Random Forests

Amber Lee

7/6/2021

Set up

Load libraries

```
library(tidyverse)
library(stringr)
library(lubridate)
library(rsample)      # data splitting
library(randomForest) # basic implementation
library(ranger)       # a faster implementation of randomForest
```

Read data

```
water20 <- read.csv("../../LTRM data/water_data_qfneg.csv", header = TRUE)
```

Data cleaning

- Add year and season variable
- Change `FLDNUM` to be categorical (character).

```
water20 <- water20 %>%
  mutate(nice_date = mdy(DATE),
         year = year(nice_date),
         quarter = quarter(nice_date, fiscal_start = 3),
         quarter = as.factor(quarter),
         FLDNUM = case_when(FLDNUM == 1 ~ "Lake City, MN",
                            FLDNUM == 2 ~ "Onalaska, WI",
                            FLDNUM == 3 ~ "Bellevue, IA",
                            FLDNUM == 4 ~ "Brighton, IL",
                            FLDNUM == 5 ~ "Jackson, MO",
                            FLDNUM == 6 ~ "Havana, IL"),
         FLDNUM = as.factor(FLDNUM),
         STRATUM = as.factor(STRATUM)) %>% # CART can split on categorical variable if
                                          # encoded as a factor variable
  select(-SHEETBAR, -nice_date, -DATE, -LOCATCD)
```

Random forests notes

UC Riverside Programming Guide

- Trees can have high variance and poor predictive performance
- Bagging trees (growing trees from a bootstrap resample of the training data) introduces randomness. However, bagged trees are still correlated because of similar structure of data. (The first few splits tend to be the same.)
- Random forests extend on bagged trees by limiting each split to a random subset of all the variables. Let p be the number of variables and m be the size of this random subset. Usually $m = p/3$. Random forests have the least correlated trees.
- Out of bag (OOB) error: as a result of the bootstrap resampling, the data that *aren't* sample provide a natural validation set. This helps to decide on the number of trees to stabilize the error rate.
- One disadvantage is computational time.
- Random forests are not able to hand missing predictor values. https://stats.stackexchange.com/questions/98953/why-doesnt-random-forest-handle-missing-values-in-predictors

Implement RF with randomForest

Unable to predict when another variable is missing

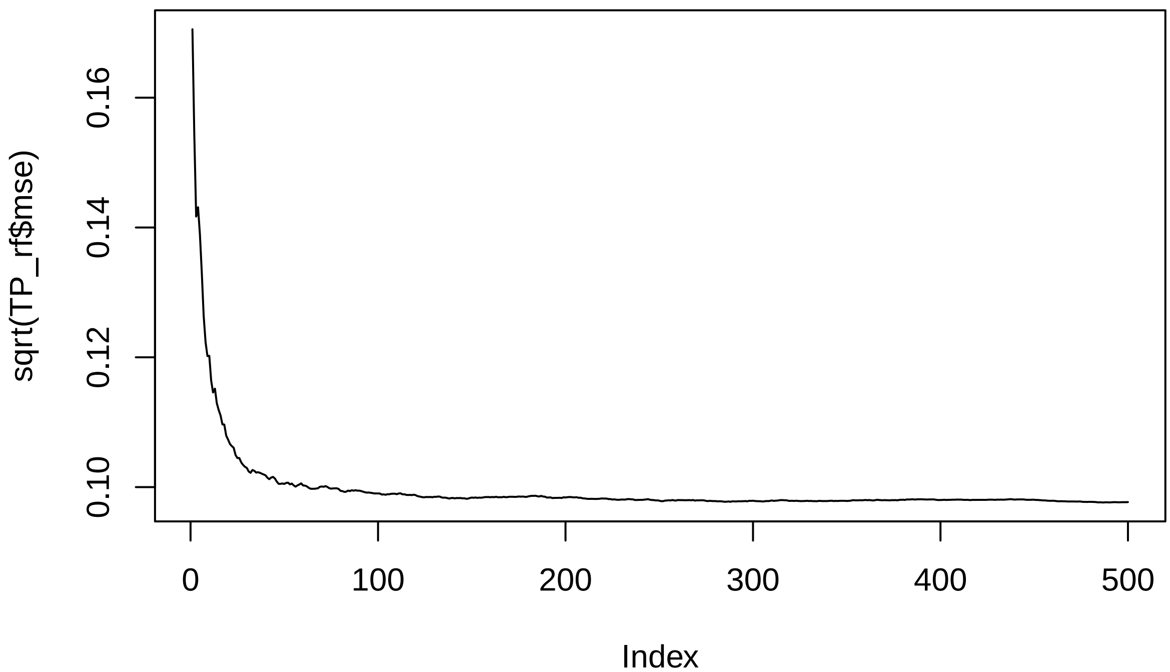
Total Phosphorous

```
set.seed(4774)

fullTP <- water20 %>% filter(!is.na(TP))

TP_split <- initial_split(fullTP, prop = 0.8)
TP_train <- training(TP_split)
TP_test <- testing(TP_split)
```

```
TP_rf <- randomForest(
  formula = TP ~ .,
  data    = TP_train,
  na.action = na.roughfix
)

TP_rf
```

```
##
## Call:
##  randomForest(formula = TP ~ ., data = TP_train, na.action = na.roughfix)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.009543517
##                     % Var explained: 75.66
```

```
plot(sqrt(TP_rf$mse), type = "l")
```




```
which.min(TP_rf$mse)
```

```
## [1] 487
```

```
TP_pred <- predict(TP_rf, newdata = TP_test)

TP_test$TP_pred <- TP_pred

TP_eval <- TP_test %>%
  filter(!is.na(TP_pred)) %>%
  mutate(residual_sq = (TP - TP_pred)^2,
         abs_residual = abs(TP - TP_pred)) %>%
  select(TP, TP_pred, residual_sq, abs_residual)

print("RSME")
```

```
## [1] "RSME"
```

```
sqrt(sum(TP_eval$residual_sq)/dim(TP_eval)[1])
```

```
## [1] 0.07242187
```

```
print("MAE")
```

```
## [1] "MAE"
```

```
sum(TP_eval$abs_residual)/dim(TP_eval)[1]
```

```
## [1] 0.03142215
```

```
dim(TP_eval)
```

```
## [1] 4023    4
```

```r
dim(TP_test)
```

```
## [1] 6290   18
```

```r
dim(TP_train)
```

```
## [1] 25160    17
```

```r
dim(fullTP)
```

