Ranking traits by their importance to growth rate

Jody Daniel

2020-11-03

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. Based on Julie's work, we know that the physical traits are poorly summarized assuming a linear, orthogonal relationship. For the environmental traits, her work suggests that a PCA does a good job in reducing features. As such, in jupyter notebook, we created a new data set that includes these environmental traits as PCs, naming each axis based on the dominant vector relationships.

There was alot of missing data, so we used predictive mean matching to fill in gaps. Based on preliminary analyses, there isn't much of a difference between the imputed and raw data. So, we can proceed with the imputed data for these analyses.

Before building the GLM, we need to pre-determine which of the traits are most important in predicting growth rate. It is not advisable that we throw a bunch of predictors at the GLM and pull out the important ones. Its best we use another modelling framework for feature reduction and use the results to inform which predictors will go into the GLM. In the past analyses, I used xgboost - here I will use random forest in tidy models.

1. Prepare data for modelling using recipe - training and test

2. Build random forest model
3. Assess performance and determine which features are most important
4. Compare performance across the 3 growth rate measures

```
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 <- read_csv(here("data/RGR_MSH_PCA.csv"),
                        guess_max = 10000,
                        col_types = cols())

rgr_msh_na <- read_csv(here("data/RGR_MSH_PCA_NA.csv"),
                        guess_max = 10000,
                        col_types = cols())

# there are NAs in the predictors - let's drop those
rgr_msh <- rgr_msh[-(which(is.na(rgr_msh$BAI_GR))),]
rgr_msh$BIO_GR <- as.numeric(rgr_msh$BIO_GR)
rgr_msh_na <- rgr_msh_na[-(which(is.na(rgr_msh_na$BAI_GR))),]

# now, let's remove columns that are either too correlated are would not be useful
labels_rgr_msh <- read_csv(here("data/labels.csv"),
                        guess_max = 10000,
                        col_types = cols())

# save for later use
write_csv(rgr_msh,here("data/RGR_MSH_Imputed-RF.csv"))
write_csv(rgr_msh_na,here("data/RGR_MSH_PCA_Raw-RF.csv"))

# remove sampleID
rgr_msh <- rgr_msh[,-1]
rgr_msh_na <- rgr_msh_na[,-1]

# what do these data look like?
kable(skim(rgr_msh),"latex", booktabs = T) %>%
   kable_styling(latex_options="scale_down")
```

