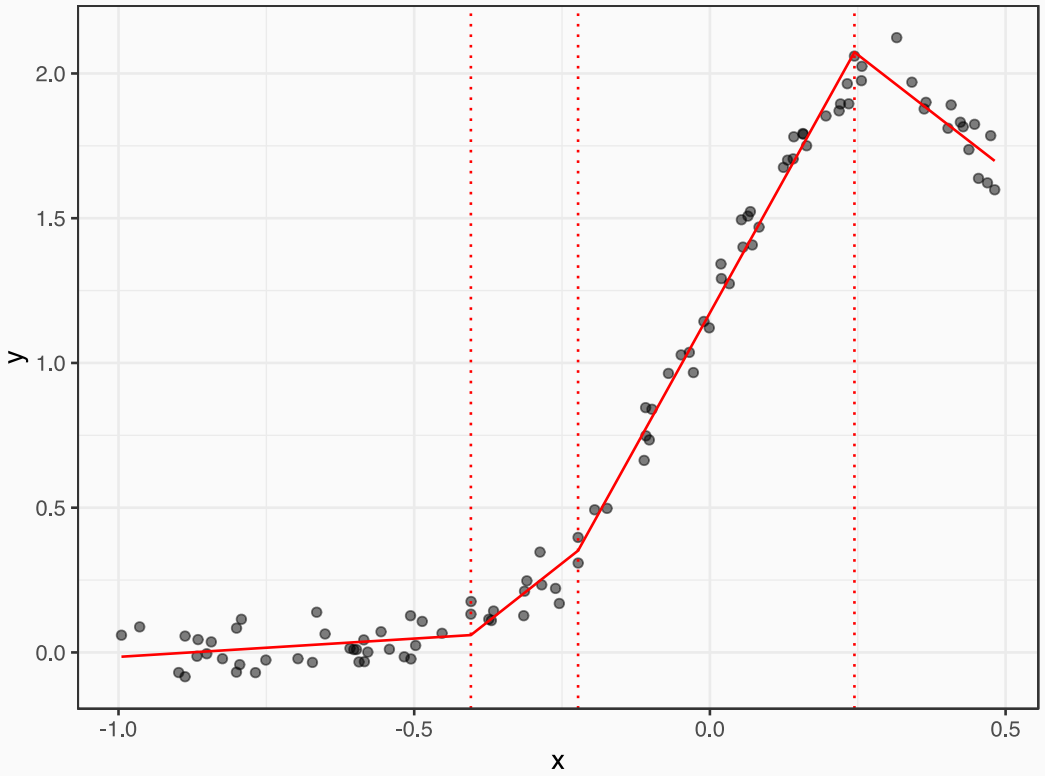
There are two approaches:

1. Use the internal CGV to prune the model to the best subset size. This is fast but you don't learn much and it may underselect terms.

2. Use the external resampling (10-fold CV here) to tune the model as you would any other.

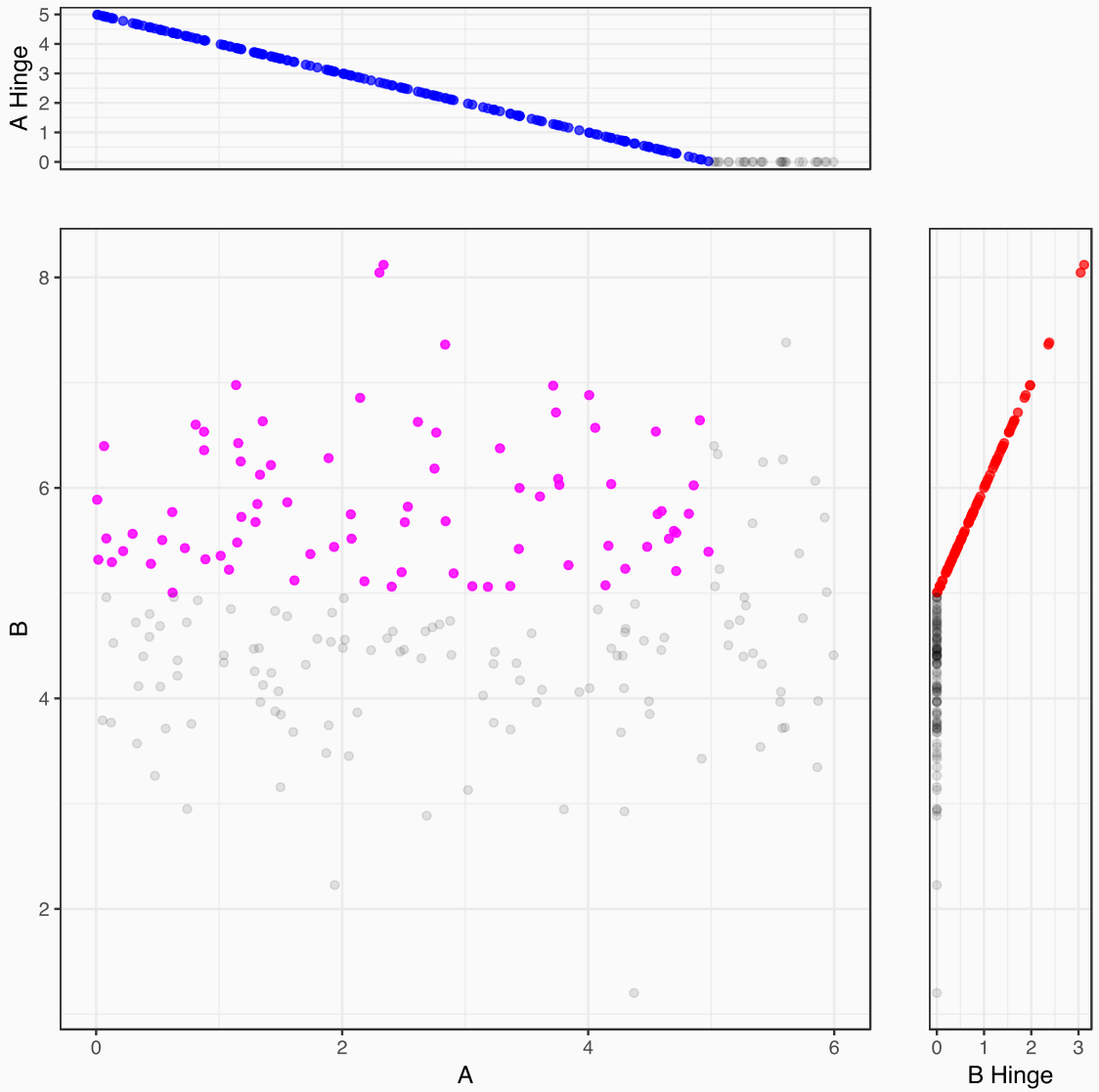
I usually don't start with GCV. Instead use method #2 above to understand the trends.

