A report generated from a pure R script

Load the data, plot the data!

```
library(gridExtra)
library(tidymodels)
```

```
-- Attaching packages --------------------------------------- tidymodels 0.1.1 --

v broom     0.7.1      v recipes   0.1.13
v dials     0.0.9      v rsample   0.0.8
v dplyr     1.0.2      v tibble    3.0.4
v ggplot2   3.3.2      v tidyr     1.1.2
v infer     0.5.3      v tune      0.1.1
v modeldata 0.0.2      v workflows 0.2.1
v parsnip   0.1.3      v yardstick 0.0.7
v purrr     0.3.4

-- Conflicts ------------------------------------------ tidymodels_conflicts() --
x dplyr::combine() masks gridExtra::combine()
x purrr::discard() masks scales::discard()
x dplyr::filter()  masks stats::filter()
x dplyr::lag()     masks stats::lag()
x recipes::step()  masks stats::step()
```

```
library(lubridate)
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
library(readxl)
library(ggcorrplot)
library(knitr)
library(rpart.plot)
```

```
Loading required package: rpart

Attaching package: 'rpart'

The following object is masked from 'package:dials':

    prune
```

```
library(randomForestExplainer)
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
library(doParallel)
```

```
Loading required package: foreach
```

```
Attaching package: 'foreach'
```

```
The following objects are masked from 'package:purrr':

    accumulate, when
```

```
Loading required package: iterators
```

```
Loading required package: parallel
```

```
library(vip)
```

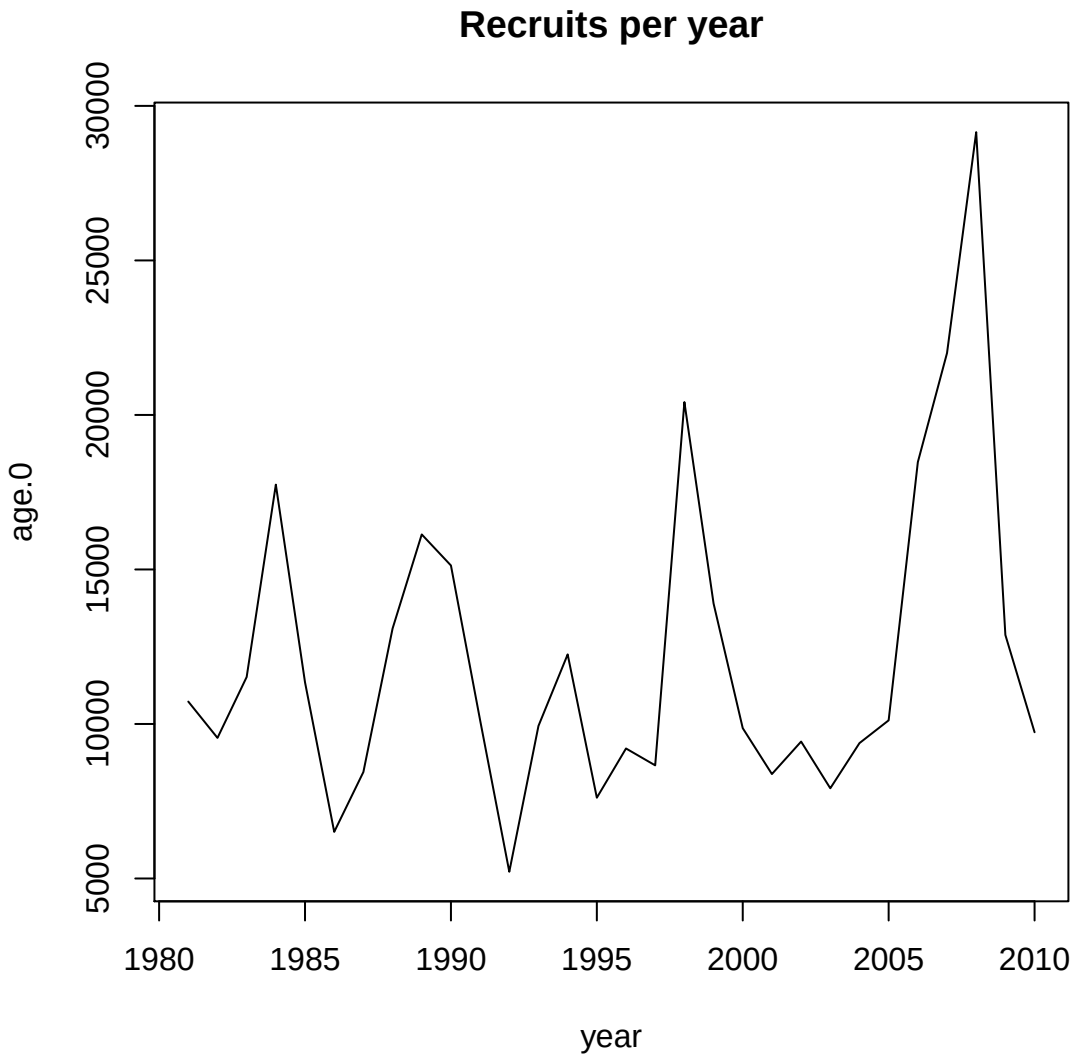
```
Attaching package: 'vip'
```

```
The following object is masked from 'package:utils':

    vi
```

```
require(ggplot2)
require(dplyr)
require(ggcorrplot)
require(broom)


data.dir <- "./data/"
petrale <- read.csv(file.path(data.dir,"Petrale_analyzed.data.csv"))
sable <- read.csv(file.path(data.dir,"Sablefish_Analyzed_Data_north.csv"))
sable_DFA <- read.csv(file.path(data.dir,"Sablefish_DFA_Data.csv"))

names(petrale)
```

```
 [1] "year"       "X"          "total.bio"  "sp.bio"     "depletion"
 [6] "age.0"      "spr"        "expl.rate"  "sp.bio.sd"  "age.0.sd"
[11] "yrminusone" "resids"     "log.resids" "SR_pred"    "Label"
[16] "dev"        "devsd"      "DDpre"      "Tpre.a"     "Tpre.b"
[21] "MLDegg"     "LSTegg"     "CSTegg1"    "DDegg1"     "CSTegg2"
[26] "LSTegg2"    "DDegg2"     "LSTlarv"    "CSTlarv"    "DDlarv"
[31] "LSTpjuv"    "CSTpjuv"    "DDpjuv"     "LSTbjuv.a"  "CSTbjuv.a"
[36] "LSTbjuv.b"  "CSTbjuv.b"
```