skim_type	skim_variable	n_missing	complete_rate	numeric.mean	numeric.sd	numeric.p0	numeric.p25	numeric.p50	numeric.p75	numeric.p100	numeric.hist
numeric	BAI_GR	0	1	1.1302322	1.1392822	0.0063889	0.3426846	0.7728631	1.4907379	6.1606526	<U+2587><U+2582><U+2581><U+2581><U+2581>
numeric	BIO_GR	0	1	22.8932768	198.8887403	-296.1120266	0.1633659	0.3219269	0.6956222	3130.5951007	<U+2587><U+2581><U+2581><U+2581><U+2581>
numeric	Height.DBH.Ratio	0	1	149.0361995	41.1440109	66.2686567	120.0000000	144.3011135	173.4627475	292.0000000	<U+2583><U+2587><U+2585><U+2582><U+2581>
numeric	Estem	0	1	4059.0536773	3348.4486979	360.8974410	1679.6504455	3142.8020830	5424.2941017	21146.1018900	<U+2587><U+2583><U+2581><U+2581><U+2581>
numeric	Branching.Distance	0	1	23.7762592	21.0002663	5.1333333	11.1027778	16.1714286	27.1500000	100.0000000	<U+2587><U+2582><U+2581><U+2581><U+2581>
numeric	Twig.Diameter	0	1	5.5824339	1.8195375	3.0250000	4.4550000	5.1450000	6.2550000	15.5950000	<U+2587><U+2583><U+2581><U+2581><U+2581>
numeric	Twig.Wood.Density	0	1	0.5019400	0.0768801	0.2697191	0.4563929	0.5109263	0.5538794	0.6641150	<U+2581><U+2582><U+2586><U+2587><U+2582>
numeric	Stem.Wood.Density	0	1	0.6172857	0.1321204	0.2648281	0.5190918	0.6063860	0.7290613	0.8974172	<U+2581><U+2585><U+2587><U+2587><U+2583>
numeric	Leaf.Mass.Fraction	0	1	1.2946970	0.6370553	0.2367848	0.8117639	1.1558553	1.6548959	3.5336659	<U+2586><U+2587><U+2583><U+2582><U+2581>
numeric	Leaf.Area	0	1	58.7264402	76.9725392	3.7444000	18.6422274	29.9528720	61.6420000	474.4577273	<U+2587><U+2581><U+2581><U+2581><U+2581>
numeric	LMA	0	1	52.5833135	24.2892878	17.4592553	32.9579384	45.4152885	67.7729873	134.2225368	<U+2587><U+2586><U+2583><U+2582><U+2581>
numeric	LCC	0	1	47.5065935	2.3816233	41.7288000	46.0629500	47.6380166	49.1964500	52.5132000	<U+2582><U+2583><U+2587><U+2586><U+2582>
numeric	LNC	0	1	2.6902394	0.5043952	1.4219000	2.3411750	2.6766500	3.0043000	4.1779000	<U+2581><U+2586><U+2587><U+2583><U+2581>
numeric	LPC	0	1	15.3850447	4.8452937	5.5412378	12.0673999	14.4771340	17.8375977	41.3020907	<U+2585><U+2587><U+2582><U+2581><U+2581>
numeric	d15N	0	1	-2.2489360	2.8294941	-9.0360275	-4.1134717	-2.2108013	-0.7612524	16.1917920	<U+2583><U+2587><U+2581><U+2581><U+2581>
numeric	t.b2	0	1	0.0249674	0.0090102	0.0007625	0.0192432	0.0233802	0.0301874	0.0580334	<U+2581><U+2587><U+2586><U+2582><U+2581>
numeric	Ks	0	1	115.1504687	138.9980985	10.9835978	39.2587720	57.8132499	122.2573950	985.8303049	<U+2587><U+2581><U+2581><U+2581><U+2581>