For the simulated data, the mars model only requires 4 features to model the data (via GCV).



The parameters are estimated by added the MARS features into ordinary linear regression models using least squares.

- The model also tests to see if a simple linear term is best (i.e. not split). This is also how dummy variables are evaluated.
- The model automatically conducts ; if a predictor is never used in a split, it is functionally independent of the model. This is really good!
- If an additive model is used (as in the previous example), the functional form of each predictor can be determined (and visualized) independently for each predictor.
- A MARS model also evaluates interactions of two hinge features (e.g.). This can be useful in isolating regions of bivariate predictor space since it divides two-dimensional space into four quadrants. (see next slide)


A Hinge
5
4
3
2
1
0
B
8
6
4
2
0
2
4
6
A
B Hinge
0
1
2
3

The package has a function but the package is far superior.

The function has both formula and non-formula interfaces. It can also be used with generalized linear models and flexible discriminant analysis.

To use the nominal growing and GCV pruning process, the syntax is

The feature creation process can be controlled using the , , and parameters although this can be somewhat complex.

There is a variable importance method that tracks the changes in the GCV results as features are added to the model.

There is a several ways to fit the MARS model using .

- avoids pruning using GCV and uses external resampling to choose the number of retained model terms (using the sub-model trick). The two tuning parameters are (number of retained features) and (the amount of interaction allowed).
- is also available and uses GCV. The parameter requires tuning.

I usually use the manual method to better understand the pruning process.

For preprocessing, there is no need to remove zero-variance predictors here (beyond computational efficiency) but dummy variables are required for qualitative predictors.

Centering and scaling are not required.

We can reuse much of the syntax to tune the model.

Running the resampling models (plus the last one), this takes 4.5m on my laptop.

There is no real barrier to running these in parallel.

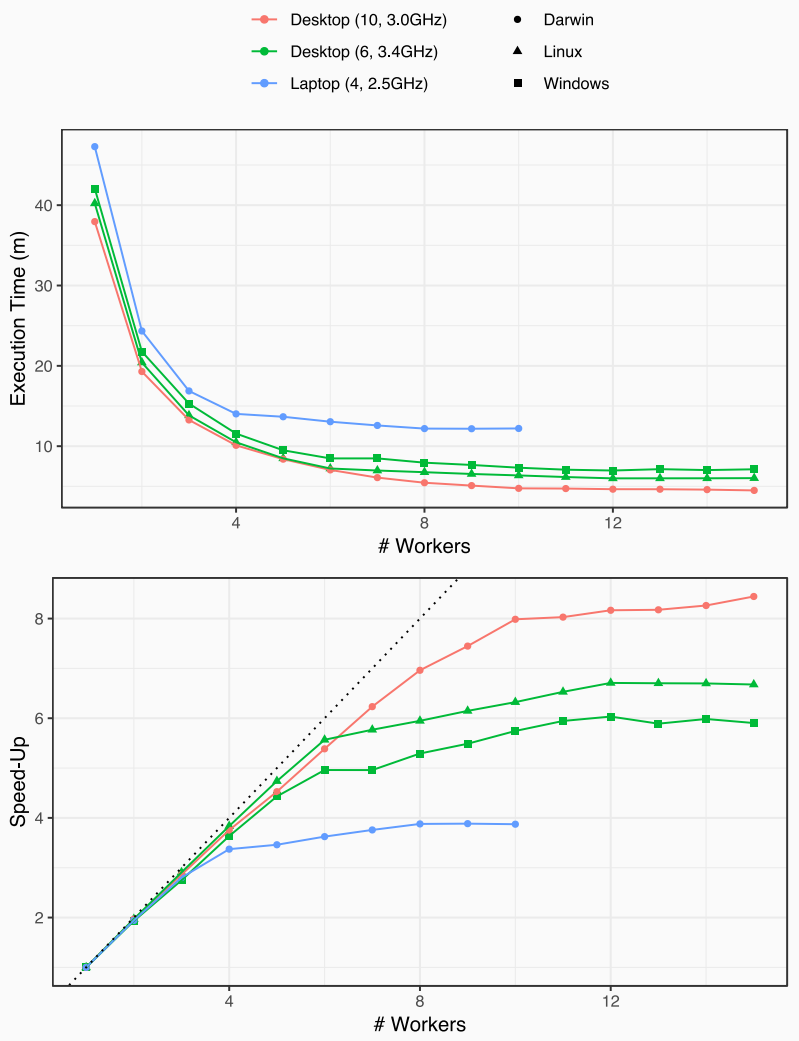
Can we benefit from splitting the fits up to run on multiple cores?

These speed-ups can be very model- and data-dependent but this pattern generally holds.