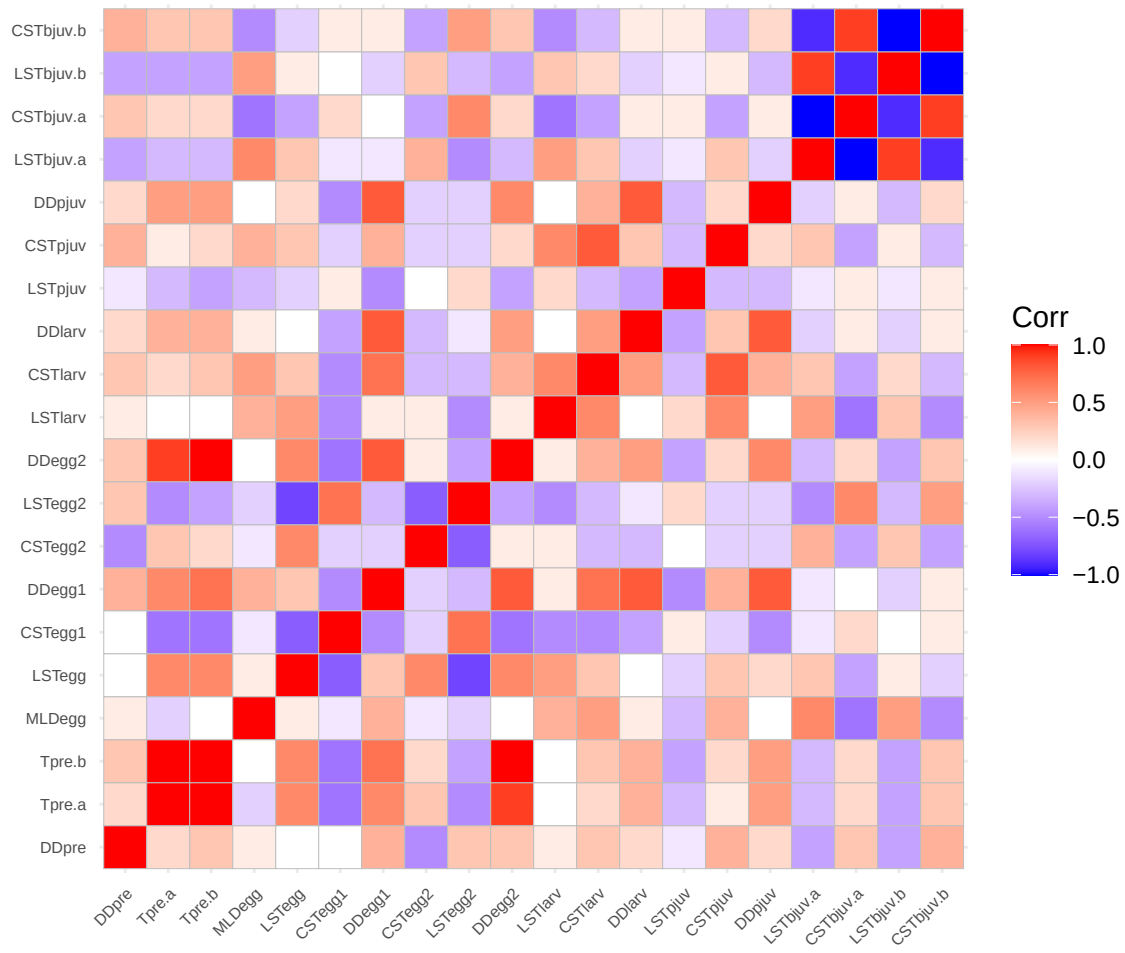
```r
#Plot recruits vs year and spawning stock biomass
plot(age.0~year,data=petrale, type="l", main="Recruits per year")
```




```r
plot(age.0~sp.bio,data=petrale, main="Recruits vs SSB")
```

Spawning biomass and recruitment are not very correlated..not surprising.

```
#Plot correlation
round(cor(petrale %>%
            dplyr::select(names(petrale)[18:37]) %>% data.frame), 1) %>%
ggcorrplot(tl.cex=6)
```




```
modpetrale <- petrale %>%
  select(-c(X,total.bio,depletion,spr,expl.rate,sp.bio.sd,age.0.sd,yrminusone,resids,log.resids,SR_pred
  pivot_longer(
    cols=-c(year,sp.bio,age.0),
    values_to="covs_val")
```

The time series for LST & CST are highly correlated, also Tpre.a and Tpre.b and DDegg and DDegg2. I remove correlated predictors below

```
#Do regression with different environmental time series
modpetrale %>%
  group_by(name) %>%
  do(glance(lm(age.0~sp.bio+covs_val,data=.))) %>%
  bind_rows(modpetrale %>%
              dplyr::select(-c(covs_val,name)) %>%
              distinct() %>%
```

```
            do(glance(lm(age.0~sp.bio,data=.))) %>%
            mutate(name="sp.bio only")) %>%
  arrange(-adj.r.squared) %>%
  mutate(across(where(is.double), round, 3)) %>%
  dplyr::select(-c(r.squared,logLik)) %>%
  data.frame %>%
  kable()
```

name	adj.r.squared	sigma	statistic	p.value	df	AIC	BIC	deviance	df.residual	nobs
MLDegg	0.220	4554.594	5.087	0.013	2	595.409	601.014	560096709	27	30
DDpre	0.191	4638.122	4.423	0.022	2	596.499	602.104	580828684	27	30
LSTbjuv.b	0.110	4865.588	2.787	0.079	2	599.372	604.977	639196568	27	30
CSTegg1	0.106	4875.859	2.718	0.084	2	599.499	605.103	641897960	27	30
DDpjuv	0.073	4964.367	2.145	0.137	2	600.578	606.183	665413350	27	30
DDlarv	0.063	4990.496	1.982	0.157	2	600.893	606.498	672436272	27	30
sp.bio only	0.047	5033.235	2.440	0.130	1	600.496	604.699	709336806	28	30
LSTegg2	0.038	5058.523	1.568	0.227	2	601.705	607.310	690893677	27	30
Tpre.a	0.024	5095.462	1.350	0.276	2	602.142	607.747	701020668	27	30
LSTegg	0.022	5099.084	1.329	0.281	2	602.184	607.789	702017715	27	30
CSTlarv	0.021	5101.732	1.314	0.285	2	602.216	607.820	702747084	27	30
CSTpjuv	0.014	5121.648	1.199	0.317	2	602.449	608.054	708244423	27	30
CSTegg2	0.013	5124.069	1.185	0.321	2	602.478	608.083	708914144	27	30
DDegg1	0.012	5124.367	1.183	0.322	2	602.481	608.086	708996791	27	30
LSTpjuv	0.012	5125.006	1.180	0.323	2	602.489	608.094	709173555	27	30
LSTlarv	0.012	5125.596	1.176	0.324	2	602.496	608.100	709336801	27	30

