Intermediate Models for Prediction and Variable Selection for Psychologists - Random Forest

Francisco N. Ramos

Intermediate Statistical Models for Prediction and Variable Selection for Psychologists - Random Forest and LASSO

In this document, I will walk through a random forest model and variable selection procedure on a psychological dataset. Although random forest models are better explained elsewhere, to be brief, random forest models are built from single decision trees, which predict a dependent variable (continuous or binary, depending on whether you are interested in regression or classification respectively) based on binary answers to other questions. For example, a tree might first check "Is the social anxiety score over or under 10.5?", with a "Yes" leading to a different prediction adjustment than a "No", then it will continuously repeat the process with new questions like "Is sex male or female?", until it has finalized its prediction. A random forest model is built from averaging the observations of thousands or more trees, all of which ask different "questions" (another tree might start with "Is social anxiety score over or under 11.3?").

Random forest models are often accurate and robust (though they can overfit), can handle missing and categorical data as well as problems where the number of predictors exceeds the sample size, and frankly are not too difficult to implement, though they can be difficult to interpret.

Please be aware that this is not a tutorial on identifying the absolute best predictive model, which is described ad infinitum elsewhere.

Another note: for didactic purposes, we will be using a method that is essentially what statisticians would call "stepwise selection", "backwards selection" to be specific. Let me just be clear - backwards selection is essentially NEVER the best way to do variable selection. We are simply going through stepwise selection as a tool to help discuss the relationships between variables in specific models. If possible, you should either a priori identify the subset of variables (or interactions between variables) to evaluate based on previous literature and expert

knowledge OR, if you must, use purely data-driven selection, e.g. automated methods with clear "reward" and "loss" functions (which we will show examples of later).

I have not seen the data ahead of time. The analyses described hereafter are entirely novel.

The Data

Importantly, we are conducting a regression task, not a classification task, as our dependent variable is continuous. The first main task we'll be attempting to address is that we would like to predict arithmetic performance score in German students from a set of predictor variables including age, sex, self-reported math anxiety, and others. Our second main task is that we would like to have a good sense of which predictor variables appear useful in predicting arithmetic performance and which ones are not, so future researchers do not have to waste resources collecting unimportant variables for no or minimal improvement in prediction.

The dataset we will investigate in this document is characteristic of many psychology datasets — it has a good but not at all large sample size (735 N before being split into test/training sets), a relatively high number of potential predictors (20+ collected as part of the survey, many of which are likely completely unhelpful in prediction), and what is likely middling to low effect sizes for predictors. To simplify our explanation, I have arbitrarily selected ten variables from the dataset for our analysis (don't do that in your real work). For the purposes of this document, only these ten variables "exist". References (including the dataset) are at the end of the LASSO addendum to this document.

Below, I will list the ten variables we are interested in studying as predictors of arithmetic performance score followed by the syntax for their object in R.

Dependent Variable:
Arithmetic performance/`sum_arith_perf`, as measured by "the number of correctly solved problems in order as instructed" on a simple arithmetic speed test.

Predictors:
Age/`age`, as measured in years.

Sex/`sex`, where 1 = male, 2 = female, and 3=other. Participants who ignored this question were removed.

Neuroticism/`score_BFI_N`, as measured by the sum score of the 8 items of the Big Five Inventory (short version) pertaining to neuroticism.

Math anxiety/`score_AMAS_total`, as measured by the sum score on the Abbreviated Math Anxiety Scale.

General trait anxiety/`score_GAD`, as measured by sum score on the Generalized Anxiety Disorder Screener (GAD-7).

Math self-efficacy/`score_PISA_ME`, as measured in the PISA 2012 study using the sum score of six items.

General state anxiety/`score_STAI_state_short`, as assessed by the sum of the five-item scale STAI-SKD.

Test anxiety/`score_TAI_short`, as measured by the sum score of the 5 items on the short version of the Test Anxiety Inventory.

Math self-concept/`score_SDQ_M`, as measured by the sum score of the four math-related statements on the Self-Description Questionnaire III. Evaluates variables such as one's comfort/enjoyment/pride with math, whereas self-efficacy evaluates one's self-confidence in math abilities.

Language self-concept/`score_SDQ_L`, as measured by the sum score of the four language-related statements on the Self-Description Questionnaire III.

Arithmetic performance/`sum_arith_perf`, as measured by "the number of correctly solved problems in order as instructed" on a simple arithmetic speed test.

Cleaning the data and a little Exploratory Data Analysis

We'll start with importing and cleaning the data to make sure it fits our task.

```
library(here)
library(tidyverse)
library(caret)
library(readxl)
library(readr)
library(ggplot2)
library(randomForest)
```

The `here()\` function is extremely convenient for creating paths and directories that won't break when files are moved or when you reproduce the work on another computer. More information is available at the following URL: https://here.r-lib.org/

Below, we'll again use the here function to tell the `readr()` function where to go to get the dataset. Our data was stored in the "Main Script" folder, in a subfolder called "OSF archive", and named "AMATUS_dataset.csv". Use that info to read the code below. Notice how we use a `normalizePath()` function to convert relative or incomplete path names to absolute path names, which helps when switching OSes or from cloud to local storage.

```
here::i_am("Dr. Lai Feature Selection Project 9.13.24.rproj")
## here() starts at /Users/frankie/Desktop/Dr. Lai Feature Selection Project 9.13.24
#I manually set "here" to the folder containing my rproj file.
```

```
file_path <- normalizePath(here::here("Main Script/OSF archive/AMATUS_dataset.csv"))
# Also adjust this as needed.

#Importing dataset
amatus <- read_csv2(file_path, c("", "NA"), col_names = TRUE)
## i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
# Note that the c() function is defining
#what NA values look like for the import function.

View(amatus)
# Best command to view data, in my opinion.

amatusclean <- amatus[!is.na(amatus$sum_arith_perf), ]
# Removing the individuals who did not complete the performance test in order as instructed.
amatusclean <- amatusclean[!(amatusclean$sample %in%
c("german_teachers", "belgian_teachers")), ]
# Removing the 2 teacher samples from the dataset, as we are only interested in the student s

#Data cleaning:
amatusclean <- amatusclean %>%
  filter(!is.na(sum_arith_perf)) %>%
  mutate(across(c(sex, age_range, breaks, honesty, native_speaker, noise), as.factor))
# `filter()` removes the
# individuals who did not complete the performance # test in order as instructed, `mutate`
# applies the `as.factor()` function to all
# categorical (factor) variables.
```