Note that there is little incremental benefit to using more workers than physical cores on the computer. Use

.

(A lot more details can be found in this blog post)



To loop through the models and data sets, uses the package, which can parallelize loops.

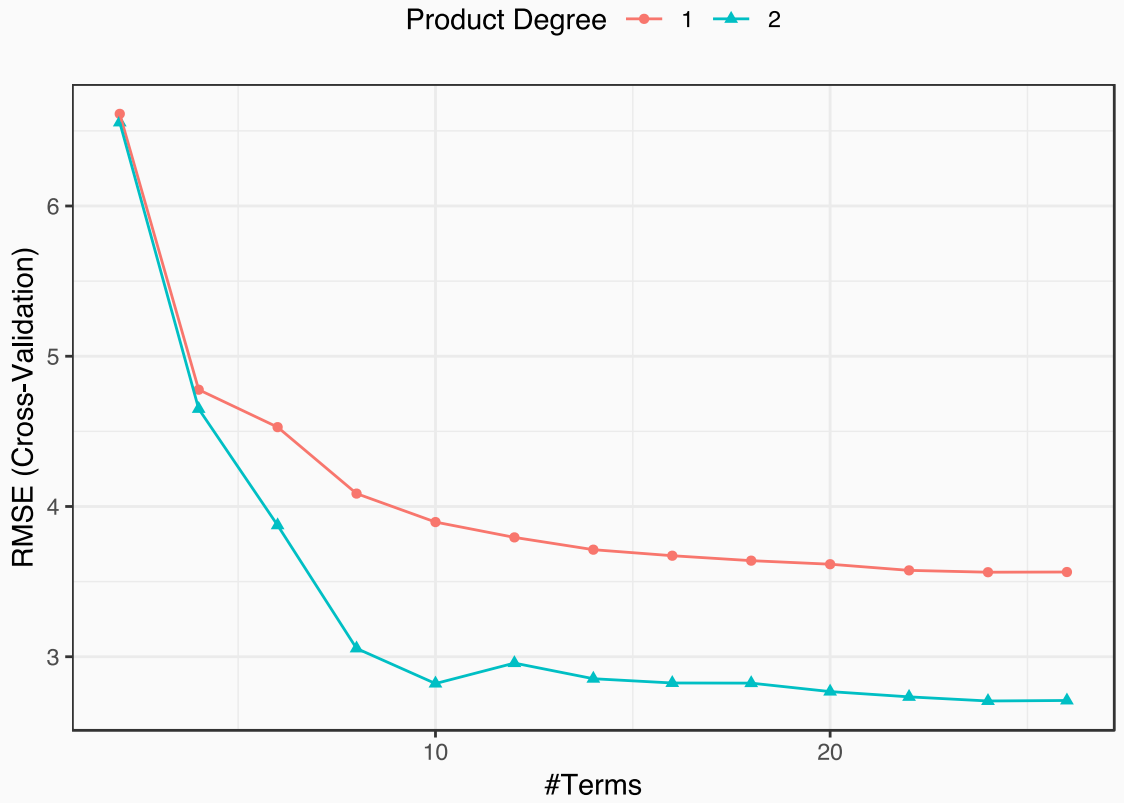
has a number of which allow various technologies to be used in conjunction with the package.

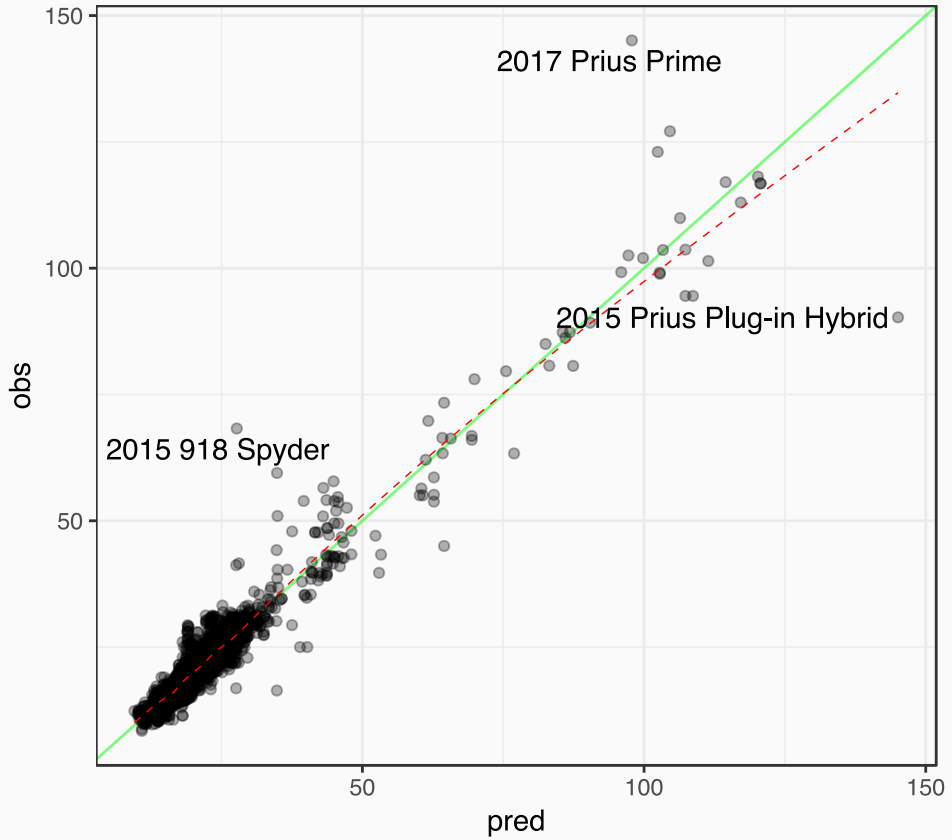
On CRAN, these are the " " packages, such as , , , , , , and .

