final RF interpolation

Amber Lee

7/15/2021

Set up

Load libraries

```r
library(tidyverse)
library(lubridate)    # date types
library(kableExtra)   # presenting tables
library(rsample)      # data splitting
# library(randomForest) # basic implementation
library(ranger)       # a faster implementation of randomForest
```

Read data

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

Data cleaning

- Add year and season variable, where year and season are both factor variables
- Change `FLDNUM` and `STRATUM` to be factor variables
- Remove sheetbar, date, location code, latitude, and longitude, from building RF

```r
cleaning_for_forests <- function(df){
  # this function will be called after imputing

  df <- df %>%
    mutate(SEASON = as.factor(SEASON),
           YEAR = as.factor(YEAR),
           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 = case_when(STRATUM == 1 ~ "Main channel",
                               STRATUM == 2 ~ "Side channel",
                               STRATUM == 3 ~ "Backwater area contiguous to the main channel",
                               STRATUM == 4 ~ "Lake Pepin or Swan Lake",
                               STRATUM == 5 ~ "Impounded",
                               STRATUM == 6 ~ "Isolated",
                               STRATUM == 9 ~ "Unexploded Ordinance Area - Pool 13",
                               TRUE ~ as.character(STRATUM)),
           STRATUM = as.factor(STRATUM),
           TN = as.numeric(TN),
           TP = as.numeric(TP),
           TEMP = as.numeric(TEMP),
           DO = as.numeric(DO),
           TURB = as.numeric(TURB),
           COND = as.numeric(COND),
           VEL = as.numeric(VEL),
           SS = as.numeric(SS),
           WDP = as.numeric(WDP),
           CHLcal = as.numeric(CHLcal),
           SECCHI = as.numeric(SECCHI))

  return(df)

}

water20 <- water20 %>%
  mutate(nice_date = mdy(DATE), # to extract year and quarter
         YEAR = year(nice_date),
         SEASON = quarter(nice_date, fiscal_start = 3)) %>%
  select(-nice_date) %>%
  # year and quarter become factors later
  cleaning_for_forests()

water_interpolation <- water20 %>%
  select(-DATE, -LOCATCD, -LATITUDE, -LONGITUDE)
```

Median/mode imputation

```r
replace_median_mode <- function(var_str, dataset, response_var){
  # for imputing test data with median/mode

  # iterate through each variable name var_str with sapply to
  # replace every variable except for response_var (no need to impute)

  # column of the dataset
  data_col <- eval(parse(text = paste("dataset$", var_str, sep = "")))

  if (var_str %in% c(response_var, "FLDNUM",  "STRATUM", "YEAR",  "SEASON", "SHEETBAR")){
    # these other variables have no missingness
    return(data_col) # don't impute the response variable
  }

  if (class(data_col) == "factor"){
    # categorical imputation
    # take the mode
    counts_table <- table(data_col)

    # retrieve mode
    imputation <- names(counts_table)[counts_table == max(counts_table)]

  } else if (class(data_col) %in% c("numeric", "integer")){
    # continuous imputation
    # take the median

    imputation <- median(data_col, na.rm = TRUE)

  } else (return("error in class of variable"))

  data_col[is.na(data_col)] <- imputation

  #   print("running replace_median_mode")
  print(paste(var_str, "imputation is", as.character(imputation)))

  if (sum(!is.na(data_col)) == length(data_col)){

    return(data_col)

  } else return("error in imputing NAs")

}

impute_data <- function(df, response_var, df_train){

  df <- data.frame(sapply(names(df), replace_median_mode, df, response_var)) %>%
    cleaning_for_forests()

  # trick to fix incompatible types for random forest
  # source: https://stackoverflow.com/questions/24829674/r-random-forest-error-type-of-predictors-in-ne

  df <- rbind(df_train[1, ] , df)
  df <- df[-1,]

  return(df)

}
```

TP

```
TP_train <- water_interpolation %>%
  filter(!is.na(TP))

TP_train <- impute_data(TP_train, "TP", TP_train)
```

```
## [1] "TN imputation is 2.541"
## [1] "TEMP imputation is 14.5"
## [1] "DO imputation is 9.7"
## [1] "TURB imputation is 21"
## [1] "COND imputation is 461"
## [1] "VEL imputation is 0.1"
## [1] "SS imputation is 26.1"
## [1] "WDP imputation is 2.22"
## [1] "CHLcal imputation is 16.37744"
## [1] "SECCHI imputation is 42"
```

```
TP_missing <- water_interpolation %>%
  filter(is.na(TP))

TP_missing <- impute_data(TP_missing, "TP", TP_train)
```

```
## [1] "TN imputation is 1.914"
## [1] "TEMP imputation is 14.7"
## [1] "DO imputation is 9.6"
## [1] "TURB imputation is 20"
## [1] "COND imputation is 463"
## [1] "VEL imputation is 0.1"
## [1] "SS imputation is 25.44"
## [1] "WDP imputation is 2.25"
## [1] "CHLcal imputation is 16.8006"
## [1] "SECCHI imputation is 41"
```

