R Code for Analysis

Last Updated: April 14, 2021

Contents

This document contains the R code used to perform the analysis in the paper "Evaluation of a Random Forest Model to Identify Invasive Carp Eggs Based on Morphometric Features". The computer platform, R version, and R package versions used to perform this analysis can be found at the end of the document. The R markdown document that generated this document and the saved versions of the results from this document are available at https://github.com/goodekat/carp-egg-rf-validation.

```r
# Load R packages
library(dplyr)
library(forcats)
library(ggplot2)
library(MLmetrics)
library(purrr)
library(randomForest)
library(stringr)
library(tidyr)
library(tidyselect)
library(tools)

# Create a results folder if not already created
if (!dir.exists("../results")) dir.create("../results")
```

1 Data Steps

This section contains the code used to prepare the data for the analysis.

1.1 Variable Vectors

Vectors of the response and predictor variables are created to easily extract the predictor variables from the data when needed. There are two vectors of predictor variables. The first vector contains the all predictor variables (17) used to fit the random forest models. The second vector contains the reduced set of predictor variables (11) obtained from variable selection in Camacho et al. 2019[1]

```r
# Make a list of the response variables
vars_resp = c(
  "Genus",
  "Common_Name",
  "Family_ACGC",
  "Genus_ACGC",
  "Common_Name_ACGC"
)

# Make a vector of the predictor variables to be used in the models
vars_pred = c(
```

[1]Camacho, C. A., C. J. Sullivan, M. J. Weber, and C. L. Pierce. 2016. Morphological identification of Bighead Carp, Silver Carp, and Grass Carp eggs using random forests machine learning classification. North American Journal of Fisheries Management DOI:10.1002/nafm.10380.

```
  "Month",
  "Julian_Day",
  "Temperature",
  "Conductivity",
  "Larval_Length",
  "Membrane_Ave",
  "Membrane_SD",
  "Membrane_CV",
  "Embryo_to_Membrane_Ratio",
  "Embryo_Ave",
  "Embryo_SD",
  "Embryo_CV",
  "Egg_Stage",
  "Compact_Diffuse",
  "Pigment",
  "Sticky_Debris",
  "Deflated"
)

# Make a vector of the reduced predictor variables to be used in the models
vars_pred_reduced = c(
  "Membrane_Ave",
  "Embryo_Ave",
  "Deflated",
  "Membrane_SD",
  "Temperature",
  "Pigment",
  "Julian_Day",
  "Embryo_to_Membrane_Ratio",
  "Membrane_CV",
  "Conductivity",
  "Embryo_SD"
)
```

1.2 Data Cleaning

The following code loads the raw egg data (eggdata141516_raw.csv), which contains the observations from 2014 to 2016, and performs initial cleaning steps to prepare the data for model fitting. The code is based on the data cleaning code from Camacho et al. 2019. It implements the same steps taken by Camacho et al. 2019, but it has been rewritten and simplified using functions from the R package `dplyr`. The cleaning steps are as follows:

1. Subset the data to only contain necessary variables for the analysis
2. Only keep observation with identified genetics
3. Remove any observations with missing values
4. Redefine factor variables to adjust levels after observations have been removed

Also, save the cleaned data in a csv file.

```r
# Import the raw 2014-2016 egg data
eggdata_raw <- read.csv("../data/eggdata141516_raw.csv") %>%
  rename_all(
    .funs = function(.) {
      str_replace_all(., "[.]", " ") %>%
        tolower() %>%
        toTitleCase() %>%
        str_replace_all(" ", "_") %>%
        str_replace("Sd", "SD") %>%
        str_replace("Cv", "CV") %>%
        str_replace("Acgc", "ACGC") %>%
        str_replace("Temp", "Temperature")
    }
  ) %>%
  # Rename variables using yolk to embryo
  rename(
      "Embryo_to_Membrane_Ratio" = "Yolk_to_Membrane_Ratio",
      "Embryo_Ave" = "Yolk_Ave",
      "Embryo_SD" = "Yolk_SD",
      "Embryo_CV" = "Yolk_CV",
  )

# Clean the data
eggdata <-
  eggdata_raw %>%
  # Add a variable for dataset
  mutate(Dataset = ifelse(Year == "2016", "validation", "original")) %>%
  # Select only necessary variables
  select(Dataset,
         Site,
         River,
         Year,
         Questionable_Genetics,
         all_of(vars_pred),
         all_of(vars_resp)) %>%
  # Only keep observations with identified genetics
  filter(Questionable_Genetics == "NO") %>%
  # Remove any observations with missing values
  na.omit() %>%
  # Fix some mistakes in site and river levels
  mutate(Site = as.character(Site), River = as.character(River)) %>%
  mutate(Site = ifelse(Site == "DMW", "DNW", Site),
         Site = ifelse(Site == "kqa", "KQA", Site),
         River = ifelse(River == "UNR", "UMR", River),
         River = ifelse(River == "UPR", "UMR", River),
         River = ifelse(River == "DMS", "DSM", River)) %>%
  mutate(Site = factor(Site), River = factor(River)) %>%
  # Redo the factor variables to get the appropriate levels
  mutate_at(
    .vars = c(
      "Egg_Stage",
      "Compact_Diffuse",
      "Pigment",
      "Sticky_Debris",
      "Deflated",
      "Questionable_Genetics",
      all_of(vars_resp)
    ),
    .funs = factor
  )

# Save the cleaned data
write.csv(eggdata, "../data/eggdata_cleaned.csv", row.names = FALSE)
```