Now, we'll just check out a quick histogram of the dependent variable, arithmetic math performance.

```
ggplot(
  amatusclean,
  aes(x = sum_arith_perf)
) +
  geom_density(fill = "lightblue", alpha = 0.7) +
  theme_minimal()
# I set up the theme for all plots in hidden code, # so i won't include it again after this.
```

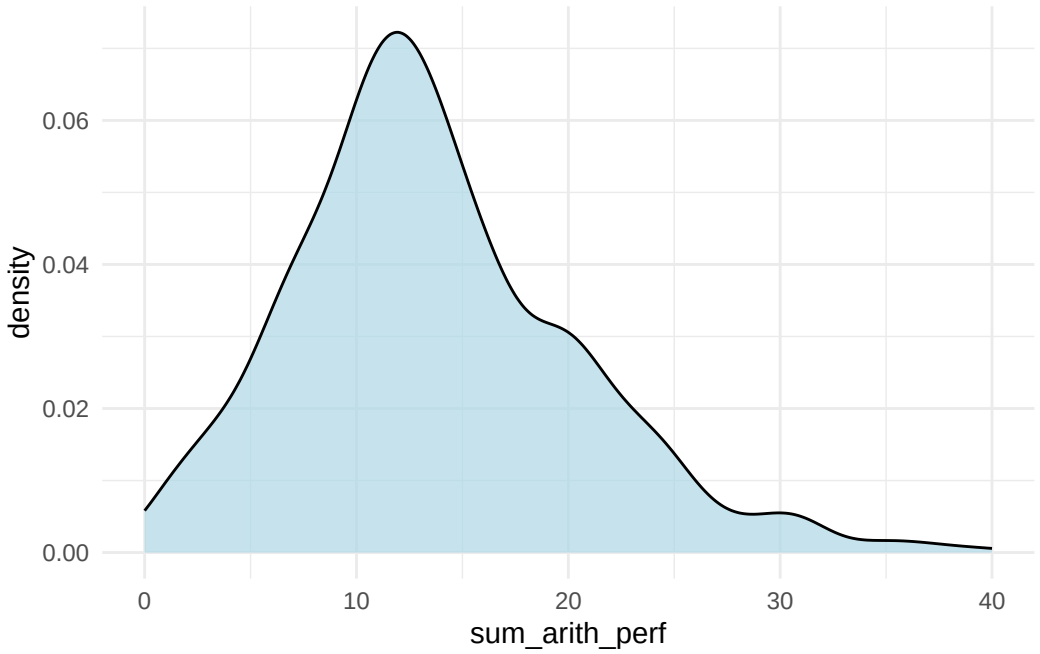


Figure 1: Distribution of outcome variable: Arithmetic Math Performance

Our data is clearly skewed right (Figure 1). This indicates that most people scored around the "low" end of all possible values.

Let's also look at some predictors.

```
ggplot(
  amatusclean,
  aes(x = score_AMAS_total)
) +
  geom_density(fill = "lightblue", alpha = 0.7)

ggplot(
  amatusclean,
  aes(x = score_BFI_N)
) +
  geom_density(fill = "lightblue", alpha = 0.7)
```