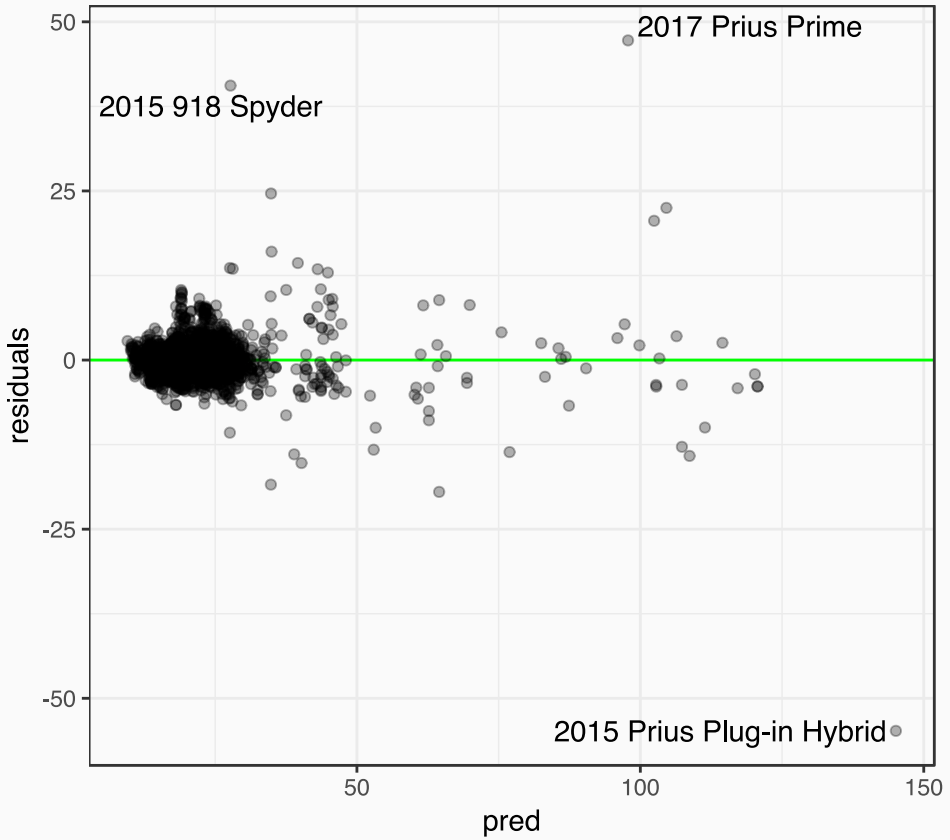
For example, uses the package, which forks processes to split computations (for unix and OS X). can be used for all operating systems.

To use parallel processing in , no changes are needed when calling .

The parallel technology must be with prior to calling :


Product Degree
1
2
RMSE (Cross-Validation)
6
5
4
3
10
20
#Terms

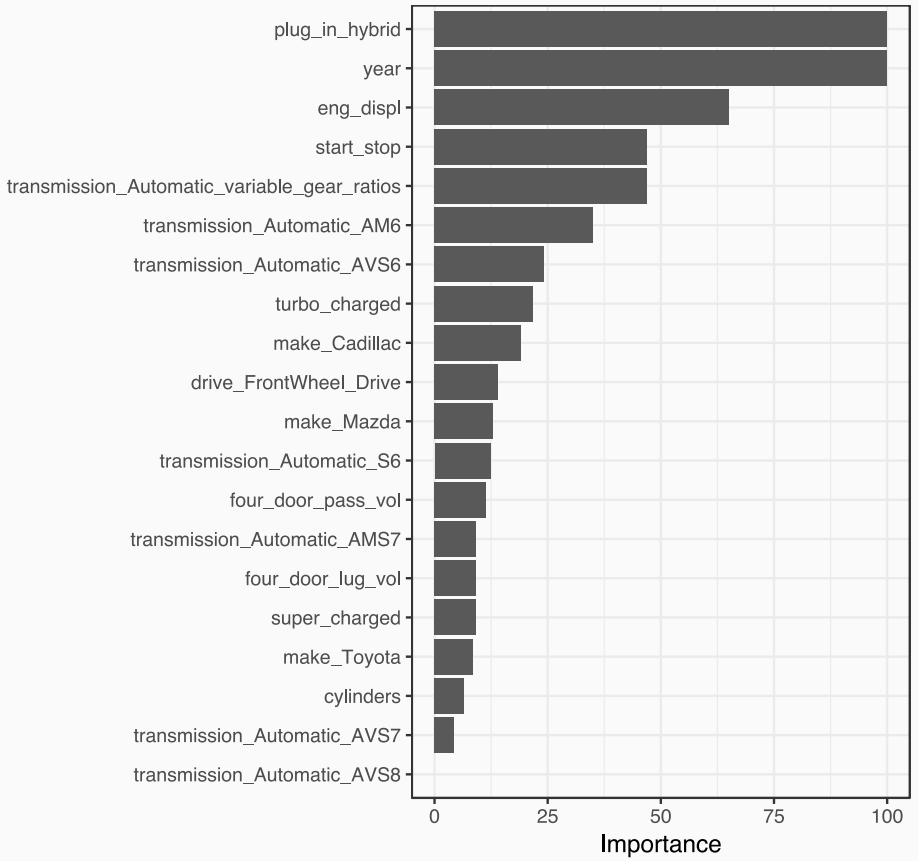

150
100
50
obs
2017 Prius Prime
2015 Prius Plug-in Hybrid
2015 918 Spyder
50
100
150
pred


50
25
0
-25
-50
residuals
2017 Prius Prime
2015 918 Spyder
2015 Prius Plug-in Hybrid
50
100
150
pred

Recall that as MARS adds or drops terms from the model, the change in the GCV statistic is used to determine the worth of the terms.

tracks the changes for each predictor and measures the variable importance based on how much the GCV error when the model term is added.