numeric	Ktwig	0	1	1158.5983036	896.0473055	90.7266105	625.9823218	902.8251040	1365.8951397	7135.0900750	<U+2587><U+2582><U+2581><U+2581><U+2581>
numeric	Huber.Value	0	1	0.1447076	0.6339975	0.0014640	0.0056621	0.0083990	0.0171957	5.9961641	<U+2587><U+2581><U+2581><U+2581><U+2581>
numeric	X.Lum	0	1	0.1321978	0.0437752	0.0355744	0.1003775	0.1287952	0.1604020	0.2781733	<U+2582><U+2587><U+2587><U+2582><U+2581>
numeric	VD	0	1	10439.0935511	9671.5466809	1988.8913790	5213.5008400	6814.0191070	10236.2653499	53249.4433200	<U+2587><U+2581><U+2581><U+2581><U+2581>
numeric	X.Sapwood	0	1	0.7647659	0.2816991	0.0089190	0.7193222	0.9002788	0.9581139	0.9958401	<U+2581><U+2581><U+2581><U+2582><U+2587>
numeric	d13C	0	1	-30.2261174	1.4707112	-33.8285092	-31.3604580	-30.4448735	-29.3061227	-25.8225092	<U+2582><U+2587><U+2587><U+2583><U+2581>
numeric	Root.Wood.Density	0	1	0.5783404	0.0943932	0.2954545	0.5161291	0.5859933	0.6437516	0.8468866	<U+2581><U+2583><U+2587><U+2585><U+2581>
numeric	Twig.branching.angle	0	1	54.0728195	16.0488974	0.0000000	47.3503625	55.0366345	64.2860058	102.8261500	<U+2581><U+2581><U+2587><U+2585><U+2581>
numeric	Hmax	0	1	20.1373057	9.2429999	5.0000000	12.0000000	25.0000000	25.0000000	35.0000000	<U+2586><U+2585><U+2582><U+2587><U+2585>
numeric	Shade.Tolerance	0	1	2.7835544	1.1094286	1.2100000	1.7000000	2.5900000	3.5600000	4.7600000	<U+2587><U+2586><U+2586><U+2585><U+2583>
numeric	Drought.Tolerance	0	1	2.4691813	0.6821171	1.5000000	1.7875000	2.3800000	2.8800000	4.0000000	<U+2587><U+2585><U+2586><U+2582><U+2582>
numeric	WaterLogging.Tolerance	0	1	1.7351036	0.7514112	1.0000000	1.0700000	1.3830000	2.4600000	3.3700000	<U+2587><U+2582><U+2582><U+2582><U+2582>
numeric	Soil.Fertility	0	1	-0.0042030	1.0058587	-3.5206805	-0.4916204	0.0991271	0.7626882	1.5478551	<U+2581><U+2581><U+2583><U+2587><U+2585>
numeric	Light	0	1	0.0016454	1.0014968	-2.5262279	-0.6070097	-0.0623770	0.7929413	2.5486147	<U+2581><U+2585><U+2587><U+2585><U+2582>
numeric	Temperature	0	1	-0.0012026	1.0060925	-3.2374882	-0.6978396	0.1481046	0.6663379	2.5475439	<U+2581><U+2583><U+2587><U+2587><U+2582>
numeric	pH	0	1	0.0000977	1.0049360	-3.1675401	-0.6580243	0.1014687	0.6471712	2.7080906	<U+2581><U+2583><U+2587><U+2587><U+2581>
numeric	Slope	0	1	0.0089926	1.0033873	-3.6341613	-0.5814711	0.0382017	0.5870221	3.6108350	<U+2581><U+2582><U+2587><U+2583><U+2581>

```
# skim(rgr_msh) - for markdown visualization

kable(skim(rgr_msh_na),"latex", booktabs = T) %>%
   kable_styling(latex_options="scale_down")
```