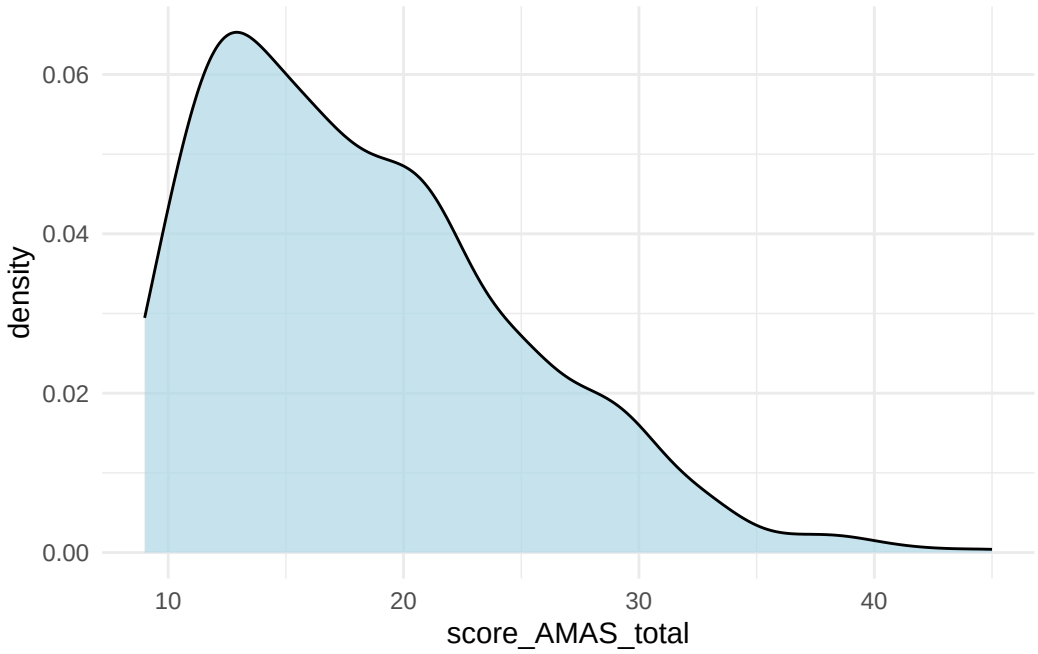


Figure 2: Distribution of math anxiety and neuroticism

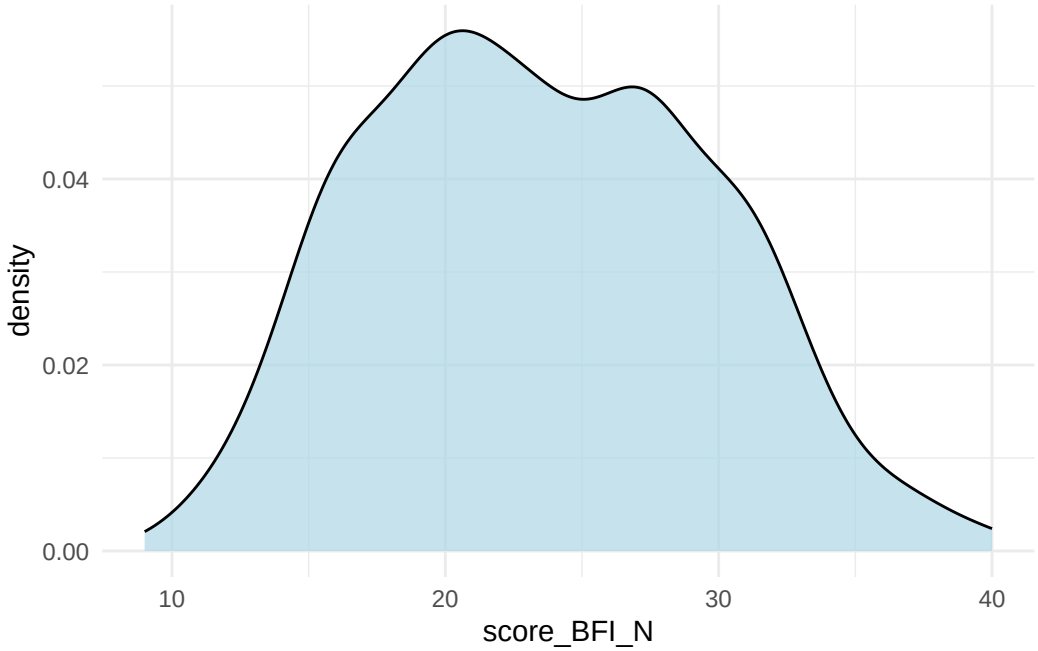


Figure 3: Distribution of math anxiety and neuroticism

One predictor's distribution is very normal, the other isn't. The rest of the features are a grab bag too. Luckily, distribution doesn't really matter for random forest models.

Normally, we wouldn't care about standardization of our variables given that random forest is a tree-based model. However, because we are interested in variable importance metrics (since we are partly focusing on feature selection), we actually do need to scale our features. This is because variable importance metrics are skewed towards predictors that have a wider range of values.

We will choose to standardize our variables using the `scale()` function, which you may recognize. The `scale()` function will center (subtract mean from the value for a mean of 0) and scale (divide value by its standard deviation for an SD of 1) our continuous predictors. We don't include categorical variables. We'll also go ahead and get started by creating our first random forest model. Let's try using the `caret` package, a relatively general machine learning package.

(Another function that can be used to standardize is the `preProcess()` function in the `caret` package, and it may be better or easier to use for your purposes. I just like `scale()`.)

First Model Setup

Setting up the train/test split

```
library(caret) #n eeded first for the inTrain function below.
set.seed(39)
# I'm going to set the random seed BEFORE every
# random generation, including model fitting, for clarity.
inTrain <- createDataPartition(amatusclean$sum_arith_perf,
  p = 0.7, list = FALSE
)
# 0.7 will sort 70% of the records into training set, 30% into the test
# set. I selected 0.7 to have a decent number of N in the test set.

# create training vs test data.
training <- amatusclean[inTrain, ]
test <- amatusclean[-inTrain, ]
View(training)
View(test)

# Create the scaled training set.
# Start by isolating the variables we are studying.
numeric_predictors <- c(
  "score_BFI_N", "score_AMAS_total", "age", "score_GAD",
```

```
  "score_PISA_ME", "score_STAI_state_short",
  "score_TAI_short", "score_SDQ_L", "score_SDQ_M"
)
categorical_variable <- "sex"
response_variable <- "sum_arith_perf"

# Create a new dataframe by scaling and centering
# numeric predictors.
scaled_training <- training %>%
  mutate(across(all_of(numeric_predictors), ~ scale(.) %>% as.vector())) %>%
  # Scale and convert to vector, so that the
  # output is not in matrix form.
  select(all_of(numeric_predictors), all_of(categorical_variable), all_of(response_variable))
# Select and retain only the relevant columns.

View(scaled_training)
```

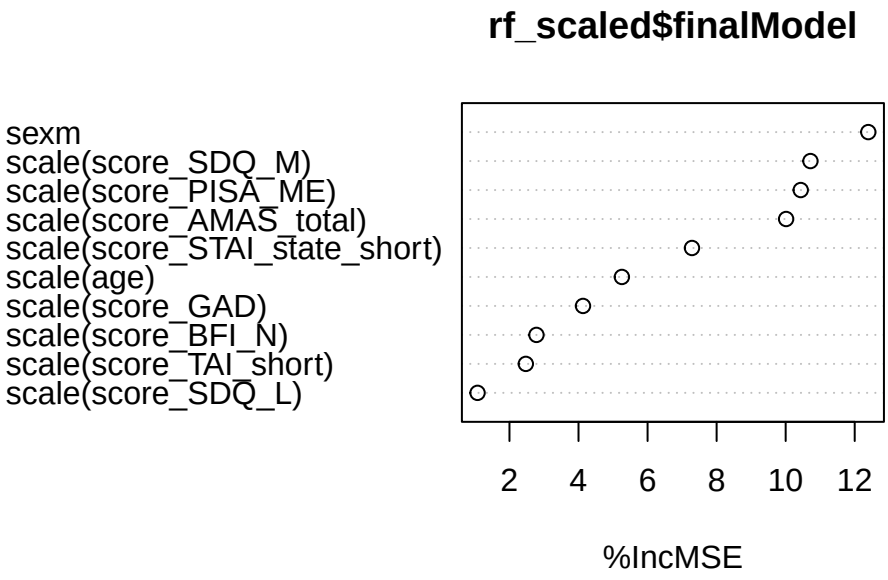
Hyperparameters and creating the Random Forest Model