The best covariates are MLDegg (Mean mixed layer depth) and spawner preconditioning degree-days, none other <.05

```
wide_petrale <- modpetrale %>%
  pivot_wider(names_from = name, values_from=covs_val) %>%
  select(-c(year))

set.seed(100)
pet_split <- initial_split(wide_petrale) #By default, 3/4 data going into training; 1/4 test
pet_train <- training(pet_split)
pet_test <- testing(pet_split)

pet_split # View the pet_split object.
```

```
<Analysis/Assess/Total>
<23/7/30>
```

```
#Package not working for now, fix later
source("R/fit_split.R")

lm_spec <- linear_reg() %>%
  set_engine("lm")

#  Use the workflows and tidymodels function from above to fit a model to the training data
lm_fit <- fit_split(age.0~sp.bio+MLDegg, # specify formula
```

```
                    model=lm_spec, # specify model
                    split=pet_split) # specify data

lm_fit %>%
  collect_metrics()
```

```
# A tibble: 2 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard   4434.
2 rsq     standard      0.0534
```

```
lm_fit %>%
  collect_predictions()
```

```
# A tibble: 7 x 4
  id                .pred  .row age.0
  <chr>             <dbl> <int> <int>
1 train/test split 11066.     6  6509
2 train/test split  9283.    10 15129
3 train/test split  9801.    14 12252
4 train/test split 10428.    16  9207
5 train/test split 14158.    19 13899
6 train/test split 14484.    23  7919
7 train/test split 15778.    25 10118
```

```
lm_fit %>%
  collect_predictions %>% # from tune package, predicting using test data
  mutate(resid=(age.0-.pred)^2) %>% #calculate squared residuals
  summarise(rmse=sqrt(mean(resid)))
```

```
# A tibble: 1 x 1
   rmse
  <dbl>
1 4434.
```

```
fit_split(age.0~sp.bio+MLDegg, # specify formula
          model=lm_spec, # specify model
          metrics=metric_set(rmse,mae,mape), #specify model evaluation criteria.
          split=pet_split) %>%
  collect_metrics()
```

```
# A tibble: 3 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard      4434.
2 mae     standard      3794.
3 mape    standard        40.4
```

```r
#Create new split
set.seed(100)
pet_split <- initial_split(wide_petrale)
pet_train <- training(pet_split)
pet_test <- testing(pet_split)

#Look at range of data to see if we need to center/scale
apply(wide_petrale,2,range)
```

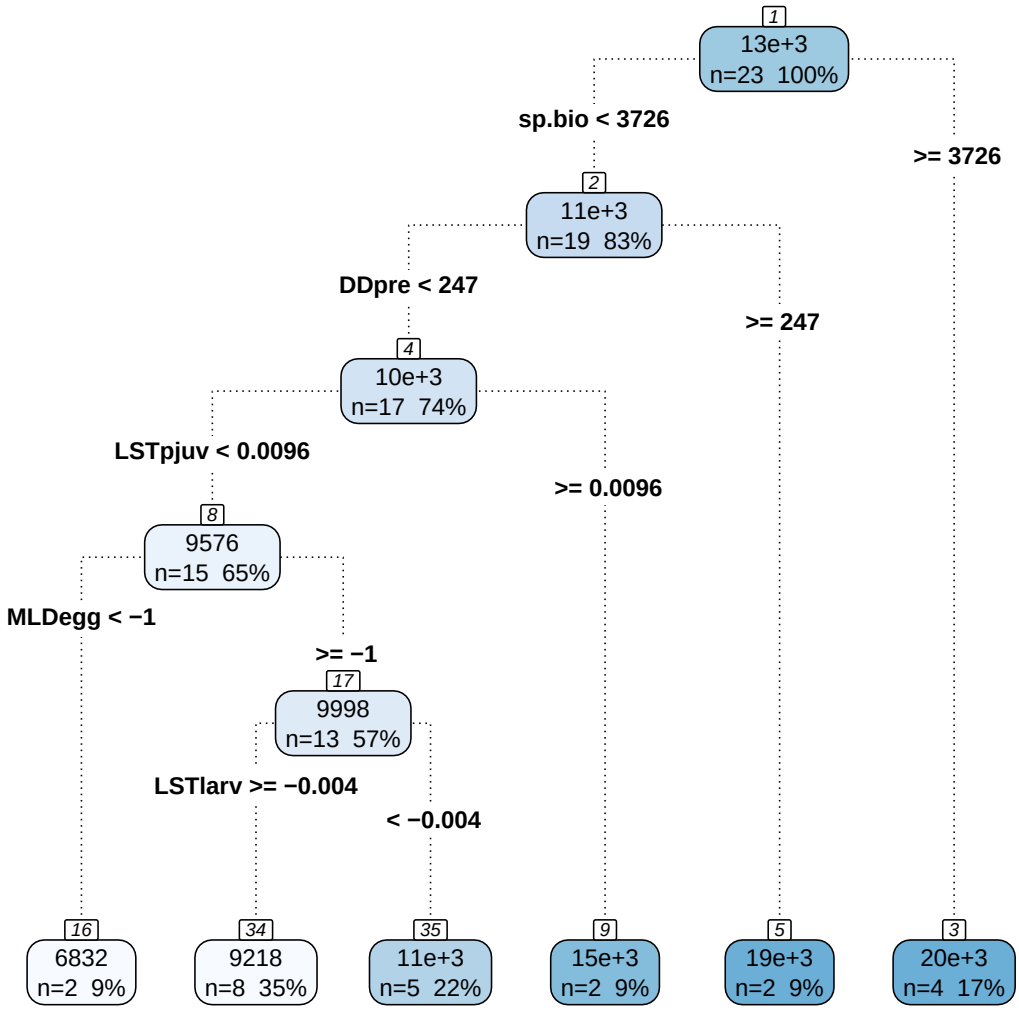
```
     sp.bio age.0   DDpre   Tpre.a   MLDegg      LSTegg      CSTegg1  DDegg1
[1,]   1667  5219 173.220 5.712132 31.04178 -0.04620180 -0.003487927 204.601
[2,]   4350 29151 255.921 6.723658 55.73978  0.07280556  0.002003658 303.633
         CSTegg2      LSTegg2     LSTlarv      CSTlarv  DDlarv     LSTpjuv
[1,] -0.01636910 -0.001686707 -0.04468344 -0.003200462 270.312 -0.06701870
[2,]  0.02833067  0.000878600  0.04510549  0.020146587 388.420  0.04350774
         CSTpjuv  DDpjuv   LSTbjuv.b
[1,] -0.00782200 157.945 0.001798998
[2,]  0.01081297 219.048 0.047165717
```

```r
#  Create a recipe.
pet_recipe <- recipe(age.0~.,data=pet_train) %>%
  step_center(MLDegg) %>%
  step_scale(MLDegg)

#  Prep the recipe.
pet_prep <- pet_recipe %>%
  prep(training=pet_train,retain=TRUE)

pet_prep
```

```
Data Recipe

Inputs:

      role #variables
   outcome          1
 predictor         16

Training data contained 23 data points and no missing data.

Operations:

Centering for MLDegg [trained]
Scaling for MLDegg [trained]
```