skim_type	skim_variable	n_missing	complete_rate	numeric.mean	numeric.sd	numeric.p0	numeric.p25	numeric.p50	numeric.p75	numeric.p100	numeric.hist
numeric	BAI_GR	0	1.0000000	1.0454229	1.0459687	0.0228348	0.3232933	0.7007034	1.3573804	5.6432654	<U+2587><U+2582><U+2581><U+2581><U+2581>
numeric	BIO_GR	0	1.0000000	6137.7830707	6184.9523580	76.6072356	1846.0686532	4341.4419884	7877.0742166	31987.3832799	<U+2587><U+2582><U+2581><U+2581><U+2581>
numeric	Height.DBH.Ratio	0	1.0000000	149.5779741	40.8828732	66.2686567	123.2000000	144.6153846	173.6111111	292.0000000	<U+2583><U+2587><U+2585><U+2582><U+2581>
numeric	Estem	0	1.0000000	4117.4235503	3377.6764997	387.9322453	1651.9923660	3162.4877140	5415.1010630	19897.4481300	<U+2587><U+2583><U+2581><U+2581><U+2581>
numeric	Branching.Distance	0	1.0000000	23.4759018	20.5649430	5.1333333	11.1666667	16.2500000	27.4000000	100.0000000	<U+2587><U+2582><U+2581><U+2581><U+2581>
numeric	Twig.Diameter	0	1.0000000	5.3574693	1.3313756	3.1000000	4.4750000	5.0800000	6.0600000	10.5250000	<U+2586><U+2587><U+2583><U+2581><U+2581>
numeric	Twig.Wood.Density	0	1.0000000	0.4999480	0.0693095	0.2881215	0.4570692	0.5080152	0.5419989	0.6569095	<U+2581><U+2582><U+2587><U+2587><U+2582>
numeric	Stem.Wood.Density	0	1.0000000	0.6156938	0.1272102	0.2781708	0.5256410	0.6031154	0.7233333	0.8890028	<U+2581><U+2585><U+2587><U+2587><U+2583>
numeric	Leaf.Mass.Fraction	0	1.0000000	1.2528141	0.5755178	0.2367848	0.8117502	1.1552226	1.6226601	3.1026451	<U+2585><U+2587><U+2585><U+2582><U+2581>
numeric	Leaf.Area	0	1.0000000	50.5151064	62.6200260	6.4111940	18.0660606	29.2886667	55.3671429	464.5518182	<U+2587><U+2581><U+2581><U+2581><U+2581>
numeric	LMA	0	1.0000000	51.6559695	24.4445179	17.4592553	32.3259757	44.4803207	64.3937761	134.2225368	<U+2587><U+2586><U+2582><U+2582><U+2581>
numeric	LCC	0	1.0000000	47.3632831	2.4085609	41.7288000	45.9704000	47.5595000	49.0191000	52.5132000	<U+2582><U+2585><U+2587><U+2586><U+2582>
numeric	LNC	0	1.0000000	2.6986188	0.5236310	1.4219000	2.3429000	2.6755000	3.0221000	4.1779000	<U+2581><U+2586><U+2587><U+2583><U+2581>
numeric	LPC	0	1.0000000	15.5720210	4.8455409	6.3720369	12.1726191	14.7828028	17.9781154	36.1435930	<U+2585><U+2587><U+2583><U+2581><U+2581>
numeric	d15N	0	1.0000000	-2.4534068	2.8469652	-9.0360275	-4.2774503	-2.5652686	-1.0526885	16.1917920	<U+2583><U+2587><U+2581><U+2581><U+2581>
numeric	t.b2	1	0.9961686	0.0252438	0.0087438	0.0033228	0.0193951	0.0236662	0.0305834	0.0580334	<U+2581><U+2587><U+2585><U+2582><U+2581>
numeric	Ks	1	0.9961686	99.2872204	110.7455970	15.5038665	39.1127792	56.0582127	102.0887041	654.6307655	<U+2587><U+2581><U+2581><U+2581><U+2581>
numeric	Ktwig	1	0.9961686	1049.8124739	672.5136471	90.7266105	606.7252595	902.8251040	1274.5754935	4529.9108300	<U+2587><U+2585><U+2581><U+2581><U+2581>
numeric	Huber.Value	1	0.9961686	0.0252333	0.0600954	0.0014640	0.0056902	0.0081820	0.0148158	0.4757235	<U+2587><U+2581><U+2581><U+2581><U+2581>
