Simulation testing:

Many statistical and machine learning models can be represented by the equation

$$y_i = F(X_i) + \epsilon_i$$

where y is the quantity we wish to model, X_i is a vector of variables that can be used to predict y, F is a function of the predictors and ϵ_i represents variations in the value of y that cannot be explained by $F(X_t)$. The goal of regression models and many machine learning algorithms is to learn the function F from a set of observations of the predictors X_i and y_i.

An important observation we can make about this model is that y_i is not only a function of the predictors X_i, but the model also includes the parameter ϵ_i, which can take on different values each time we make an observation of y_i. This means than no matter how well we are able to reconstruct the function F from a data set there will be a limit on how well we are able to predict the value y_i, because we do not know what the value of ϵ_i will be in that particular case.

This creates issues when we are trying to estimate the function F from a data set. Our data will contain information about the function F, but this information will be corrupted by noise introduced by the error or residual terms ϵ_i. This will limit our ability to reconstruct the function F exactly from a finite data set, regardless of the model we choose.

Statistical modeling techniques use probability theory to determine how well the function F can be estimated from a data set, but these methods cannot be readily applied to machine learning algorithms. Instead we can test how well an algorithm is able to estimate the relationship between the predictors X and y empirically by

simulating data sets where the relationship between the predictors and y is known. This allows us to compare the relationship estimated by our model to the true relationship to determine how well the algorithm works.

Steps:

Simulation testing requires three steps: simulating data sets, fitting models and comparing the results of the fitted models to the functions used to simulate the data. I have already done step one, generating data sets.The next steps will be to write code to fit models to these data sets and compare the resulting models to the functions used to generate the data set.

In terms of practical implementation we will need to write a method that takes the simulated data set as an argument and returns a random forest model. Currently each of the data sets are saved in their own directory along with information about how they were made. The y values are saved in files called y.csv and the predictors are saved in files called X.csv. Ideally we would write functions that take the directory as an argument, read in y.csv and X.csv and use them to fit a model that is returned by the function.

Once this is done we will need to develop a way to compare the random forest models we fit to the models used to generate the data. This cannot be done directly because the functional form of the random forest models does not match the models used to generate the simulated data sets, but we can start by comparing some key features. The first comparison that we can make is the R^2 value for the random forest with the proportion of the variance that can be explained in the simulated data set. This value is saved for each data set in the hyper_params.csv file in the column named total effect.

The next step will be to compare the amount each of the variables contributes to in the function use to simulate the data set to the variable importance scores produced by

the random forest models. The “importance” of each variable in the simulated data sets can be calculated by squaring the effect sizes of each variable. These values can be found in the par.csv file in the column labeled effect_size. These values will not be on the same scale as the variable importance scores produced by the random forest models, but they will each define an ordering of the variables from most important to least important. We can compare these orderings to assess the skill of the random forest models. These steps are summarized below.

3 steps:

Write a function that takes the directory with the simulated data set and returns a random forest model.

Compare the total effect of the predictors (found in hyper_params.csv) used to simulate the dat sets to the R^2 values of the random forest models. How do the R^2 values change as the total effect value for the simulated data increases?

Compare the variable importance scores produced by the random forest models to the squared effect sizes (found in par.csv). To determine how well the random forest models can identify the predictors that explain the most variability in the values of y.