A side note: There are several different validation methods you can use in `caret`, including bootstrapping, cross-validation, and repeated cross-validation. Repeated cross-validation should pretty much always be used when computationally appropriate. Bootstrapping may be compelling here due to the small dataset, but repeated cross-validation should work just as well as bootstrap in smaller samples and should do the job for us. In the `trainControl()` function you'll see below, you can increase the `number` argument to increase the number of folds in the dataset and you can increase the `repeat` argument to increase the number of times the entire cross validation gets repeated, at the expense of computational power.

```
# The argument `mtry` is a hyperparameter of the number of randomly
# selected predictors to consider splitting at each node of the decision
# tree.
# More on this after the code.
control <- trainControl(
  method = "repeatedcv", number = 10, repeats = 3,
  search = "random"
)
# `random` as the value for the`search` argument will test
# `mtry` values in the range of [1, # of predictors].
# Can try the same value twice, so not ideal for us.
control <- trainControl(
  method = "repeatedcv", number = 10, repeats = 3,
  search = "grid"
```

```
)

# `grid` as the value of the `search` argument will pick preselected values # for `mtry`, wh
tunegrid <- expand.grid(.mtry = c(1:10))
# This function creates a data frame from all combinations of the
# supplied vectors/factors. This is the grid of values for the
# hyperparameter `mtry`.

# Thus, this would test values of mtry including and between 1 and 10.
# You can't try more `mtry` values than there are predictors.

# Let's start with the code above to test all current possible values of `mtry` (1 to number

############
# You can also try to use "default" values.
# In random forest for regression, predictors/3 is
# often the "default" value of this parameter.
# In classification, it is the square root of the # of predictors.
# Just including some code here in case you want to try it.
# But make sure to fix your `tuneGrid` argument too!
x <- amatusclean[, 1:10]
mtry <- ncol(x) / 3

# As you'll remember, the REAL parameters are below:
control <- trainControl(
  method = "repeatedcv", number = 10, repeats = 3,
  search = "grid"
)
tunegrid <- expand.grid(.mtry = c(1:10))

# Now, to run the random forest
set.seed(40)
rf_scaled <- train(
  sum_arith_perf ~ scale(score_AMAS_total) + sex + scale(age) +
    scale(score_BFI_N) + scale(score_GAD) + scale(score_PISA_ME) +
    scale(score_STAI_state_short) + scale(score_TAI_short) +
    scale(score_SDQ_M) + scale(score_SDQ_L),
  data = training,
  importance = TRUE, method = "rf", tuneGrid=tunegrid, trControl = control
)
# Notice that in this example, we are using data = training, NOT the
# scaled training set, as well as the `scale()` function again.
```

```r
# Also, no `tuneLength` argument. Just `trControl`.

print(rf_scaled)
## Random Forest
##
## 516 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 465, 466, 462, 465, 463, 464, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##    1    6.101332  0.1544490  4.748449
##    2    6.146862  0.1415140  4.780453
##    3    6.199815  0.1318351  4.826108
##    4    6.214933  0.1304793  4.836937
##    5    6.240131  0.1273513  4.858912
##    6    6.253618  0.1261627  4.864408
##    7    6.268388  0.1240168  4.886805
##    8    6.279579  0.1227821  4.889534
##    9    6.278758  0.1229835  4.888298
##   10    6.271230  0.1252135  4.883523
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 1.
varImpPlot(rf_scaled$finalModel, type = 1)
```



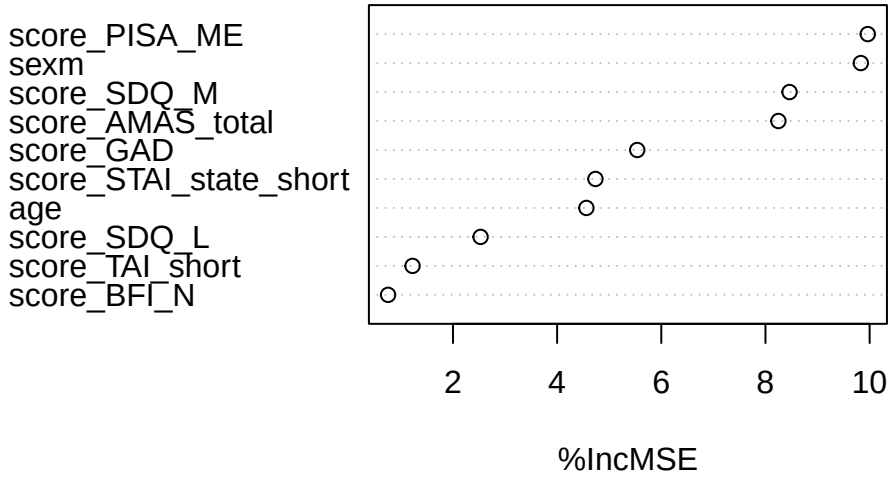
As you can see in the first model (`rf_scaled`), if it was up to caret, the best predictive method would have us randomly sample 1 predictor to consider splitting at each node of the decision tree (e.g., `mtry`=1). That is a very random method, and means we probably have very few predictors that actually provide value in predicting math performance. This is expected in this dataset and in much psychological research, as we stated early on. Although `mtry=1` is too random, we clearly need to use a low value for mtry. The difference between the default (# of predictors/3) number for `mtry` and `mtry=2` isn't too much, so we'll set 2 as the minimum value and make sure our code tests a range of mtries, just in case another value ends up being better than 2 (which I don't expect to be the case).