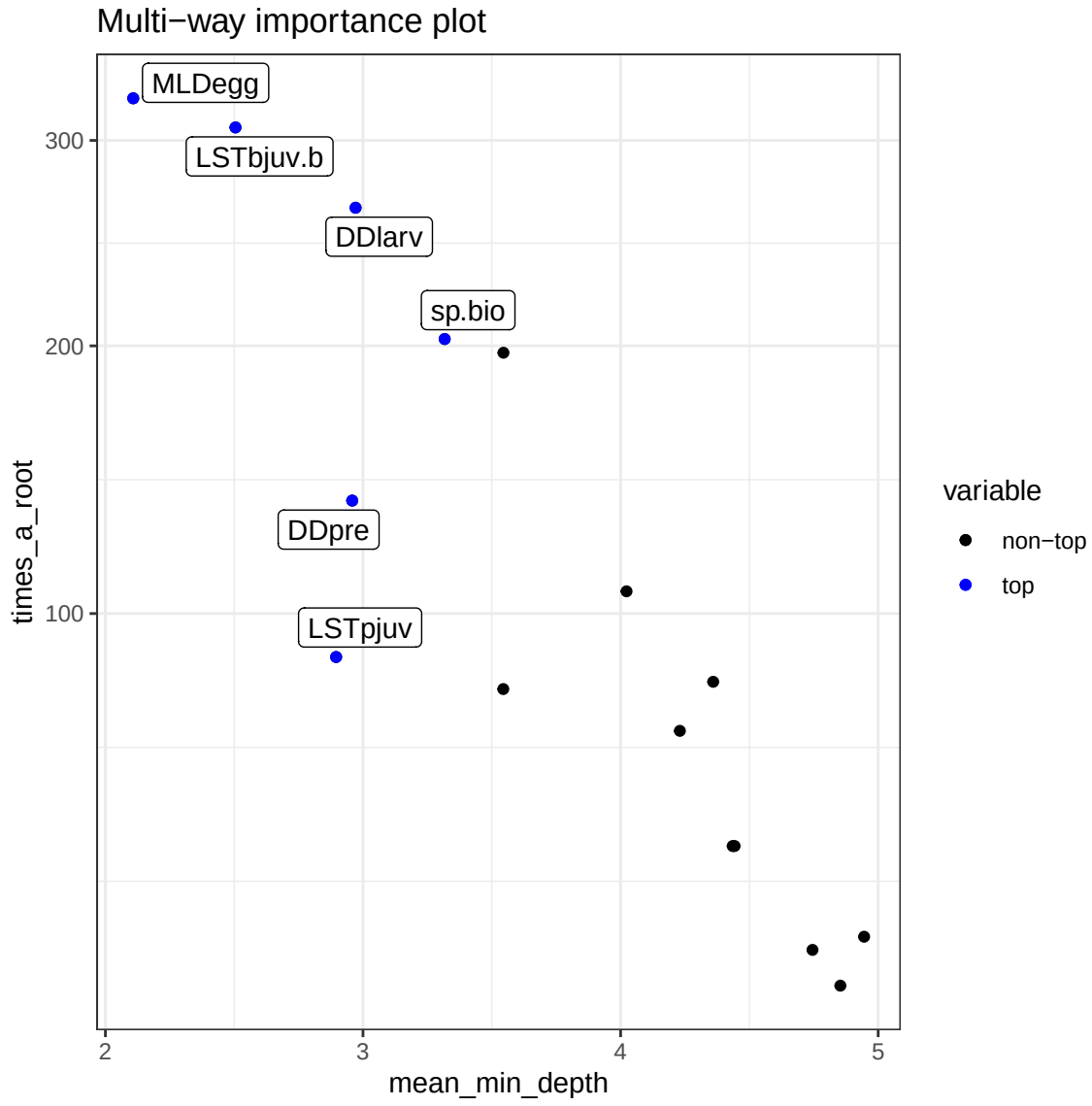
```r
dt_model <- decision_tree(min_n=5,tree_depth=10) %>%
  set_engine("rpart") %>%  # Specify the R library
  set_mode("regression") %>%  # Specify regression or classification
  fit(age.0~.,data=juice(pet_prep))

rpart.plot::rpart.plot(dt_model$fit,
                       type=4,
```

```
                    extra=101,
                    branch.lty=3,
                    nn=TRUE,
                    roundint=FALSE)
```




```
set.seed(100)
pet_split <- initial_split(wide_petrale,prop=0.90)
pet_train <- training(pet_split)

pet_recipe <- recipe(age.0~.,data=pet_train)

pet_prep <- pet_recipe %>%
  prep(training=pet_train,retain=TRUE)

rf_model <- rand_forest(trees=2000,mtry=4,mode="regression") %>% #rand_forest is a function in parsnip.
  set_engine("ranger",importance="permutation") %>% # rand_forest is part of the ranger package. We hav
```

```
  fit(age.0~.,data=juice(pet_prep))


#library(randomForestExplainer)
impt_frame<-measure_importance(rf_model$fit)

#impt_frame %>% head()
#  I like this plot as a way to illustate how several of the different RF hyperparameters fall out for
plot_multi_way_importance(impt_frame,no_of_labels = 6)
```



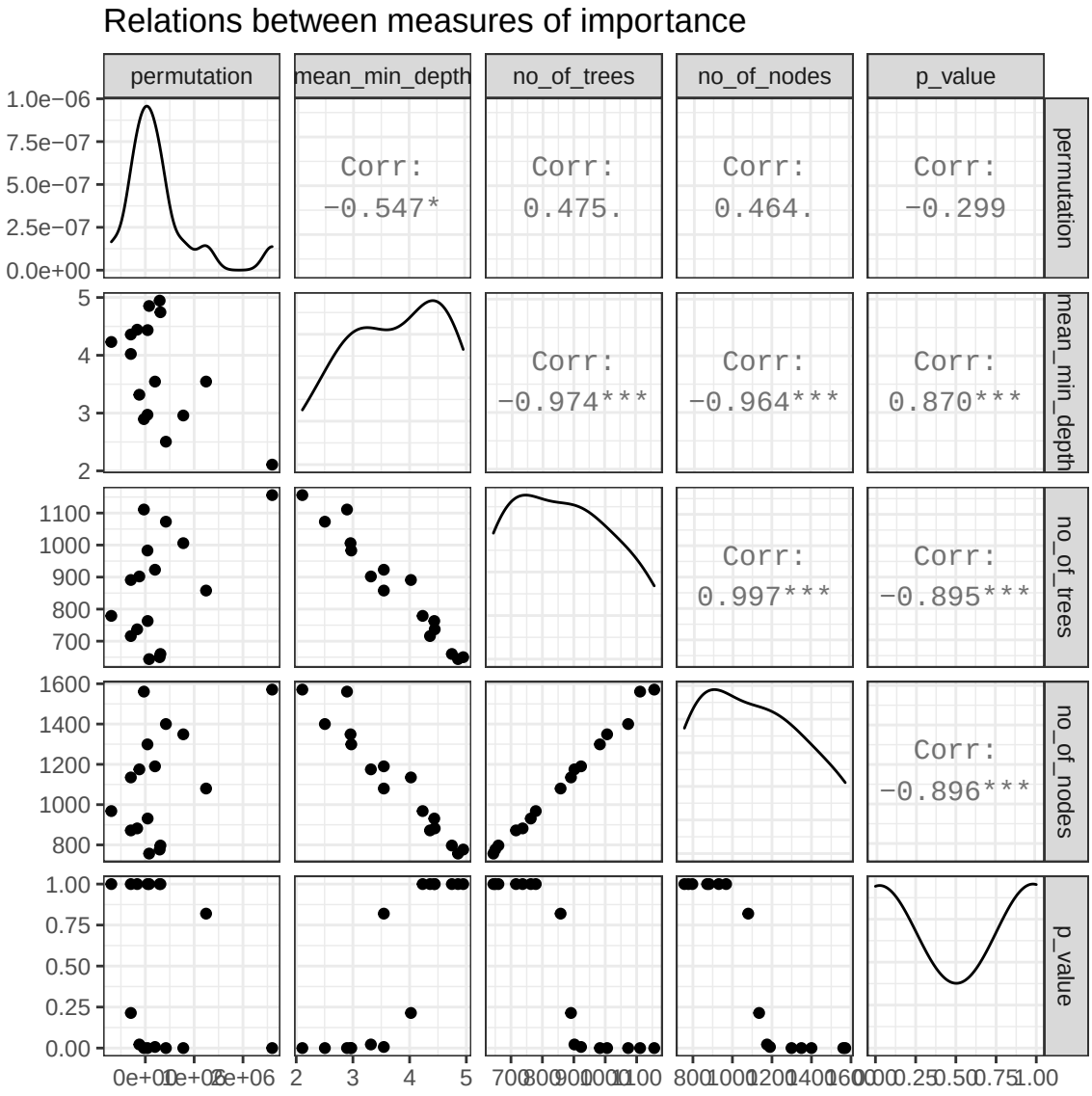
So lots of other variables seem to be more important than spawning biomass.

```
modpetrale %>%
  group_by(name) %>%
  do(glance(lm(age.0~covs_val,data=.))) %>%
  bind_rows(modpetrale %>%
```