This is when multiple terms involve the same predictor multiple times.



The results are very similar to the previous model. This is not typical but it is faster (4.9-fold). We will exploit this in a moment.

Bagging is a method of creating .

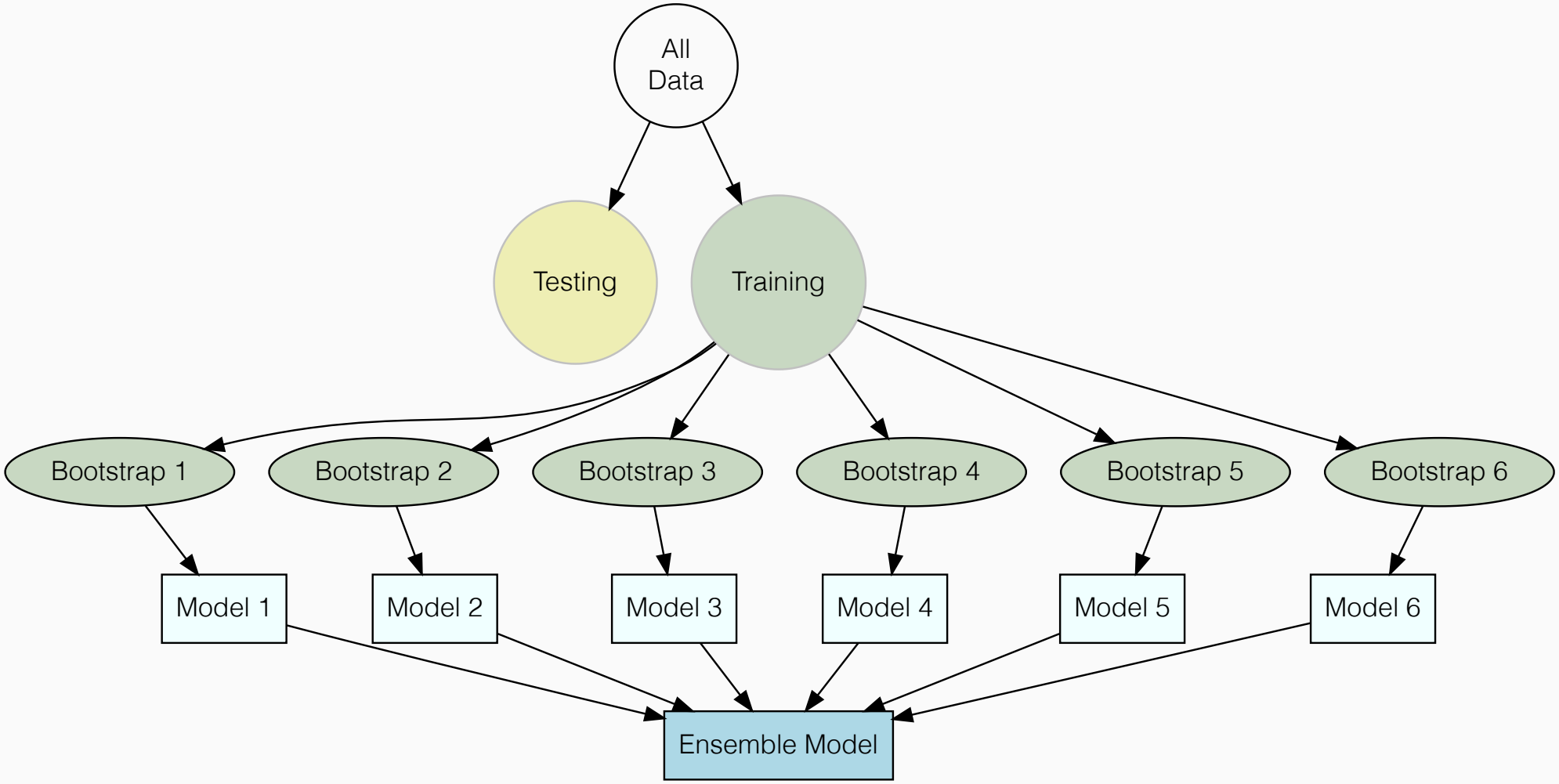
Instead of using the training set, many variations of the data are created that spawn multiple of the same model.