The structure of the data after initial cleaning is shown below.

```
str(eggdata)
```

```
## 'data.frame':    1981 obs. of  27 variables:
##  $ Dataset                  : chr  "original" "original" "original" "original" ...
##  $ Site                     : Factor w/ 21 levels "CLF","CON","DNC",..: 18 6 12 12 12 2 10 10 12 12 .
##  $ River                    : Factor w/ 7 levels "CDR","DSM","IAR",..: 6 6 3 3 3 1 2 2 2 2 ...
##  $ Year                     : int  2014 2014 2014 2014 2014 2015 2015 2015 2015 2015 ...
##  $ Questionable_Genetics    : Factor w/ 1 level "NO": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Month                    : int  7 8 6 6 8 8 6 6 5 5 ...
##  $ Julian_Day               : int  209 227 172 172 236 222 161 161 151 151 ...
##  $ Temperature              : num  24.7 25.3 26.3 26.3 25.5 26.3 23.1 23.1 19.2 19.2 ...
##  $ Conductivity             : num  522 440 473 473 498 514 654 654 674 674 ...
##  $ Larval_Length            : num  0 0 0 0 0 ...
##  $ Membrane_Ave             : num  1.43 1.24 3.77 2.84 1.42 ...
##  $ Membrane_SD              : num  0.0436 0.0291 0.1044 0.0685 0.0263 ...
##  $ Membrane_CV              : num  0.0305 0.0234 0.0277 0.0241 0.0185 ...
##  $ Embryo_to_Membrane_Ratio: num  1 0.821 0.336 0.568 0.78 ...
##  $ Embryo_Ave               : num  1.43 1.02 1.27 1.61 1.11 ...
##  $ Embryo_SD                : num  0.0436 0.092 0.0187 0.2585 0.1403 ...
##  $ Embryo_CV                : num  0.0305 0.0901 0.0148 0.1602 0.1268 ...
##  $ Egg_Stage                : Factor w/ 10 levels "1","2","3","4",..: 10 5 4 6 6 8 8 4 10 10 ...
##  $ Compact_Diffuse          : Factor w/ 2 levels "C","D": 2 1 1 1 1 1 1 1 2 2 ...
##  $ Pigment                  : Factor w/ 2 levels "N","Y": 2 2 1 1 2 1 1 1 1 1 ...
##  $ Sticky_Debris            : Factor w/ 2 levels "N","Y": 1 1 1 1 1 2 1 1 2 2 ...
##  $ Deflated                 : Factor w/ 2 levels "N","Y": 1 1 2 2 1 2 2 2 2 2 ...
##  $ Genus                    : Factor w/ 19 levels "Alosa","Aplodinotus",..: 2 2 4 4 2 16 11 11 11 11
##  $ Common_Name              : Factor w/ 31 levels "Banded Darter",..: 13 13 16 16 13 20 23 23 23 23 .
##  $ Family_ACGC              : Factor w/ 8 levels "ACGC","Catostomidae",..: 8 8 1 1 8 4 1 1 1 1 ...
##  $ Genus_ACGC               : Factor w/ 18 levels "ACGC","Alosa",..: 3 3 1 1 3 15 1 1 1 1 ...
##  $ Common_Name_ACGC         : Factor w/ 29 levels "ACGC","Banded Darter",..: 13 13 1 1 13 19 1 1 1 1
##  - attr(*, "na.action")= 'omit' Named int [1:68] 37 133 162 193 258 305 331 347 387 449 ...
##   ..- attr(*, "names")= chr [1:68] "37" "133" "162" "193" ...
```

1.3 Original Models Data (2014-2015)

The following code prepares the data for the models fit using the observations from 2014 and 2015. The steps taken are as follows:

1. Remove the variable `Questionable_Genetics`
2. Subset the data to only contain observations from 2014 and 2015
3. Redefine the factor levels to make sure the levels are correct after the removal of 2016 observations

The data is also saved as a csv file.

```
# Clean the data
eggdata1415 <- eggdata %>%
  # Remove the questionable genetics variable
  select(-Questionable_Genetics) %>%
  # Subset to only keep the 2014-2015 data
  filter(Year %in% c(2014, 2015)) %>%
  # Redo the factor variables to get the appropriate levels
  mutate_at(
    .vars = c(
      "Egg_Stage",
      "Compact_Diffuse",
      "Pigment",
      "Sticky_Debris",
      "Deflated",
      all_of(vars_resp)
    ),
    .funs = factor
  )

# Save the data
write.csv(eggdata1415, "../data/eggdata1415.csv", row.names = FALSE)
```

The structure of the 2014-2015 prepared data is shown below.

```
str(eggdata1415)
```

```
## 'data.frame':    1275 obs. of  26 variables:
##  $ Dataset                  : chr  "original" "original" "original" "original" ...
##  $ Site                     : Factor w/ 21 levels "CLF","CON","DNC",..: 18 6 12 12 12 2 10 10 12 12 .
##  $ River                    : Factor w/ 7 levels "CDR","DSM","IAR",..: 6 6 3 3 3 1 2 2 2 2 ...
##  $ Year                     : int  2014 2014 2014 2014 2014 2015 2015 2015 2015 2015 ...
##  $ Month                    : int  7 8 6 6 8 8 6 6 5 5 ...
##  $ Julian_Day               : int  209 227 172 172 236 222 161 161 151 151 ...
##  $ Temperature              : num  24.7 25.3 26.3 26.3 25.5 26.3 23.1 23.1 19.2 19.2 ...
##  $ Conductivity             : num  522 440 473 473 498 514 654 654 674 674 ...
##  $ Larval_Length            : num  0 0 0 0 0 ...
##  $ Membrane_Ave             : num  1.43 1.24 3.77 2.84 1.42 ...
##  $ Membrane_SD              : num  0.0436 0.0291 0.1044 0.0685 0.0263 ...
##  $ Membrane_CV              : num  0.0305 0.0234 0.0277 0.0241 0.0185 ...
##  $ Embryo_to_Membrane_Ratio: num  1 0.821 0.336 0.568 0.78 ...
##  $ Embryo_Ave               : num  1.43 1.02 1.27 1.61 1.11 ...
##  $ Embryo_SD                : num  0.0436 0.092 0.0187 0.2585 0.1403 ...
##  $ Embryo_CV                : num  0.0305 0.0901 0.0148 0.1602 0.1268 ...
##  $ Egg_Stage              : Factor w/ 10 levels "1","2","3","4",..: 10 5 4 6 6 8 8 4 10 10 ...
##  $ Compact_Diffuse        : Factor w/ 2 levels "C","D": 2 1 1 1 1 1 1 1 2 2 ...
##  $ Pigment                : Factor w/ 2 levels "N","Y": 2 2 1 1 2 1 1 1 1 1 ...
##  $ Sticky_Debris          : Factor w/ 2 levels "N","Y": 1 1 1 1 1 2 1 1 2 2 ...
##  $ Deflated               : Factor w/ 2 levels "N","Y": 1 1 2 2 1 2 2 2 2 2 ...
##  $ Genus                  : Factor w/ 17 levels "Alosa","Aplodinotus",..: 2 2 4 4 2 14 9 9 9 9 ...
##  $ Common_Name            : Factor w/ 29 levels "Banded Darter",..: 12 12 15 15 12 18 21 21 21 21 .
##  $ Family_ACGC            : Factor w/ 8 levels "ACGC","Catostomidae",..: 8 8 1 1 8 4 1 1 1 1 ...
##  $ Genus_ACGC             : Factor w/ 16 levels "ACGC","Alosa",..: 3 3 1 1 3 13 1 1 1 1 ...
##  $ Common_Name_ACGC       : Factor w/ 27 levels "ACGC","Banded Darter",..: 12 12 1 1 12 17 1 1 1 1
```

1.4 Augmented Models Data (2014-2016)

The code chunk below prepares the data for the models fit using the observations from 2014, 2015, and 2016. The steps taken are as follows:

1. Remove the variable `Questionable_Genetics`
2. Remove any observation with Genus or species levels that are not in 2014 and 2015
3. Redefine the factor levels to make sure the levels are correct after the removal of observations with different factor levels

The data is also saved as a csv file.

```
eggdata141516 <-
  eggdata %>%
  # Remove the questionable genetics variable
  select(-Questionable_Genetics) %>%
  # Remove observations with response variable levels not in the
  # 2014-2015 data
  filter(Genus %in% levels(eggdata1415$Genus)) %>%
  filter(Common_Name %in% levels(eggdata1415$Common_Name)) %>%
  # Redo the factor variables to get the appropriate levels
  mutate_at(
    .vars = c(
      "Egg_Stage",
      "Compact_Diffuse",
      "Pigment",
      "Sticky_Debris",
      "Deflated",
      all_of(vars_resp)
    ),
    .funs = factor
  )

# Save the data
write.csv(eggdata141516, "../data/eggdata141516.csv", row.names = FALSE)
```