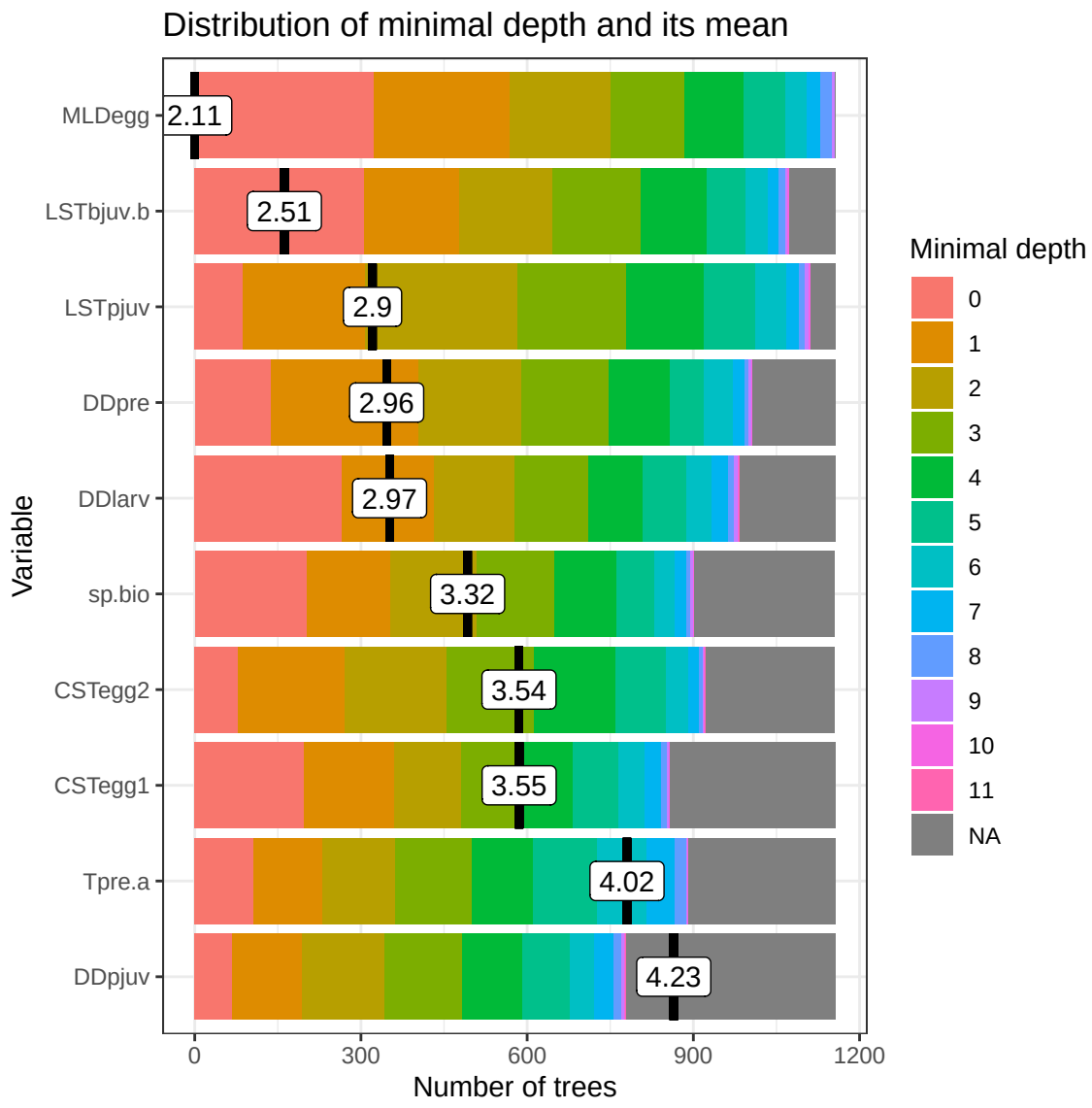
```
            dplyr::select(-c(covs_val,name)) %>%
            distinct() %>%
            do(glance(lm(age.0~sp.bio,data=.))) %>%
            mutate(name="sp.bio only")) %>%
  arrange(-adj.r.squared) %>%
  mutate(across(where(is.double), round, 3)) %>%
  dplyr::select(-c(r.squared,logLik)) %>%
  data.frame %>%
  kable()
```

name	adj.r.squared	sigma	statistic	p.value	df	AIC	BIC	deviance	df.residual	nobs
MLDegg	0.231	4521.897	9.713	0.004	1	594.068	598.271	572531575	28	30
LSTbjuv.b	0.122	4832.227	5.025	0.033	1	598.050	602.254	653811791	28	30
CSTegg1	0.089	4922.288	3.827	0.060	1	599.158	603.362	678409764	28	30
sp.bio only	0.047	5033.235	2.440	0.130	1	600.496	604.699	709336806	28	30
DDpre	0.042	5047.130	2.272	0.143	1	600.661	604.865	713258614	28	30
DDpjuv	0.013	5122.238	1.391	0.248	1	601.547	605.751	734645045	28	30
Tpre.a	0.009	5134.383	1.252	0.273	1	601.689	605.893	738132917	28	30
DDlarv	0.004	5146.356	1.116	0.300	1	601.829	606.033	741579492	28	30
LSTegg	-0.004	5166.082	0.894	0.352	1	602.059	606.262	747275404	28	30
LSTegg2	-0.009	5179.257	0.747	0.395	1	602.212	606.415	751091554	28	30
CSTlarv	-0.022	5213.963	0.366	0.550	1	602.612	606.816	761191376	28	30
CSTpjuv	-0.031	5236.641	0.121	0.731	1	602.873	607.076	767827595	28	30
LSTlarv	-0.034	5242.367	0.060	0.809	1	602.938	607.142	769507417	28	30
CSTegg2	-0.036	5247.620	0.003	0.954	1	602.998	607.202	771050514	28	30
DDegg1	-0.036	5247.644	0.003	0.956	1	602.999	607.202	771057579	28	30
LSTpjuv	-0.036	5247.934	0.000	0.999	1	603.002	607.206	771142643	28	30

```
plot_importance_ggpairs(impt_frame)
```




```
md_frame <- min_depth_distribution(rf_model$fit)

plot_min_depth_distribution(md_frame, mean_sample = "top_trees")
```




```
fit_split(age.0 ~ .,
          model = rand_forest() %>%
            set_engine("ranger") %>%
            set_mode("regression"),
          split = pet_split) %>%
  collect_metrics()
```

```
# A tibble: 2 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       3793.
2 rsq     standard          1
```