Refining the First Model

Let's exclude 1 from the range of mtry values and go again.

```
control <- trainControl(
  method = "repeatedcv", number = 10, repeats = 3,
  search = "grid"
)

tunegrid <- expand.grid(.mtry = c(2:8))
# This one says try values between and including 2-8.
# Since our optimal mtry at first was 1, the optimal mtry is probably
# going to be low (and will likely be 2).



set.seed(40)
```

```r
rf_mtry2 <- train(
  sum_arith_perf ~ score_AMAS_total + sex + age + score_BFI_N +
    score_GAD + score_PISA_ME + score_STAI_state_short + score_TAI_short +
    score_SDQ_M + score_SDQ_L,
  data = scaled_training, importance = TRUE,
  method = "rf", tuneGrid = tunegrid, trControl = control
)
# Notice we keep using the importance=TRUE parameter, which calculates
# variable importance metrics. We need this to be set to TRUE to be able
# to see varimp plots later.

print(rf_mtry2)
## Random Forest
##
## 516 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 465, 466, 462, 465, 463, 464, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     6.148020  0.1418479  4.781207
##   3     6.186392  0.1348754  4.821405
##   4     6.211853  0.1314235  4.835274
##   5     6.245703  0.1259847  4.862120
##   6     6.255958  0.1264024  4.861685
##   7     6.262649  0.1245588  4.872607
##   8     6.283629  0.1229101  4.900340
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
varImpPlot(rf_mtry2$finalModel, type = 1)
```



2 was the best mtry value for this 10-variable model, as expected. R squared of around .14ish is obviously a bit low, but it's actually decent for psychological research. In published papers, you'll often see r-squareds anywhere from .10 to .25 or so. It's very hard to build an extremely accurate model of human behavior.

The `VarImpplot()` function assesses the variable importance for each of the variables in our model, measured by the increase in mean square error when the variable's values are permuted (shuffled). A negative value for variable importance means that the error was higher when using the original predictor variable values than when using the permuted ones (not a good predictor).

As you can see, the lowest variable, neuroticism, is extremely irrelevant to the model. Neuroticism seems to be a bad predictor of arithmetic performance, but be careful — however unlikely, it may be the case that the variance neuroticism explains is better explained by other variables in the model (if correlation is high), so neuroticism may actually show decent prediction in a simpler model with 1-2 variables, including neuroticism, and the DV. In this model, it may even improve model performance through stronger correlation with the IVs than the DV (called a "suppressor" variable, for which you'd need to check semipartial correlations). However, it's likely just a bad predictor.

It looks like the models we've run so far think that math self-concept, sex, math self-efficacy and possibly math anxiety are some of the most important variables. Looking at the graphs, we might guess that there's around 2-5 variables here that predict a significant amount of variance in our DV, so that's probably a good number for a parsimonious model.

Now, to fit our goal of parsimony, we want to cut out the variables that don't predict the DV very well. Though random forest can handle excessive weak predictors (especially if they do

provide a bit of information), we run the risk of adding too much noise and diluting the true signal.

Looking at the variable importance metrics, it seems that there's only one, maybe two predictors that are extremely irrelevant to this specific model. This is probably in part because the effect sizes of most variables are so weak that removing any one variable can significantly reduce predictive power. In other words, the best pure predictive model MAY be a 9-10 variable model, but the best parsimonious model (containing only the most relevant predictors at the cost of not including variables that may provide very minor predictive power) may be somewhere in the 2-5 variable range.

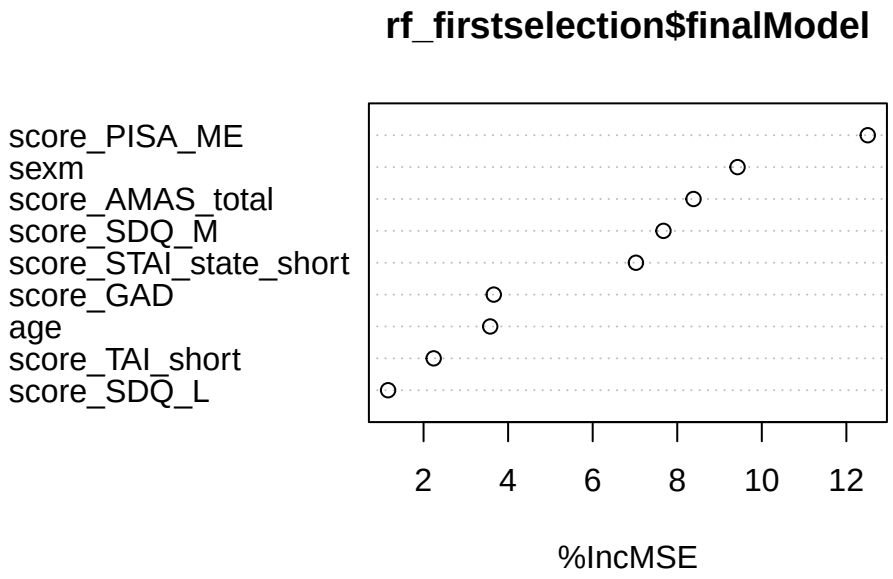
We'll start with a 9 variable model, which we'll create FIRST by cutting `BFI_N`.

First Selection Model

```
set.seed(41)
rf_firstselection <- train(
  sum_arith_perf ~ score_AMAS_total + sex + age +
    score_TAI_short + score_GAD + score_PISA_ME + score_STAI_state_short +
    score_SDQ_M + score_SDQ_L,
  data = scaled_training, method = "rf",
  importance = TRUE, tuneGrid = tunegrid, trControl = control
)
print(rf_firstselection)
## Random Forest
##
## 516 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 465, 466, 464, 464, 464, 463, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     6.140897  0.1449747  4.776474
##   3     6.191097  0.1362896  4.819736
##   4     6.209984  0.1342421  4.833911
##   5     6.223170  0.1333188  4.832254
##   6     6.240649  0.1305266  4.844531
##   7     6.256191  0.1290219  4.857287
##   8     6.257063  0.1292962  4.858542
##
## RMSE was used to select the optimal model using the smallest value.
```

```
## The final value used for the model was mtry = 2.
# we'll look at variable importance again in a minute.
```

```
varImpPlot(rf_firstselection$finalModel, type = 1)
```




```
# type 1 refers to estimates following perturbation.
```

We're seeing a similar pattern. Math self-efficacy and sex seem to be two of the most powerful predictors of math performance in German students in every model, while math self-concept and some math anxiety variables trail not too far behind (the rest seem to hang in the balance). `RMSE` and R squared improved too! But that doesn't always mean the model is strictly better. We'll take a look at how these models predict the test set later.