numeric	X.Lum	1	0.9961686	0.1325866	0.0449628	0.0428215	0.0998291	0.1270350	0.1618095	0.2781733	<U+2583><U+2587><U+2586><U+2582><U+2581>
numeric	VD	1	0.9961686	9433.0478997	8290.4867215	1988.8913790	5225.8313807	6631.2128815	9380.6818992	53001.6502400	<U+2587><U+2581><U+2581><U+2581><U+2581>
numeric	X.Sapwood	1	0.9961686	0.7758689	0.2779943	0.0089190	0.7421005	0.9000072	0.9580286	0.9958401	<U+2581><U+2581><U+2581><U+2582><U+2587>
numeric	d13C	0	1.0000000	-30.2604413	1.4300913	-33.1030037	-31.4328102	-30.4244853	-29.4120506	-25.8225092	<U+2582><U+2587><U+2587><U+2585><U+2582>
numeric	Root.Wood.Density	94	0.6398467	0.5729642	0.0958391	0.2975610	0.5090564	0.5730905	0.6451332	0.8468866	<U+2581><U+2586><U+2587><U+2586><U+2581>
numeric	Twig.branching.angle	63	0.7586207	54.5597571	16.4889420	0.0000000	47.9907688	55.4161000	65.1914906	102.8261500	<U+2581><U+2581><U+2587><U+2585><U+2581>
numeric	Hmax	0	1.0000000	20.8620690	8.9419139	5.0000000	12.0000000	25.0000000	25.0000000	35.0000000	<U+2585><U+2585><U+2582><U+2587><U+2585>
numeric	Shade.Tolerance	22	0.9157088	2.8244351	1.1280699	1.2100000	1.7000000	2.7500000	3.5600000	4.7600000	<U+2587><U+2586><U+2586><U+2585><U+2586>
numeric	Drought.Tolerance	22	0.9157088	2.4044770	0.6451334	1.5000000	1.7700000	2.3400000	2.8800000	4.0000000	<U+2587><U+2583><U+2585><U+2582><U+2581>
numeric	WaterLogging.Tolerance	22	0.9157088	1.8238494	0.7695711	1.0000000	1.1100000	1.5000000	2.5000000	3.3700000	<U+2587><U+2582><U+2582><U+2583><U+2582>
numeric	Soil.Fertility	0	1.0000000	-0.0049155	0.9987448	-2.8679123	-0.6214113	0.0731471	0.8799620	1.4154004	<U+2581><U+2582><U+2585><U+2587><U+2587>
numeric	Light	0	1.0000000	-0.0044356	0.9993356	-2.4381007	-0.7536513	-0.0543415	0.8098024	2.4469225	<U+2582><U+2587><U+2587><U+2586><U+2582>
numeric	Temperature	0	1.0000000	-0.0007245	1.0018523	-2.5157203	-0.7799819	0.0795782	0.6330801	3.0066055	<U+2581><U+2585><U+2587><U+2583><U+2581>
numeric	pH	0	1.0000000	0.0051742	0.9984012	-2.0427819	-0.7837677	0.0523284	0.7008817	2.8023401	<U+2583><U+2586><U+2587><U+2583><U+2581>
numeric	Slope	0	1.0000000	-0.0029185	1.0008026	-5.6374272	-0.3699992	0.1091224	0.5707877	2.7637757	<U+2581><U+2581><U+2582><U+2587><U+2581>

```
# skim(rgr_msh_na) - for markdown visualization

# 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 <- names(which(sapply(colnames(rgr_msh_na), function (x) {sum(is.na(rgr_msh_na[,x]))/nrow(rgr
rgr_msh <- rgr_msh[,missing.rf]
rgr_msh_na <- na.omit(rgr_msh_na[,missing.rf])

# need a training and test set assess the performance of the model
```

```r
# 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,
    prop = 0.70)
train_rgr_msh <- split_train_test %>% training()
test_dta_msh  <- split_train_test %>% testing()

# now, I will work with the raw data
set.seed(634)
split_train_test_na <-
  initial_split(
    data = rgr_msh_na,
    prop = 0.70)
train_rgr_msh_na <- split_train_test_na %>% training()
test_rgr_msh_na  <- split_train_test_na %>% testing()
```