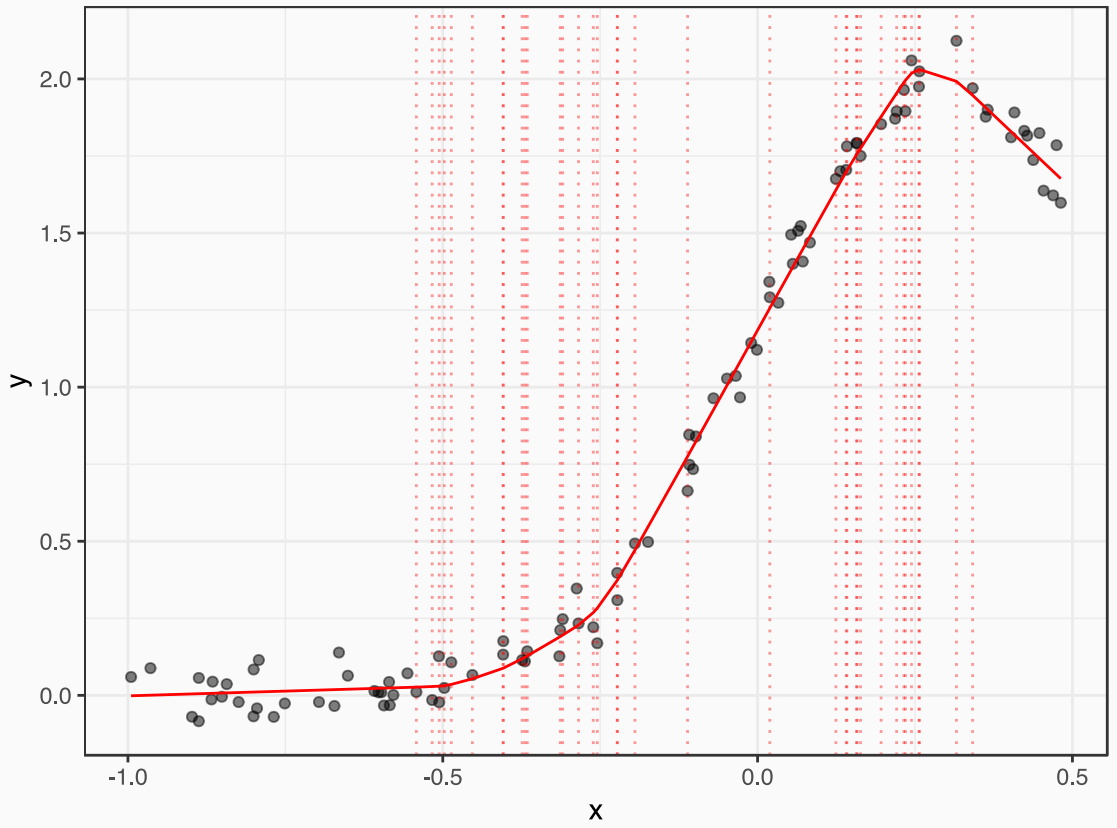
When predicting a new sample, the individual predictions are generated for each model in the ensemble, and these are blended into a single value. This reduces the variation in the predictions since are averaging pseudo-replicates of the model.

Bagging creates the data sets using a of the training set.

Bagging is most useful when the underlying model has some . This means that slight variations in the data cause significant changes in the model fit.

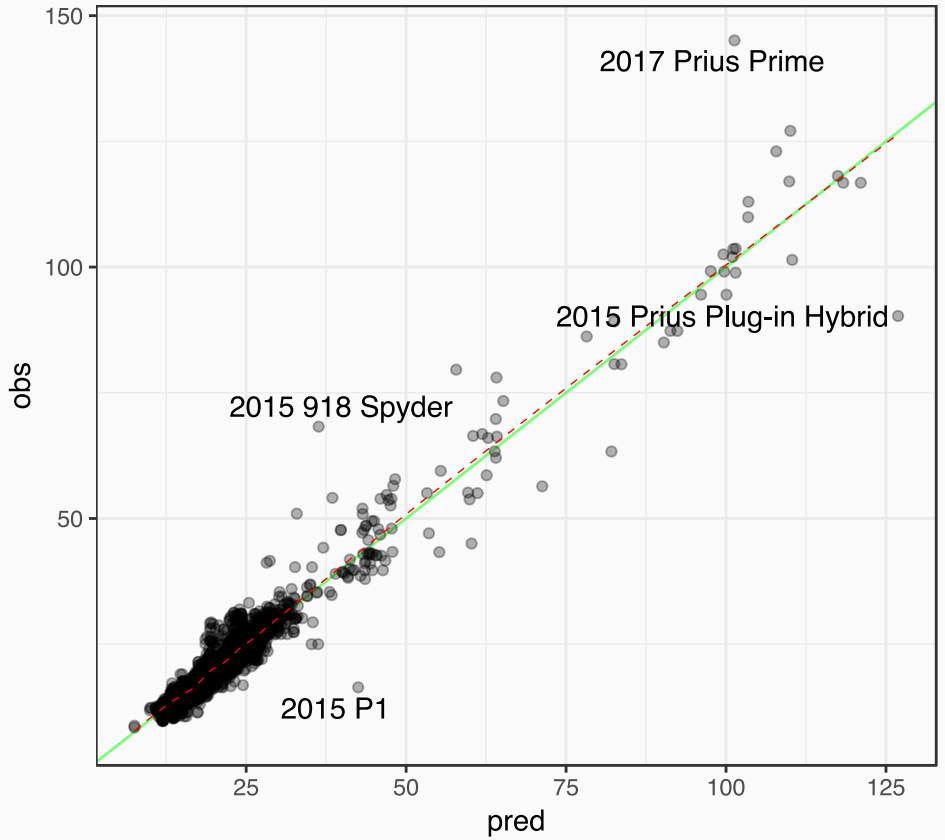
For example, simple linear regression would not be a suitable candidate for ensembles but MARS has for improvement. It does have the effect of the model predictions.