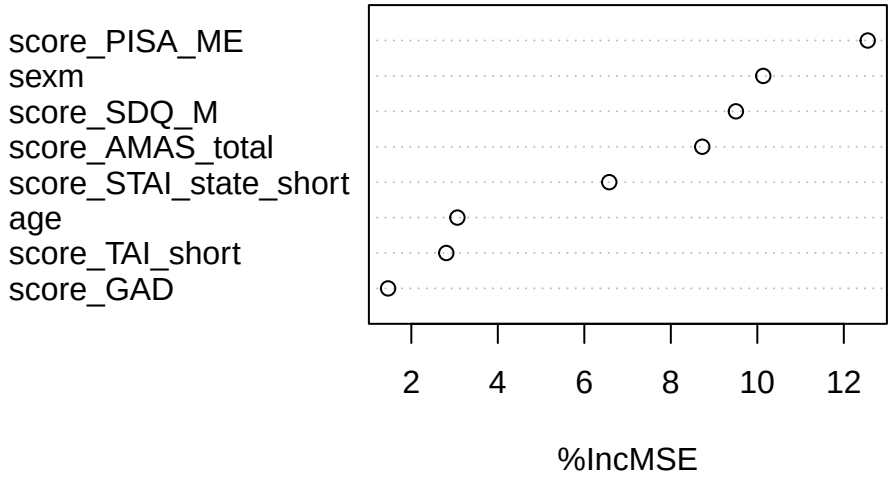
Since `AMAS` measures math anxiety, which might map on to math performance better than general anxiety (I haven't read the literature so take that with a grain of salt), it makes sense that state anxiety and general anxiety are also relatively unimportant predictors. General anxiety doesn't seem to add much value to predicting math performance when math anxiety is accounted for, and state anxiety isn't much better. Test anxiety (from earlier) being a weak predictor is not as intuitive, but it's possible that the most salient aspect here is that the test is on math, not that there's a test itself, so anxiety of tests in general doesn't really have predictive power of performance the same way math anxiety does. Regardless, each also likely covers a lot of the other's correlation.

Now, we're going to try to run more random forest models with a reduced set of predictors since we are starting to identify the most important predictors. Let's see if we can find a more parsimonious model with still good predictive power. Language self-efficacy goes next.

Second Selection Model

```
View(scaled_training)
set.seed(42)
rf_secondselection <- train(
  sum_arith_perf ~ score_AMAS_total + sex + age +
     score_TAI_short + score_STAI_state_short + score_PISA_ME + score_SDQ_M +
    score_GAD,
  data = scaled_training, method = "rf", importance = TRUE,
  tuneGrid = tunegrid, trControl = control
)
print(rf_secondselection)
## Random Forest
##
## 516 samples
##   8 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 465, 464, 465, 465, 464, 465, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     6.125338  0.1498388  4.756372
##   3     6.183191  0.1406933  4.806327
##   4     6.196442  0.1386158  4.810257
##   5     6.222795  0.1343677  4.830288
##   6     6.237778  0.1334048  4.842479
##   7     6.236520  0.1330846  4.844355
##   8     6.241556  0.1341616  4.843493
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
varImpPlot(rf_secondselection$finalModel, type = 1)
```




```
# type 1 refers to estimates following perturbation.
```

Slight reduction in `RMSE` and increase in (training data) R-squared! Importantly, the variable importance metrics are starting to look SLIGHTLY better — the lowest `incMSE%` is around 3%. Good news for our goal of parsimonious modeling. However, I'm not sure if this will continue. Remember that we're looking to explain the most variance with as few variables as needed, so sometimes parsimony is worth slight decreases in predictive accuracy (though you should make sure you're not cutting a seemingly low-predictive power variable that has explanatory importance).

This may be a decent model. The fact that the `$incMSE` is starting to approach reasonable numbers means we're starting to run out of questionable predictors. However, all those measures of anxiety may not significantly improve the model fit or predictive accuracy, and like we said earlier, we likely are looking for a 2-5 variable model to cover as much variance with the fewest variables possible. So, we should keep going.

Let's cut the next least important variable - `score_GAD`.

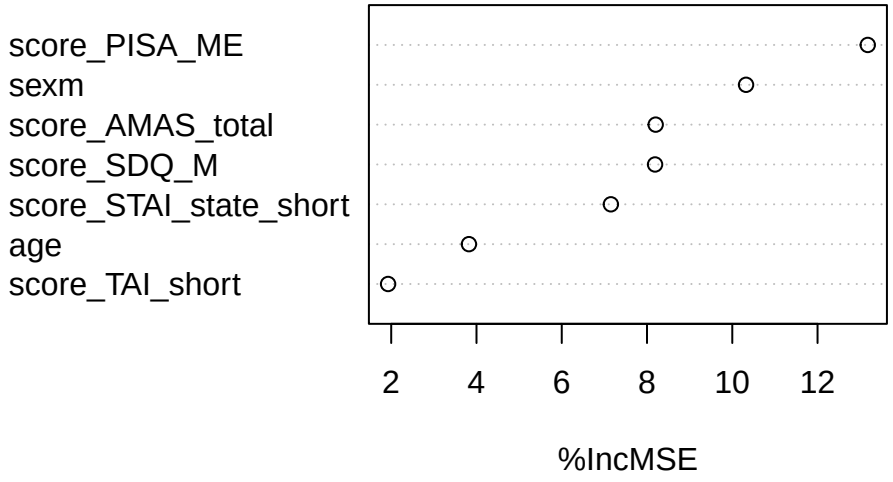
Third Selection Model

```
set.seed(43) #
rf_thirdselection <- train(
  sum_arith_perf ~ score_AMAS_total + sex + age +
    score_STAI_state_short + score_PISA_ME + score_TAI_short + score_SDQ_M,
  data = scaled_training, method = "rf", importance = TRUE,
  tuneGrid = tunegrid, trControl = control
)
```

```
print(rf_thirdselection)
## Random Forest
##
## 516 samples
##   7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 466, 462, 466, 465, 464, 465, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     6.141674  0.1447578  4.750357
##   3     6.191125  0.1376804  4.784046
##   4     6.219143  0.1330534  4.799277
##   5     6.226742  0.1338611  4.803360
##   6     6.248217  0.1296759  4.814006
##   7     6.266422  0.1280605  4.828042
##   8     6.268034  0.1271231  4.824699
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.

# we'll look at variable importance again later.
```