Hack the leave one out validation

```
initial_split(wide_petrale, prop=0.96)
```

```
<Analysis/Assess/Total>
<29/1/30>
```

```
set.seed(100)
pet_split <- initial_split(wide_petrale,prop=0.95)
pet_train <- training(pet_split)
#  First specify the model framework
tune_spec <- rand_forest(mtry = tune(),
              trees = tune(),
              min_n = tune()) %>%
  set_engine("ranger") %>%
  set_mode("regression")

# Now specify the model
tune_wf <- workflow() %>%
  add_recipe(pet_recipe) %>%
  add_model(tune_spec)
```

```
dim(vfold_cv(wide_petrale,100))
```

```
[1] 30  2
```

```
#set.seed(234) If you were actually doing random subsets, you'd want to set the seed.
pet_folds <- vfold_cv(pet_train,100)

#  Setup parallel processing.
cl <- makeCluster(3)
doParallel::registerDoParallel(cl)
clusterEvalQ(cl, .libPaths("C:/~/R/win-library/4.0/"))
```

```
[[1]]
[1] "C:/~/R/win-library/4.0"            "C:/Program Files/R/R-4.0.1/library"

[[2]]
[1] "C:/~/R/win-library/4.0"            "C:/Program Files/R/R-4.0.1/library"

[[3]]
[1] "C:/~/R/win-library/4.0"            "C:/Program Files/R/R-4.0.1/library"
```

```
#  Now run the model, which is going to try 20 different values for each of the tuning hyperparameters
set.seed(345)
tune_res <- tune_grid(
  tune_wf,
  resamples = pet_folds,
  grid = 20
)
```

```
i Creating pre-processing data to finalize unknown parameter: mtry
```

```
tune_res
```

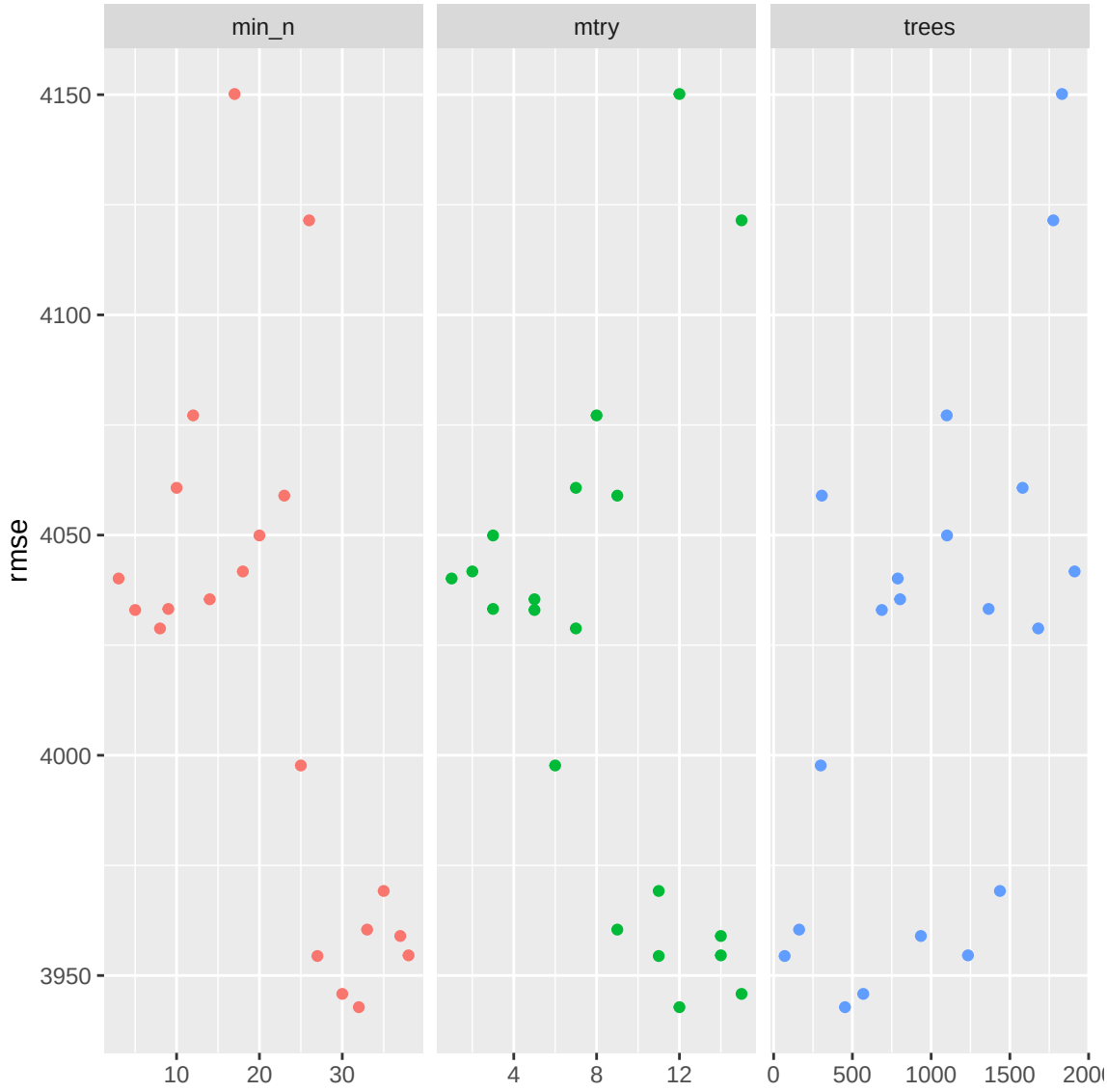
```
# Tuning results
# 100-fold cross-validation
# A tibble: 29 x 4
   splits         id     .metrics          .notes
   <list>         <chr>  <list>            <list>
 1 <split [28/1]> Fold01 <tibble [40 x 7]> <tibble [0 x 1]>
 2 <split [28/1]> Fold02 <tibble [40 x 7]> <tibble [0 x 1]>
 3 <split [28/1]> Fold03 <tibble [40 x 7]> <tibble [0 x 1]>
 4 <split [28/1]> Fold04 <tibble [40 x 7]> <tibble [0 x 1]>
 5 <split [28/1]> Fold05 <tibble [40 x 7]> <tibble [0 x 1]>
 6 <split [28/1]> Fold06 <tibble [40 x 7]> <tibble [0 x 1]>
 7 <split [28/1]> Fold07 <tibble [40 x 7]> <tibble [0 x 1]>
 8 <split [28/1]> Fold08 <tibble [40 x 7]> <tibble [0 x 1]>
 9 <split [28/1]> Fold09 <tibble [40 x 7]> <tibble [0 x 1]>
10 <split [28/1]> Fold10 <tibble [40 x 7]> <tibble [0 x 1]>
# ... with 19 more rows
```

```
tune_res %>%
  collect_metrics() %>%
  filter(.metric=="rmse") %>%
  arrange(mean)
```

```
# A tibble: 20 x 9
    mtry trees min_n .metric .estimator  mean     n std_err .config
   <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
 1    12   453    32 rmse    standard   3943.    29    669. Model04
 2    15   569    30 rmse    standard   3946.    29    671. Model07
 3    11    70    27 rmse    standard   3954.    29    653. Model16
 4    14  1234    38 rmse    standard   3955.    29    668. Model20
 5    14   935    37 rmse    standard   3959.    29    669. Model15
 6     9   162    33 rmse    standard   3960.    29    665. Model03
 7    11  1437    35 rmse    standard   3969.    29    668. Model18
 8     6   299    25 rmse    standard   3998.    29    640. Model06
 9     7  1680     8 rmse    standard   4029.    29    625. Model12
10     5   687     5 rmse    standard   4033.    29    618. Model17
11     3  1365     9 rmse    standard   4033.    29    608. Model14
12     5   803    14 rmse    standard   4035.    29    625. Model19
13     1   788     3 rmse    standard   4040.    29    618. Model13
14     2  1911    18 rmse    standard   4042.    29    621. Model08
15     3  1101    20 rmse    standard   4050.    29    613. Model05
16     9   306    23 rmse    standard   4059.    29    657. Model10
17     7  1581    10 rmse    standard   4061.    29    628. Model02
18     8  1099    12 rmse    standard   4077.    29    629. Model11
19    15  1776    26 rmse    standard   4121.    29    658. Model09
20    12  1831    17 rmse    standard   4150.    29    643. Model01
```

```
tune_res %>%
  collect_metrics() %>%
  filter(.metric == "rmse") %>%
  select(mean, min_n, mtry, trees) %>%
```

```
  pivot_longer(c(min_n,mtry,trees),
    values_to = "value",
    names_to = "parameter"
  ) %>%
  ggplot(aes(value, mean, color = parameter)) +
  geom_point(show.legend = FALSE) +
  facet_wrap(~parameter, scales = "free_x") +
  labs(x = NULL, y = "rmse")
```