Building the random forest modelS

The first step in building the random forest models is to tuning. There are four models to tune, because we have two datasets and two predictors. The predictors are basal area growth rate and biomass growth rate. Also, we have a raw dataset, and another that was built from predictive mean matching - the past examinations suggest they are about the same, but I'd like to ensure that the converge on model predictions.

```r
# 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 <- validation_split(train_rgr_msh[,-(which(names(train_rgr_msh)=="BIO_GR"))],
                                prop = 0.80)
val_set_bio <- validation_split(train_rgr_msh[,-(which(names(train_rgr_msh)=="BAI_GR"))],
                                prop = 0.80)

val_set_bai_na <- validation_split(train_rgr_msh_na[,-(which(names(train_rgr_msh_na)=="BIO_GR"))],
                                prop = 0.80)
val_set_bio_na <- validation_split(train_rgr_msh_na[,-(which(names(train_rgr_msh_na)=="BAI_GR"))],
                                prop = 0.80)
```

```
#### making a recipe for what goes into each of the four models
# imputed data
randomforest_recipe_bai <-
  recipe(BAI_GR ~ ., data = train_rgr_msh[,-(which(names(train_rgr_msh)=="BIO_GR"))])
randomforest_recipe_bio <-
  recipe(BIO_GR ~ ., data = train_rgr_msh[,-(which(names(train_rgr_msh)=="BAI_GR"))])

randomforest_recipe_bai_na <-
  recipe(BAI_GR ~ ., data = train_rgr_msh_na[,-(which(names(train_rgr_msh_na)=="BIO_GR"))])
randomforest_recipe_bio_na <-
  recipe(BIO_GR ~ ., data = train_rgr_msh_na[,-(which(names(train_rgr_msh_na)=="BAI_GR"))])

# the workflow helps in given steps in the tuning process
# imputed data
randomforest_workflow_bai <-
  workflow() %>%
  add_model(randomforest_mod) %>%
  add_recipe(randomforest_recipe_bai)
randomforest_workflow_bio <-
  workflow() %>%
  add_model(randomforest_mod) %>%
  add_recipe(randomforest_recipe_bio)

randomforest_workflow_bai_na <-
  workflow() %>%
  add_model(randomforest_mod) %>%
  add_recipe(randomforest_recipe_bai_na)
randomforest_workflow_bio_na <-
  workflow() %>%
  add_model(randomforest_mod) %>%
  add_recipe(randomforest_recipe_bio_na)

# now, we can set the terms for what we use to validate
# imputed data
randomforest_res_bai <-
  randomforest_workflow_bai %>%
  tune_grid(val_set_bai,
            grid = 10,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(rmse))
randomforest_res_bio <-
  randomforest_workflow_bio %>%
  tune_grid(val_set_bio,
            grid = 10,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(rmse))

randomforest_res_bai_na <-
  randomforest_workflow_bai_na %>%
  tune_grid(val_set_bai_na,
            grid = 10,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(rmse))
```

```
randomforest_res_bio_na <-
  randomforest_workflow_bio_na %>%
  tune_grid(val_set_bio_na,
            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 %>%
  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    24    18 rmse    standard   0.823     1      NA Model05
## 2    18     6 rmse    standard   0.825     1      NA Model10
## 3    20    11 rmse    standard   0.828     1      NA Model09
## 4    25    34 rmse    standard   0.828     1      NA Model07
## 5    15    21 rmse    standard   0.839     1      NA Model06
```

```
randomforest_res_bai_na %>%
  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    24    29 rmse    standard    1.09     1      NA Model10
## 2    12    16 rmse    standard    1.09     1      NA Model06
## 3    25    10 rmse    standard    1.09     1      NA Model07
## 4    17     3 rmse    standard    1.10     1      NA Model08
## 5    19     7 rmse    standard    1.10     1      NA Model01
```

```
# Biomass Growth Rate
randomforest_res_bio %>%
  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    26    20 rmse    standard    369.     1      NA Model07
## 2    23    30 rmse    standard    374.     1      NA Model04
## 3    21    27 rmse    standard    382.     1      NA Model02
## 4    17     5 rmse    standard    383.     1      NA Model08
## 5    17    23 rmse    standard    385.     1      NA Model01
```

```
randomforest_res_bio_na %>%
  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    17     4 rmse    standard   4244.     1      NA Model09
## 2    23    15 rmse    standard   4270.     1      NA Model08
## 3    27    21 rmse    standard   4287.     1      NA Model07
## 4    15    12 rmse    standard   4318.     1      NA Model06
## 5    21    27 rmse    standard   4337.     1      NA Model04
```

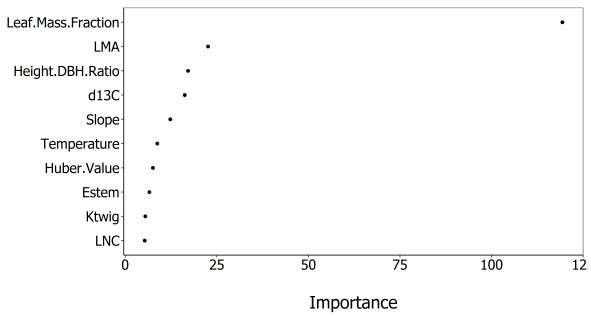
```
random_forest_regressor_bai <-
  rand_forest(mtry = 24, min_n = 18, trees = 1000) %>%
  set_engine("ranger", num.threads = cores, importance = "impurity") %>%
  set_mode("regression") %>%
  fit(BAI_GR ~ ., data = train_rgr_msh[,-(which(names(train_rgr_msh)=="BIO_GR"))])

random_forest_regressor_bai_na <-
  rand_forest(mtry = 24, min_n = 29, trees = 1000) %>%
  set_engine("ranger", num.threads = cores, importance = "impurity") %>%
  set_mode("regression") %>%
  fit(BAI_GR ~ ., data = train_rgr_msh_na[,-(which(names(train_rgr_msh_na)=="BIO_GR"))])


random_forest_regressor_bio <-
  rand_forest(mtry = 26, min_n = 20, trees = 1000) %>%
  set_engine("ranger", num.threads = cores, importance = "impurity") %>%
  set_mode("regression") %>%
  fit(BIO_GR ~ ., data = train_rgr_msh[,-(which(names(train_rgr_msh)=="BAI_GR"))])
random_forest_regressor_bio_na <-
  rand_forest(mtry = 17, min_n = 4, trees = 1000) %>%
  set_engine("ranger", num.threads = cores, importance = "impurity") %>%
  set_mode("regression") %>%
  fit(BIO_GR ~ ., data = train_rgr_msh_na[,-(which(names(train_rgr_msh_na)=="BAI_GR"))])
```