Slight reduction in R squared seems odd, but it's definitely possible that the model got worse after removing GAD. Or, the accuracy increase from retaining GAD may not be large enough to justify recording GAD. We'll see.

```
varImpPlot(rf_thirdselection$finalModel, type = 1)
```




```
# type 1 refers to estimates following perturbation.
```

The `RMSE` did go up this time. This model doesn't seem better than the previous models at the moment. Around this step or the next step is where I would begin to suspect that predictive accuracy is declining fast. Remember that we are currently more interested in, for example, identifying a 5-variable model that explains 15% of the variance over a 7- or 8- variable model that explains say, 17% of the variance, so let's keep going in our pseudo-variable selection procedure.

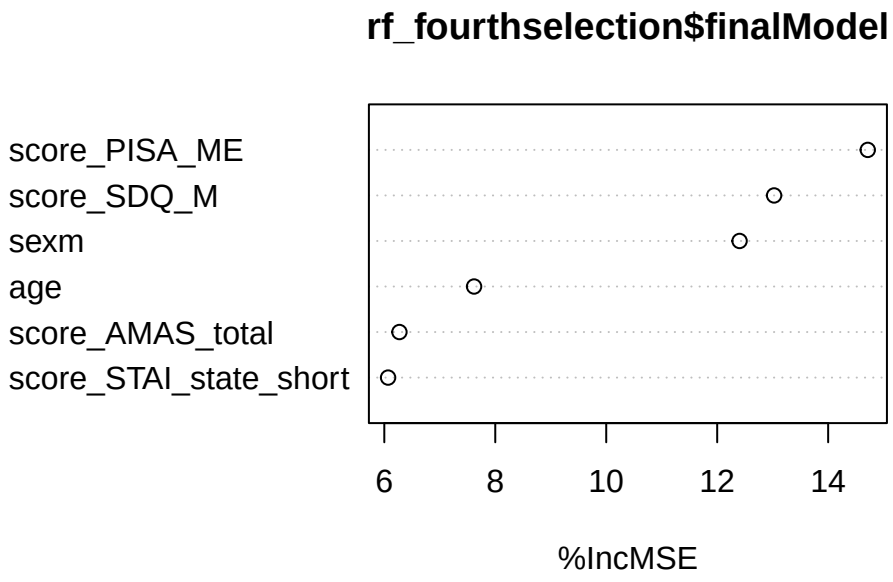
By the way, at this point the highest value of `mtry` being tried (7) is greater than number of predictors (6). We'll start to include a cap line at the beginning of the chunk of code for your convenience (before the set seed, importantly). The code still works without it but it produces a lot of annoying warnings. We're cutting test anxiety next.

Fourth Selection Model

```
tunegrid <- expand.grid(.mtry = c(2:6))
set.seed(44)
rf_fourthselection <- train(
  sum_arith_perf ~ score_AMAS_total + sex + age +
    score_STAI_state_short + score_PISA_ME + score_SDQ_M,
  data = scaled_training,
  method = "rf", importance = TRUE, tuneGrid = tunegrid, trControl = control
)
print(rf_fourthselection)
## Random Forest
```

```
##
## 516 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 465, 463, 465, 464, 465, 463, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     6.157843  0.1444593  4.752846
##   3     6.197738  0.1406118  4.780874
##   4     6.227917  0.1362172  4.809120
##   5     6.241076  0.1365309  4.820602
##   6     6.261390  0.1328107  4.827044
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
# we'll look at variable importance again later.
```

```
varImpPlot(rf_fourthselection$finalModel, type = 1)
```




```
# type 1 refers to estimates following perturbation.
```

Hmm, looks like the anxiety variables aren't really helping us all that much. It is a bit odd that these anxiety variables tend to flip flop with their importance in the importance graphs.

That makes me wonder if there is redundancy in including all of them, even though it appears that at least some of them do add predictive power. The `%IncMSE` for our anxiety variables is relatively high, enough that this sort of model starts to look pretty appealing. However, I am still curious if the most parsimonious model may either have only one anxiety variable (likely math anxiety) or no anxiety variables.

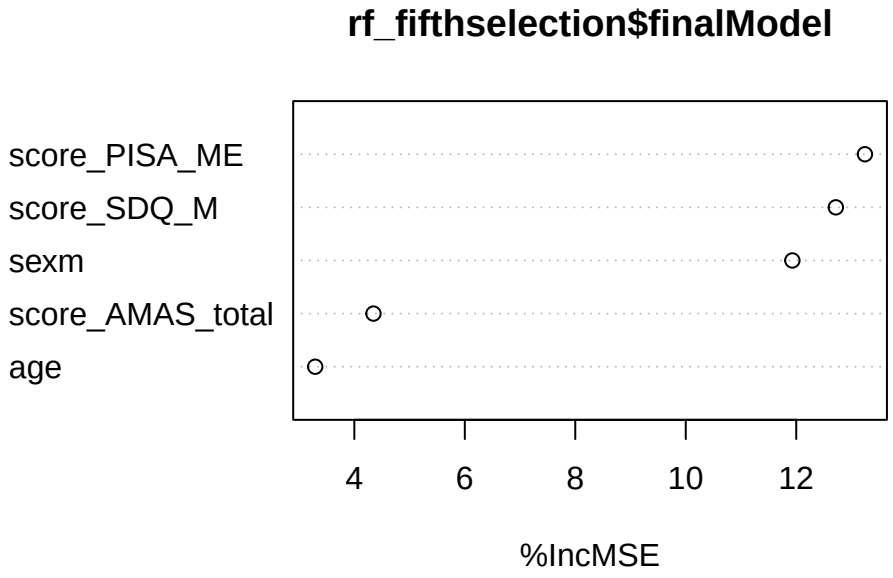
You may have also noticed that `age`, which was once near the bottom of the "importance" graph, continues to move upwards.

Let's cut state anxiety, then.

Fifth Selection Model

```
tunegrid <- expand.grid(.mtry = c(2:5))
set.seed(45)
rf_fifthselection <- train(
  sum_arith_perf ~ sex + age + score_AMAS_total +
    score_PISA_ME + score_SDQ_M,
  data = scaled_training, method = "rf",
  importance = TRUE, tuneGrid = tunegrid, trControl = control
)
print(rf_fifthselection)
## Random Forest
##
## 516 samples
##   5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 465, 464, 465, 465, 466, 465, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   MAE
##   2     6.257386  0.1269729  4.889007
##   3     6.346357  0.1173184  4.967766
##   4     6.385608  0.1147678  4.996582
##   5     6.413973  0.1123130  5.026841
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```
varImpPlot(rf_fifthselection$finalModel, type = 1)
```




```
# type 1 refers to estimates following perturbation.
```

The models continue to worsen. It seems like some of the relatively weak variables that we've removed may actually improve predictive accuracy, but we will check this with test data. However, it does seem that we have identified the key variables as being math self-efficacy and self-concept as well as sex, possibly with math (or state) anxiety and age.

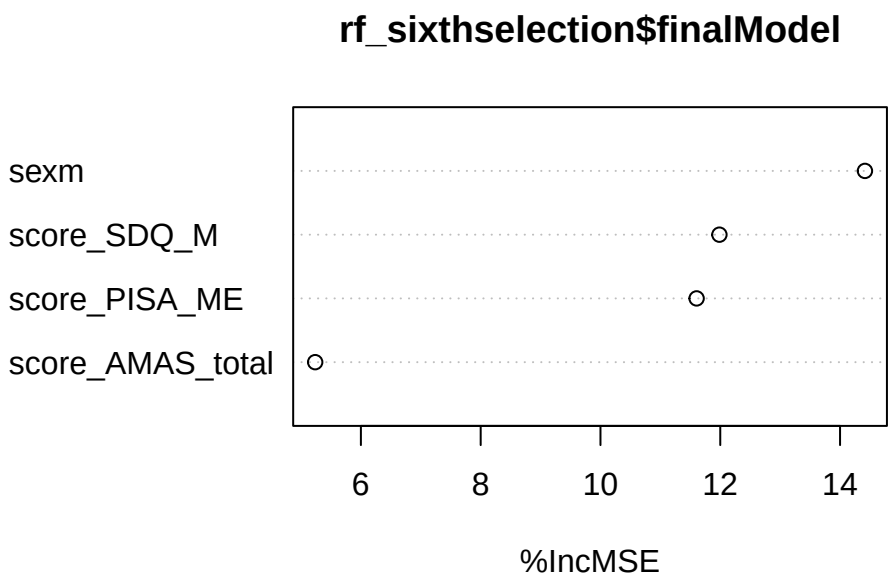
Also, `age` used to be one of the predictors with the lowest `incMSE%` in the full model, but has stuck in the models we are testing. That indicates to me that age may have value in some models, though it seems to generally have little relationship with the dependent variable. I wonder if age is either being overrated by our backwards selection, or if age really does contribute important information in our lower-dimensional models. `Age` likely should only be used in the model if it seems to remain in the best predictive model or if you are convinced of a key interaction with another variable. If this were an actual research project, I would have conducted more exploratory data analysis to check `age`s correlation with other variables as well as evaluated the part correlation in the full model to see if variables such as age were acting as suppressors of key variables.

Sixth Selection Model

```
# just for fun, we'll use the in-formula scale() method to scale data.
tunegrid <- expand.grid(.mtry = c(2:4))
set.seed(46)
rf_sixthselection <- train(
  sum_arith_perf ~ sex + score_PISA_ME + score_SDQ_M +
    score_AMAS_total,
```

```r
  data = scaled_training, method = "rf", importance = TRUE,
  tuneGrid = tunegrid, trControl = control
)
print(rf_sixthselection)
## Random Forest
##
## 516 samples
##   4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 465, 465, 464, 465, 465, 463, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared    MAE
##   2     6.391807  0.10604623  5.018587
##   3     6.519660  0.09707006  5.120164
##   4     6.571421  0.09421629  5.160373
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

```r
varImpPlot(rf_sixthselection$finalModel, type = 1)
```



```
# type 1 refers to estimates following perturbation.
```

The `RMSE` went up again, and by a decent amount. At this point, it is safe to say that the models are not fitting any better. The four variables above do seem to be the most "important" variables in terms of identifying predictors that genuinely correlate with the dependent variable, and math self-efficacy and sex are likely the top two. Of course, this ignores sheer accuracy of prediction. The 4 or 5 variable models (the 5th variable being age) seem to be pretty good parsimonious models if we want to collect more than only 2 variables.

In other words, we've identified 5 variables that can produce a good model that captures a good amount of the variability in scores. This model may be the best combination of parsimony and predictive power (it may even be flat out the best predictive model), but this model is likely not the most parsimonious (that seems to be a model with only a couple predictors, as we've indicated before). There is also a chance it is not the most predictive (which we'll test in the Random Forest Test Predictions addendum).

In my second addendum to this blog about automated feature selection methods, we will devise a "parsimonious" two-predictor model. In my fourth addendum to this blog, entitled "Random Forest Test Prediction", we will see that its predictive accuracy rivals that of some bigger models we created above.

Let's look at some automated variable selection procedures in the next addendum, then we'll test all our predictions in a fourth addendum.