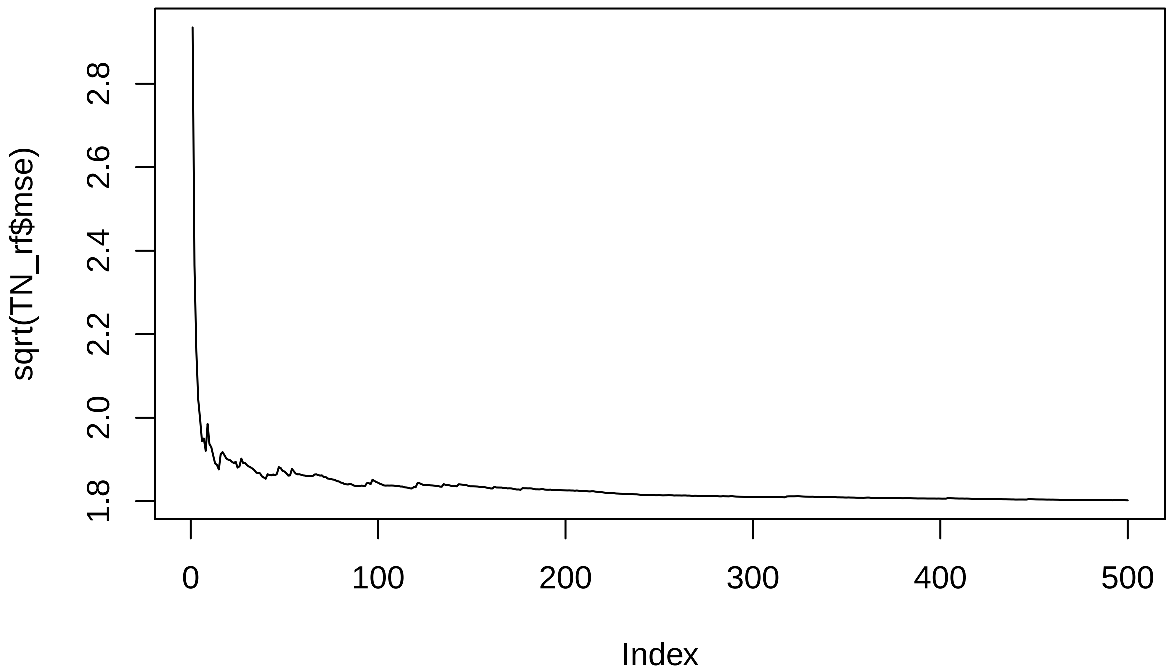
```
## [1] 31450    17
```

Total Nitrogen

```r
fullTN <- water20 %>% filter(!is.na(TN)) %>% sample_frac(0.5)
TN_split <- initial_split(fullTN, prop = 0.8)
TN_train <- training(TN_split)
TN_test <- testing(TN_split)

# default RF model
TN_rf <- randomForest(
  formula = TN ~ .,
  data    = TN_train,
  na.action = na.roughfix
)

TN_rf
```

```
##
## Call:
##  randomForest(formula = TN ~ ., data = TN_train, na.action = na.roughfix)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 3.247657
##                     % Var explained: 32.28
```

```r
plot(sqrt(TN_rf$mse), type = "l")
```




```
which.min(TN_rf$mse)
```

```
## [1] 500
```

```
TN_test$TN_pred <- predict(TN_rf, newdata = TN_test)

TN_eval <- TN_test %>%
  filter(!is.na(TN_pred)) %>%
  mutate(residual_sq = (TN - TN_pred)^2,
         abs_residual = abs(TN - TN_pred)) %>%
  select(TN, TN_pred, residual_sq, abs_residual)

print("RSME")
```

```
## [1] "RSME"
```

```
sqrt(sum(TN_eval$residual_sq)/dim(TN_eval)[1])
```

```
## [1] 0.7342809
```

```
print("MAE")
```

```
## [1] "MAE"
```

```
sum(TN_eval$abs_residual)/dim(TN_eval)[1]
```

```
## [1] 0.4483802
```

```
dim(TN_eval)
```

```
## [1] 1992    4
```

```
dim(TN_test)
```

```
## [1] 3219   18
```

```
dim(TN_train)
```

```
## [1] 12875    17
```

```r
dim(fullTN)
```