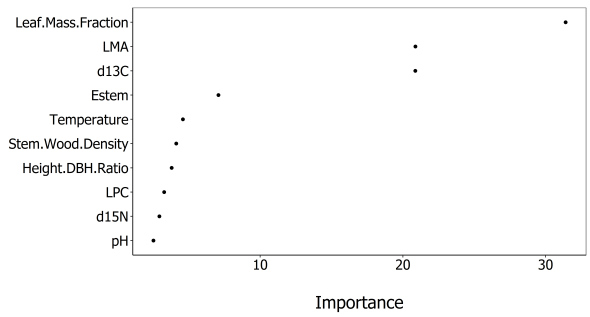
```
## pdf
##   2
```

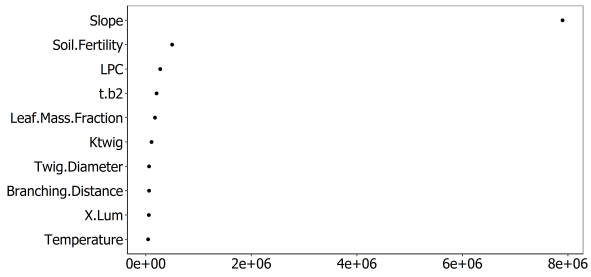
```
## pdf
##   2
```

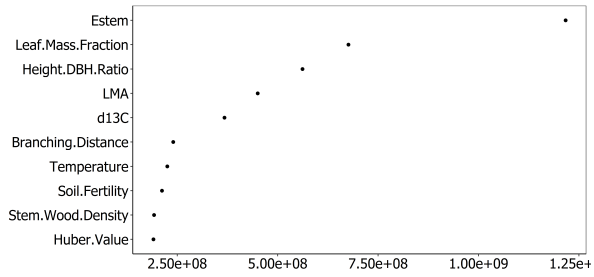
```
## pdf
##   2
```

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









After some thought, we realized that using imputed data to measure growth rate was not a good move because measures like tree age/height may be incorrect. Also, the basal growth rate is a better go because: 1. Julie measured it directly, and so there is less "noise" 2. It does not require us to make assumptions about the tree growth form (cylindrical vs cone)

Next, I will change for the error rates from the basal growth rate model on the raw data.

```
# 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 <-
  random_forest_regressor_bai %>%
  predict(new_data = train_rgr_msh_na[,-(which(names(train_rgr_msh_na)=="BIO_GR"))]) %>%
  bind_cols(train_rgr_msh_na)
predictions_rf_bai_test <-
  random_forest_regressor_bai %>%
  predict(new_data = test_rgr_msh_na[,-(which(names(test_rgr_msh_na)=="BIO_GR"))]) %>%
  bind_cols(test_rgr_msh_na)


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

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.721
## 2 rsq     standard       0.547
## 3 mae     standard       0.468
```

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

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.783
## 2 rsq     standard       0.505
## 3 mae     standard       0.501
```

```
# they are pretty similar (RSME) - so I am not concerned about overfitting
```

Conclusion

We will use the basal growth rate as the response variable, going forward, for the glm. We will not use the imputed data. One issue is that some of the variables that went into estimating biomass growth rate had missing values, so the imputed data gave results that did not make much biological sense. Also, biomass growth rate makes a bunch of assumptions about each individual tree, which we are not confident is applicable to all trees in the dataset (different species with different ages).

There's a concern that sampling date has affacted some of the meausred trait values - to deal with this, we will run a linear regression on each trait vs sampling date. If there is a signicabt effect, we will only use the residuals vs the raw data.