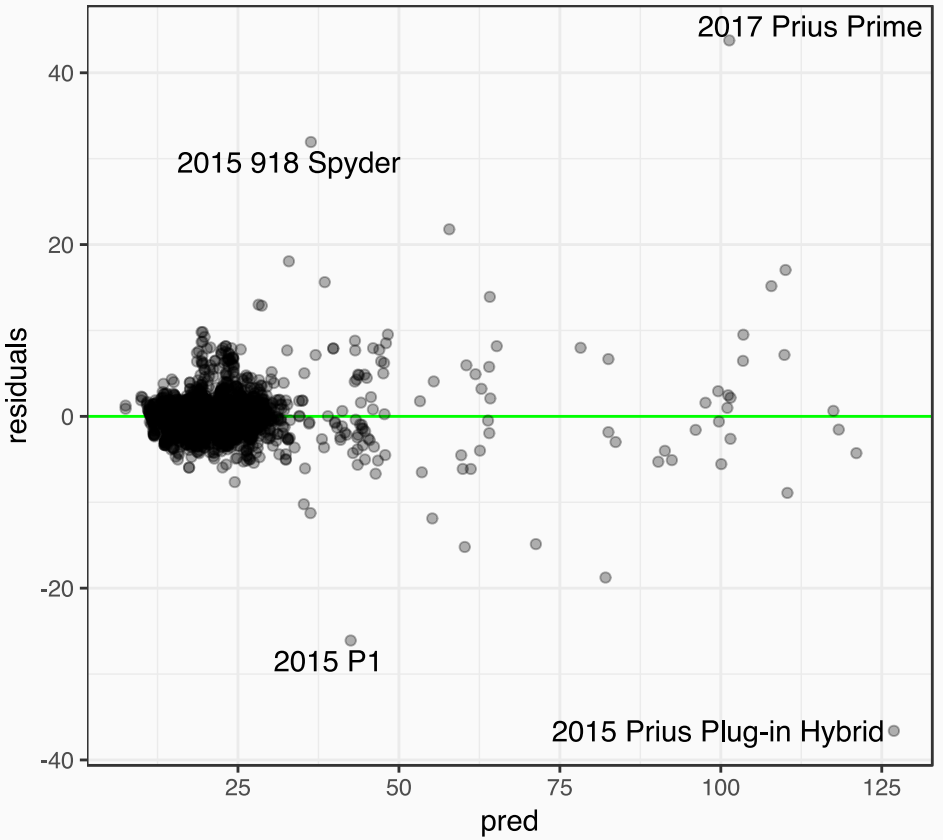

All Data
Testing
Training
Bootstrap 1
Bootstrap 2
Bootstrap 3
Bootstrap 4
Bootstrap 5
Bootstrap 6
Model 1
Model 2
Model 3
Model 4
Model 5
Model 6
Ensemble Model


2.0
1.5
1.0
0.5
0.0
y
-1.0
-0.5
0.0
0.5
x

On my laptop, this will take about 39m to run without parallel processing 😵

Smaller RMSE (was 2.69 mpg) but is it real?


150
100
50
obs
25
50
75
100
125
pred
2017 Prius Prime
2015 Prius Plug-in Hybrid
2015 918 Spyder
2015 P1


40
20
0
-20
-40
residuals
25
50
75
100
125
pred
2017 Prius Prime
2015 918 Spyder
2015 P1
2015 Prius Plug-in Hybrid

First, we can use the function in to collect and collate the cross-validation results across the different models.

The package is designed to estimate the relationship between the (i.e. RMSE) as a function of the model type (i.e. MARS) in a way that takes into account the resample-to-resample covariances that can occur.

A Bayesian linear model is used here for that purpose.

I recommend the book if you are new to Bayesian analysis.

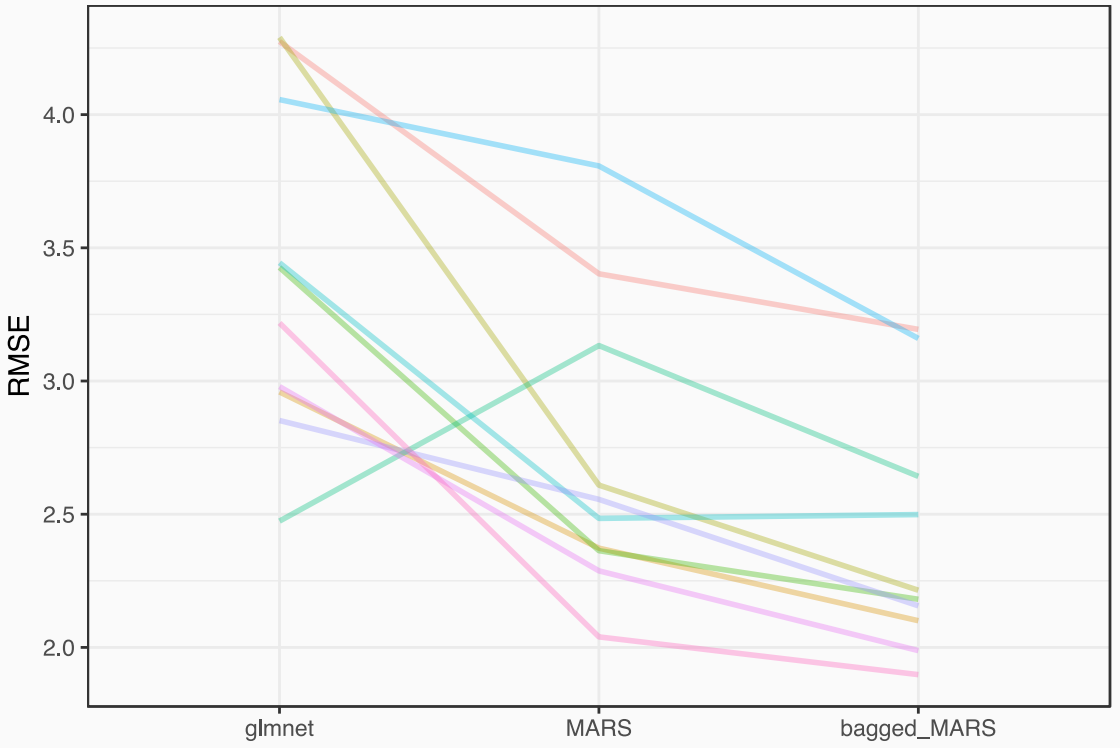
Bayes' Rule will be discussed in more detail in the Classification notes to come.

If we did a basic ANOVA model to compare models, it might look like:

where the are indicator variables for the model (, MARS, etc).

However, there are usually resample-to-resample effects. To account for this, we can make this ANOVA model :

where is the th cross-validation fold.