Train and interpolate

```
TP_rf <- ranger(
  formula = TP ~ .,
  data    = TP_train %>% select(-SHEETBAR)
)

TP_missing$predicted_TP <- round(predict(TP_rf,
                                         data = TP_missing %>%
                                           select(-SHEETBAR))$predictions, 3)

TP_missing <- TP_missing %>% select(SHEETBAR, predicted_TP)
```

TN

```
TN_train <- water_interpolation %>%
  filter(!is.na(TN))

TN_train <- impute_data(TN_train, "TN", TN_train)
```

```
## [1] "TP imputation is 0.163"
## [1] "TEMP imputation is 14.7"
## [1] "DO imputation is 9.7"
## [1] "TURB imputation is 21"
## [1] "COND imputation is 461"
## [1] "VEL imputation is 0.1"
## [1] "SS imputation is 26.4"
## [1] "WDP imputation is 2.22"
## [1] "CHLcal imputation is 16.48571"
## [1] "SECCHI imputation is 41"
```

```
TN_missing <- water_interpolation %>%
  filter(is.na(TN))

TN_missing <- impute_data(TN_missing, "TN", TN_train)
```

```
## [1] "TP imputation is 0.17"
## [1] "TEMP imputation is 14.7"
## [1] "DO imputation is 9.7"
## [1] "TURB imputation is 20"
## [1] "COND imputation is 463"
## [1] "VEL imputation is 0.1"
## [1] "SS imputation is 25.2"
## [1] "WDP imputation is 2.26"
## [1] "CHLcal imputation is 16.72338"
## [1] "SECCHI imputation is 41"
```

Train and interpolate

```
TN_rf <- ranger(
  formula = TN ~ .,
  data    = TN_train %>% select(-SHEETBAR)
)

TN_missing$predicted_TN <- round(predict(TN_rf,
                                     data = TN_missing %>%
                                       select(-SHEETBAR))$predictions, 3)

TN_missing <- TN_missing %>% select(SHEETBAR, predicted_TN)
```

VEL

```
VEL_train <- water_interpolation %>%
  filter(!is.na(VEL))

VEL_train <- impute_data(VEL_train, "VEL", VEL_train)
```

```
## [1] "TN imputation is 2.386"
## [1] "TP imputation is 0.15"
## [1] "TEMP imputation is 14.4"
## [1] "DO imputation is 9.9"
## [1] "TURB imputation is 16"
## [1] "COND imputation is 443"
## [1] "SS imputation is 20.38"
## [1] "WDP imputation is 1.5"
## [1] "CHLcal imputation is 16.54612"
## [1] "SECCHI imputation is 46"
```

```r
VEL_missing <- water_interpolation %>%
  filter(is.na(VEL))

VEL_missing <- impute_data(VEL_missing, "VEL", VEL_train)
```

```
## [1] "TN imputation is 2.854"
## [1] "TP imputation is 0.197"
## [1] "TEMP imputation is 15.2"
## [1] "DO imputation is 9.2"
## [1] "TURB imputation is 36"
## [1] "COND imputation is 516"
## [1] "SS imputation is 46.3"
## [1] "WDP imputation is 5.3"
## [1] "CHLcal imputation is 16.80465"
## [1] "SECCHI imputation is 33"
```

Train and interpolate

```r
VEL_rf <- ranger(
  formula = VEL ~ .,
  data    = VEL_train %>% select(-SHEETBAR)
)

VEL_missing$predicted_VEL <- round(predict(VEL_rf,
                                     data = VEL_missing %>%
                                       select(-SHEETBAR))$predictions, 2)

VEL_missing <- VEL_missing %>% select(SHEETBAR, predicted_VEL)
```

```r
water20 <- water20 %>%
  left_join(TP_missing, by = "SHEETBAR") %>%
  left_join(TN_missing, by = "SHEETBAR") %>%
  left_join(VEL_missing, by = "SHEETBAR") %>%
  mutate(TP = case_when(is.na(TP) ~ predicted_TP,
                        TRUE ~ TP),
         TN = case_when(is.na(TN) ~ predicted_TN,
                        TRUE ~ TN),
         VEL = case_when(is.na(VEL) ~ predicted_VEL,
                        TRUE ~ VEL)) %>%
  select(-predicted_TP, -predicted_TN, -predicted_VEL)

water20 <- water20 %>%
  filter_all(all_vars(!is.na(.)))

write.csv(water20, "../../LTRM data/water_final_interpolated.csv", row.names = FALSE)
```