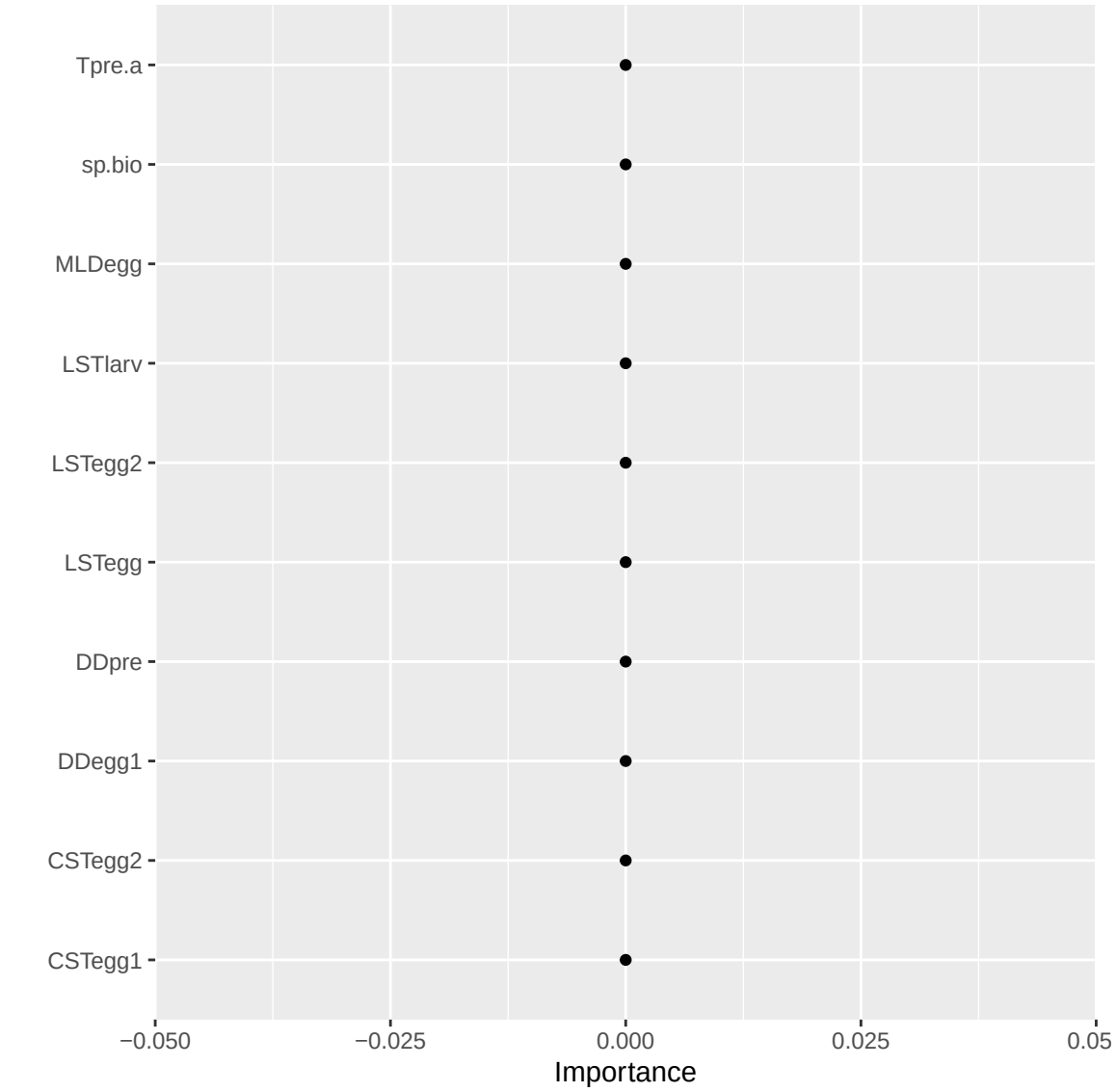



```
best_rmse <- select_best(tune_res,"rmse")

best_rmse
```