RMSE
4.0
3.5
3.0
2.5
2.0
glmnet
MARS
bagged_MARS

We might assume that each

and that the parameters have some multivariate normal distribution with mean and some covariance matrix. The distribution of the values, along with a distribution for the variance parameter, are the

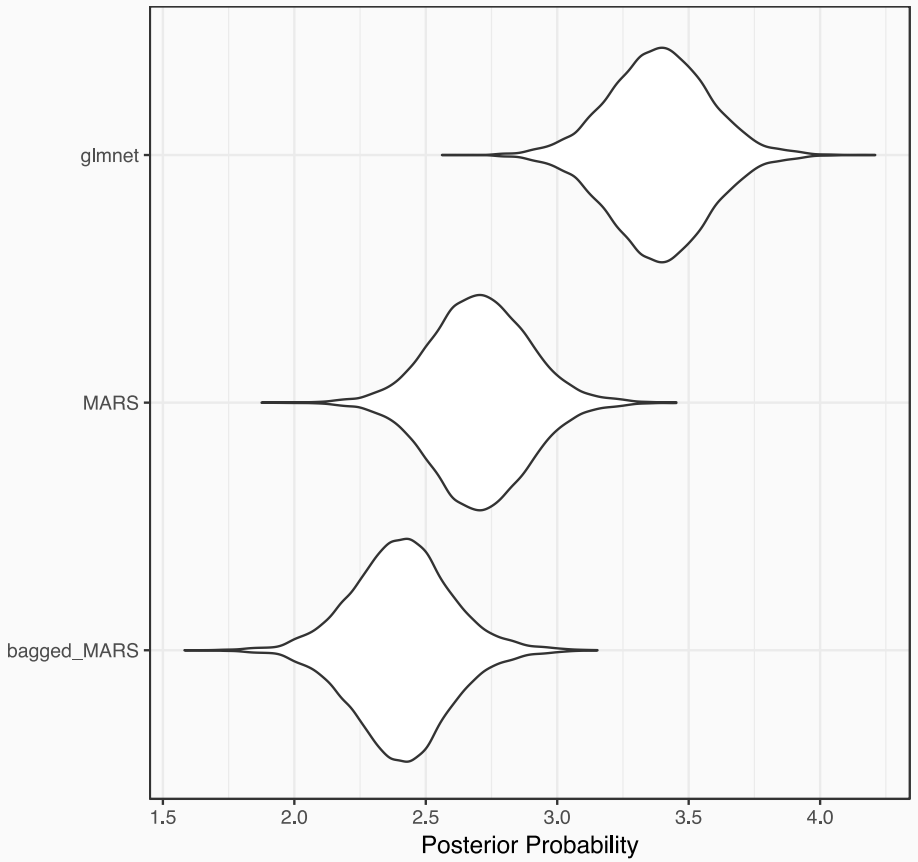
.

Bayesian analysis can be used to estimate these parameters. uses Stan to fit the model.

There are options to change the assumed distribution of the metric (i.e. gamma instead of normality) or to transform the metric to normality. Different variances per model can also be estimated and the priors can be changed.

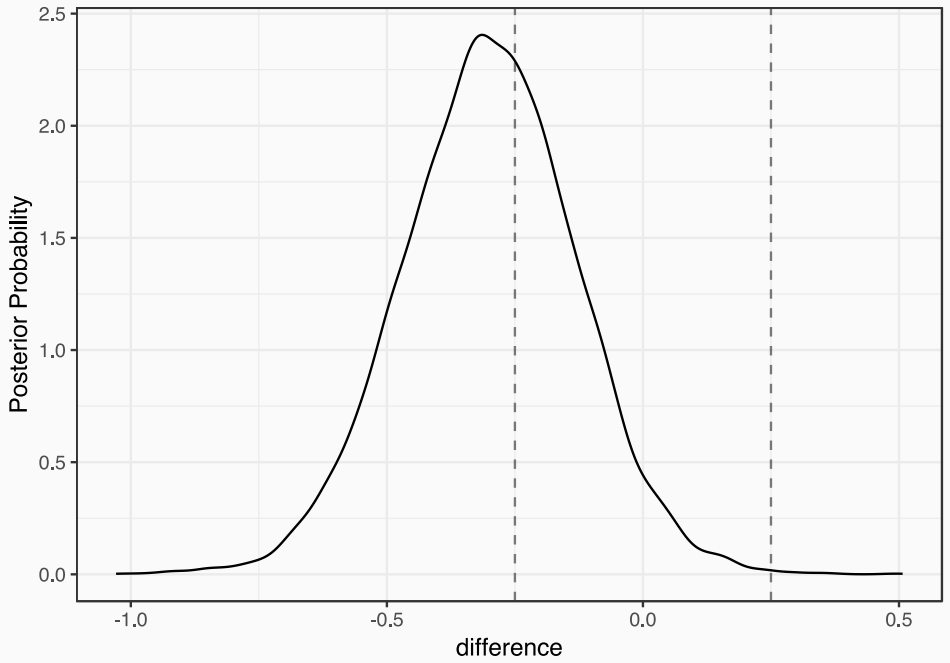
can take the object as input, configure the Bayesian model, and estimate the parameters:

Bayesian analysis can produce probability distributions for the estimated parameters (aka the). These can be used to compare models.



These are 90% .

Once the posteriors for each model are calculated, it is pretty easy to compute the posterior for the differences between models and plot them.



If we know the size of a practical difference in RMSE values, this can be included into the analysis to get ROPE estimates (Region of Practical Equivalence).

If we think that 0.25 MPG is a real difference, we can assess which models are practically different from one another.

```

```

```

```

is the probability that the difference in RMSE is practically negative based on our thoughts about .

Making predictions on new data is pretty simple: