Final screening of traits after removing the influence of sampling date

Jody Daniel

2020-11-06

Contents

Background

We plan to build a general linear model that predicts tree growth rate. But, we have lots of traits as predictors - environmental and physical. Previously, we built a random forest model that is an attempt ascertain which traits are of greatest importance. Following, we used extracted the residuals of a model evaluating the influence of sampling date on traits - this was to remove the influence of sampling date. This can change the model's predictions on which traits are of greatest importance, and so I think it is best to re-run the random forest model with these new data - of course using basal growth rate.

```
library(vip)
library(knitr)
library(kableExtra)
library(tidyverse)
library(dplyr)
library(here)
library(skimr)
library(reshape2)
library(tidymodels)
library(qdapTools)
library(rsample)
library(corrr)
library(broom)
library(vegan)
library(extrafont)
library(viridis)
library(car)
source(here("scripts/archive/1. functions.R"))
theme_set(theme_special())
```

Preparing data for random forest model

Assuming that the first 10000 rows describes the data well, I can use col_types = cols(). If not, I will need to specify the class of each column.

```
# prior error suggests that there are single quotation marks used to parse numbers
# I specify that we should read those as a thousand separator
rgr_msh_residuals_julian_df <- read_csv(here("data/rgr_msh_residuals_julian_df.csv"),
                        guess_max = 10000,
                        col_types = cols())
rgr_msh_residuals_julian_df <- column_to_rownames(rgr_msh_residuals_julian_df,
                                                  var = "SampleID")
```

```
# drop variables with more than 2 missing values -
# to compare across models, we can't have any missing rows
# first, I will remove problem variables  - next I will remove any rows (from the raw data)
# that are have missing values
missing.rf.ii <- names(which(sapply(colnames(rgr_msh_residuals_julian_df),
                                    function (x)
                                      {sum(is.na(rgr_msh_residuals_julian_df[,x]))/nrow(rgr_msh_residuals_
rgr_msh_residuals_julian_df_na_omit <- rgr_msh_residuals_julian_df[,missing.rf.ii]

# need a training and test set assess the performance of the model
# a 70:30 split is typical
# first, I will work with the imputed data
set.seed(634)
split_train_test <-
  initial_split(
    data = rgr_msh_residuals_julian_df_na_omit,
    prop = 0.70)
train_rgr_residuals <- split_train_test %>% training()
test_rgr_residuals  <- split_train_test %>% testing()
```

Building the random forest modelS

The first step in building the random forest models is to tuning.

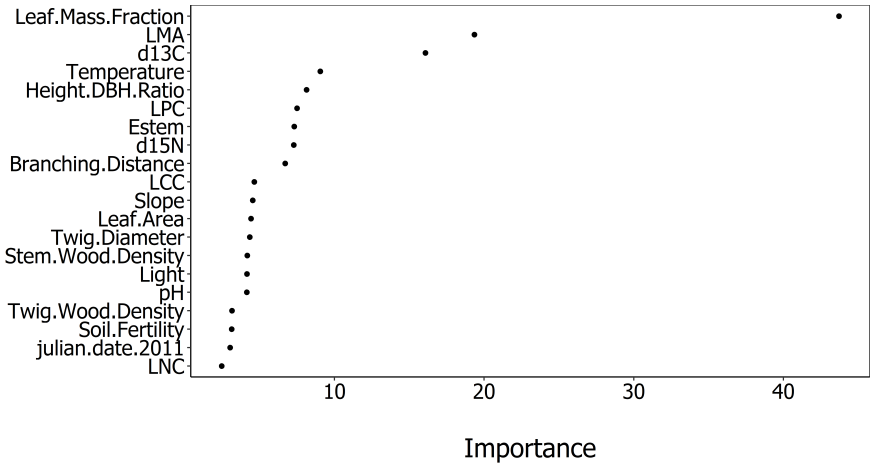
```
# first, we must tune the random forest model. i will use a 80:20 split.
# i will tune for the number of predictors that will be randomly pooled/available
# for splitting at each node (mtry)
# i will also tune min_n, which is minimum number of observation required to split a node further

# because there are two predictors with 2 different versions (imputed vs raw)
# I will need to validate/build 4 models

cores <- parallel::detectCores()
randomforest_mod <-
  rand_forest(mtry = tune(), min_n = tune(), trees = 1000) %>%
  set_engine("ranger", num.threads = cores) %>%
  set_mode("regression")

# need split the training data to tune the model (validation)
set.seed(345)
```

```
val_set_bai_residuals <- validation_split(train_rgr_residuals,
                            prop = 0.80)

#### making a recipe for what goes into the models
# imputed data
randomforest_recipe_bai_residuals <-
  recipe(BAI_GR ~ ., data =  train_rgr_residuals)


# the workflow helps in given steps in the tuning process
# imputed data
randomforest_workflow_bai_residuals <-
  workflow() %>%
  add_model(randomforest_mod) %>%
  add_recipe(randomforest_recipe_bai_residuals)

# now, we can set the terms for what we use to validate
# imputed data
randomforest_res_bai_residuals <-
  randomforest_workflow_bai_residuals %>%
  tune_grid(val_set_bai_residuals,
            grid = 10,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(rmse))

# now, we can look at the model performance
# Basal Area Growth Rate
randomforest_res_bai_residuals %>%
  show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1    21    11 rmse    standard    1.03     1      NA Model08
## 2     9     7 rmse    standard    1.04     1      NA Model05
## 3    13    14 rmse    standard    1.05     1      NA Model02
## 4    15    24 rmse    standard    1.06     1      NA Model07
## 5     6    19 rmse    standard    1.06     1      NA Model01
```

```
random_forest_regressor_bai_residuals <-
  rand_forest(mtry = 21, min_n = 11, trees = 1000) %>%
  set_engine("ranger", num.threads = cores, importance = "impurity") %>%
  set_mode("regression") %>%
  fit(BAI_GR ~ ., data = train_rgr_residuals)
```

```
## pdf
##   2
```



These do not look vastly different from the orginal model - though sampling date appears to increase in importance.

```
# now, I will compare the RSME between the training and test data
# I am hoping that there is not really much of a difference between the two
# if there is (much higher for the test)
# I will be concerned about overfitting - meaning the model does not generalize well
predictions_rf_bai_train_residuals <-
  random_forest_regressor_bai_residuals %>%
  predict(new_data = train_rgr_residuals) %>%
  bind_cols(train_rgr_residuals)

predictions_rf_bai_test_residuals <-
  random_forest_regressor_bai_residuals %>%
  predict(new_data = test_rgr_residuals) %>%
  bind_cols(test_rgr_residuals)



metrics(predictions_rf_bai_train_residuals, truth = "BAI_GR", estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.488
## 2 rsq     standard       0.851
## 3 mae     standard       0.331
```

```
metrics(predictions_rf_bai_test_residuals, truth = "BAI_GR", estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      1.06
## 2 rsq     standard      0.0622
## 3 mae     standard      0.743
```

```
# evidence of overfitting
```

Conclusion

There is overfitting for this model, and the rank of variable importance did not really change.