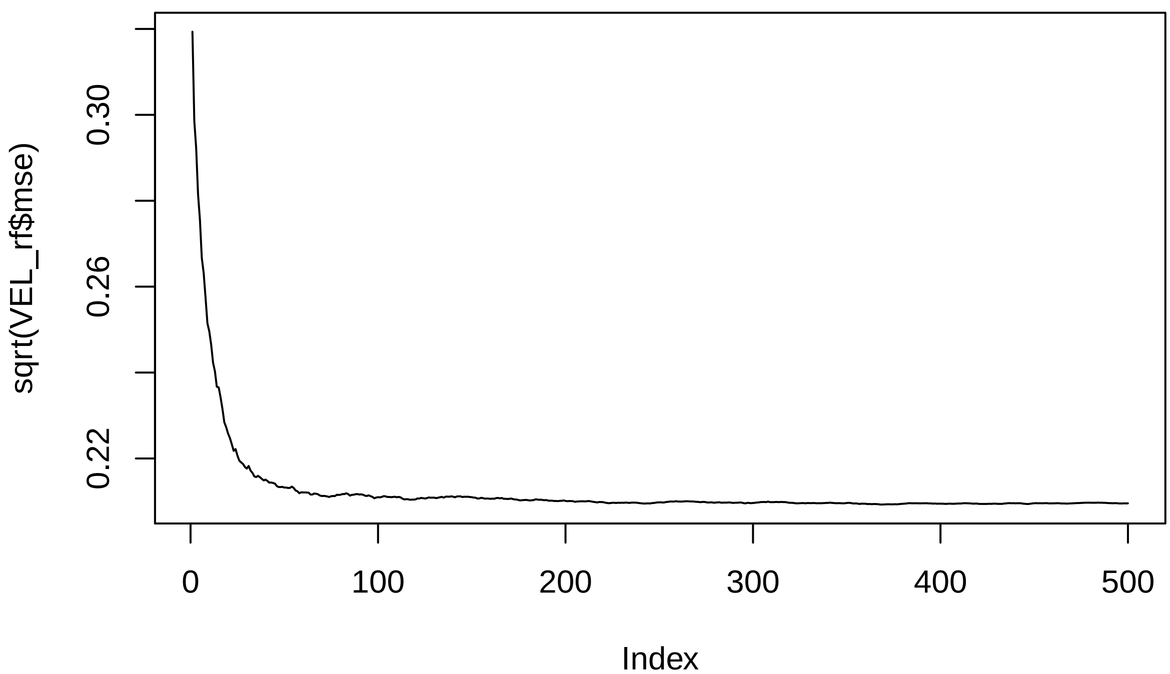
```
## [1] 16094    17
```

Velocity

```r
fullVEL <- water20 %>% filter(!is.na(VEL)) %>% sample_frac(0.5)
VEL_split <- initial_split(fullVEL, prop = 0.8)
VEL_train <- training(VEL_split)
VEL_test <- testing(VEL_split)

VEL_rf <- randomForest(
  formula = VEL ~ .,
  data    = VEL_train,
  na.action = na.roughfix
)

VEL_rf
```

```
##
## Call:
##  randomForest(formula = VEL ~ ., data = VEL_train, na.action = na.roughfix)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 0.04391329
##                     % Var explained: 72.28
```

```r
plot(sqrt(VEL_rf$mse), type = "l")
```



```r
which.min(VEL_rf$mse)
```

```
## [1] 368
```

```r
VEL_test$VEL_pred <- predict(VEL_rf, newdata = VEL_test)

VEL_eval <- VEL_test %>%
  filter(!is.na(VEL_pred)) %>%
  mutate(residual_sq = (VEL - VEL_pred)^2,
         abs_residual = abs(VEL - VEL_pred)) %>%
  select(VEL, VEL_pred, residual_sq, abs_residual)

print("RSME")
```

```
## [1] "RSME"
```

```r
sqrt(sum(VEL_eval$residual_sq)/dim(VEL_eval)[1])
```

```
## [1] 0.1982307
```

```r
print("MAE")
```

```
## [1] "MAE"
```

```r
sum(VEL_eval$abs_residual)/dim(VEL_eval)[1]
```

```
## [1] 0.1035669
```

```r
dim(VEL_eval)
```

```
## [1] 2027    4
```

```r
dim(VEL_test)
```

```
## [1] 5618   18
```

```r
dim(VEL_train)
```

```
## [1] 22472    17
```

```r
dim(fullVEL)
```

```
## [1] 28090    17
```