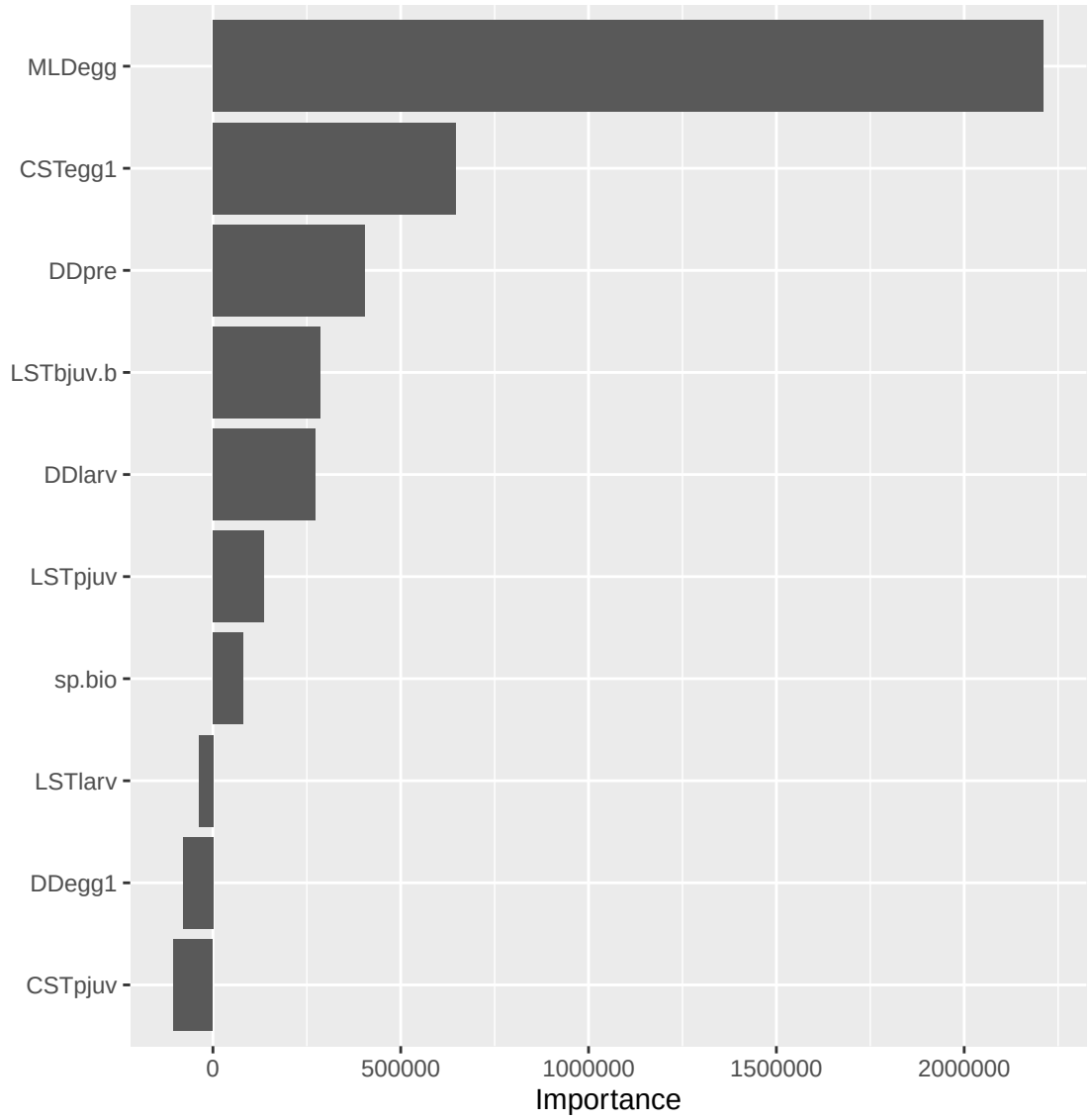
```
# A tibble: 1 x 4
   mtry trees min_n .config
  <int> <int> <int> <chr>
```

```
1    12   453    32 Model04
```

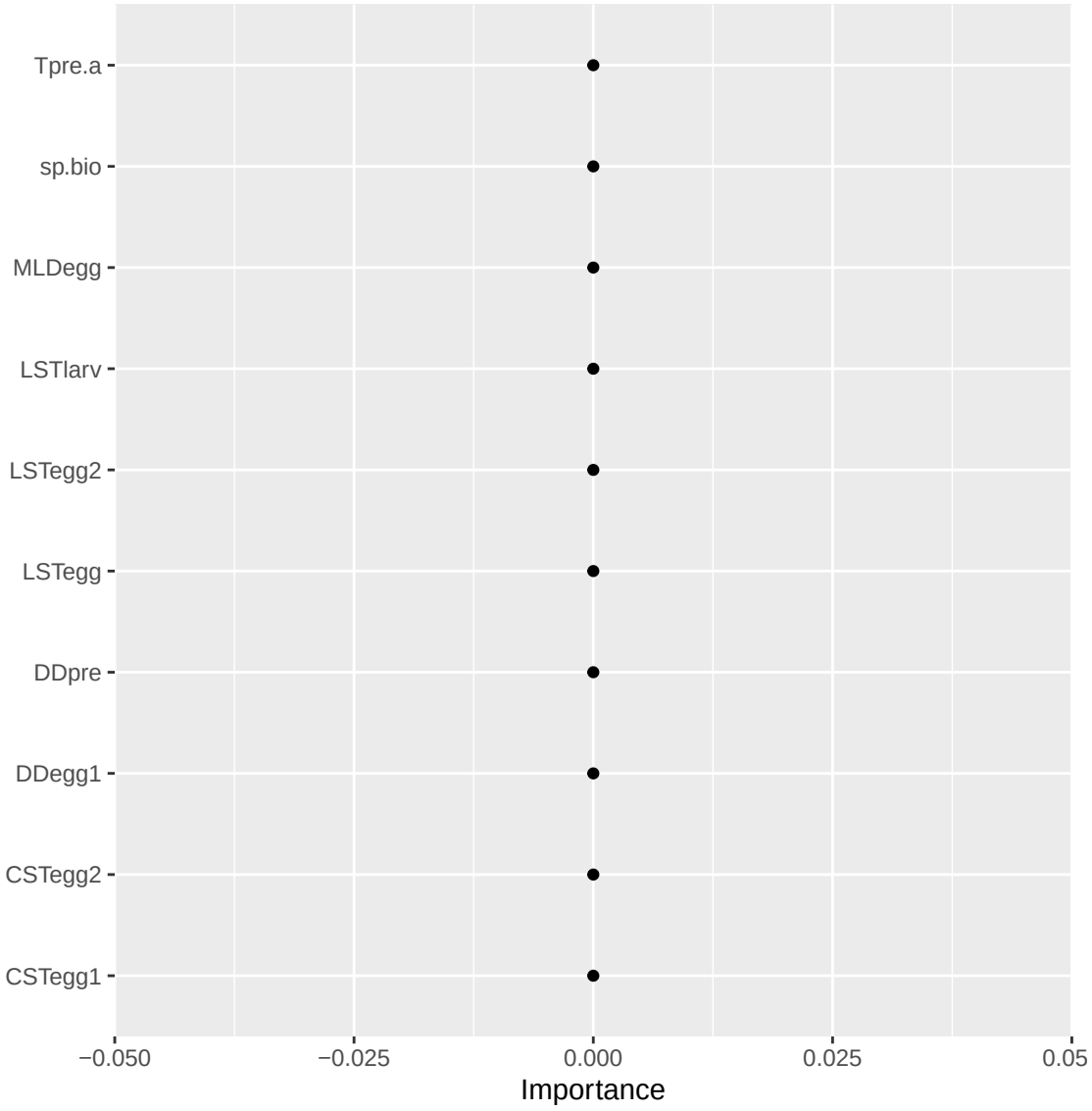
```
final_rf <- finalize_model(
  tune_spec,
  best_rmse
)

final_rf %>%
  set_engine("ranger", importance = "permutation") %>%
  fit(age.0 ~ .,
    data = juice(pet_prep)) %>%
  vip::vip(geom = "point")
```




```
rand_forest(trees=1911,mtry=7,min_n=18,mode="regression") %>% #rand_forest is a function in parsnip.
  set_engine("ranger",importance="permutation") %>% # rand_forest is part of the ranger package. We hav
  fit(age.0~.,data=juice(pet_prep)) %>%
  vip()
```




```
final_rf %>%
  set_engine("ranger", importance = "impurity") %>%
  fit(age.0 ~ .,
    data = juice(pet_prep)) %>%
  vip::vip(geom = "point")
```


Tpre.a
sp.bio
MLDegg
LSTlarv
LSTegg2
LSTegg
DDpre
DDegg1
CSTegg2
CSTegg1
−0.050
−0.025
0.000
0.025
0.05
Importance