The structure of the 2014-2016 prepared data is shown below.

```
str(eggdata141516)
```

```
## 'data.frame':    1978 obs. of  26 variables:
##  $ Dataset                  : chr  "original" "original" "original" "original" ...
##  $ Site                     : Factor w/ 21 levels "CLF","CON","DNC",..: 18 6 12 12 12 2 10 10 12 12 .
##  $ River                    : Factor w/ 7 levels "CDR","DSM","IAR",..: 6 6 3 3 3 1 2 2 2 2 ...
##  $ Year                     : int  2014 2014 2014 2014 2014 2015 2015 2015 2015 2015 ...
##  $ Month                    : int  7 8 6 6 8 8 6 6 5 5 ...
##  $ Julian_Day               : int  209 227 172 172 236 222 161 161 151 151 ...
##  $ Temperature              : num  24.7 25.3 26.3 26.3 25.5 26.3 23.1 23.1 19.2 19.2 ...
##  $ Conductivity             : num  522 440 473 473 498 514 654 654 674 674 ...
##  $ Larval_Length            : num  0 0 0 0 0 ...
##  $ Membrane_Ave             : num  1.43 1.24 3.77 2.84 1.42 ...
##  $ Membrane_SD              : num  0.0436 0.0291 0.1044 0.0685 0.0263 ...
##  $ Membrane_CV              : num  0.0305 0.0234 0.0277 0.0241 0.0185 ...
##  $ Embryo_to_Membrane_Ratio: num  1 0.821 0.336 0.568 0.78 ...
##  $ Embryo_Ave               : num  1.43 1.02 1.27 1.61 1.11 ...
##  $ Embryo_SD                : num  0.0436 0.092 0.0187 0.2585 0.1403 ...
##  $ Embryo_CV                : num  0.0305 0.0901 0.0148 0.1602 0.1268 ...
##  $ Egg_Stage                : Factor w/ 10 levels "1","2","3","4",..: 10 5 4 6 6 8 8 4 10 10 ...
##  $ Compact_Diffuse          : Factor w/ 2 levels "C","D": 2 1 1 1 1 1 1 1 2 2 ...
##  $ Pigment                  : Factor w/ 2 levels "N","Y": 2 2 1 1 2 1 1 1 1 1 ...
##  $ Sticky_Debris            : Factor w/ 2 levels "N","Y": 1 1 1 1 1 2 1 1 2 2 ...
##  $ Deflated                 : Factor w/ 2 levels "N","Y": 1 1 2 2 1 2 2 2 2 2 ...
##  $ Genus                    : Factor w/ 17 levels "Alosa","Aplodinotus",..: 2 2 4 4 2 14 9 9 9 9 ...
##  $ Common_Name              : Factor w/ 29 levels "Banded Darter",..: 12 12 15 15 12 18 21 21 21 21 .
##  $ Family_ACGC              : Factor w/ 8 levels "ACGC","Catostomidae",..: 8 8 1 1 8 4 1 1 1 1 ...
##  $ Genus_ACGC               : Factor w/ 16 levels "ACGC","Alosa",..: 3 3 1 1 3 13 1 1 1 1 ...
##  $ Common_Name_ACGC         : Factor w/ 27 levels "ACGC","Banded Darter",..: 12 12 1 1 12 17 1 1 1 1
```

The factor levels should be the same for both the 2014-2015 data and 2014-2016 data. The code below checks this. An error will be returned if the levels do not match.

```
# Extract factor levels from both datasets
fctlvls1415 <-
  eggdata1415 %>%
  summarise_if(.predicate = is.factor, .funs = function(.) list(levels(.)))
fctlvls141516 <-
  eggdata141516 %>%
  summarise_if(.predicate = is.factor, .funs = function(.) list(levels(.)))

# Check that the factor levels are equal and return an error if not
if (identical(fctlvls1415, fctlvls141516)) {
  print("Factor levels are the same for the 2014-2015 data and 2014-2016 data.")
} else {
  stop("Factor levels are NOT the same for the 2014-2015 data and 2014-2016 data.")
}
```

```
## [1] "Factor levels are the same for the 2014-2015 data and 2014-2016 data."
```

1.5 Validation Data (2016)

The code below extracts the 2016 observations from the cleaned 2014-2016 data. This data will be used for the validation of the model fit using the 2014-2015 data. The data is saved as a csv file.

```r
# Extract the observation from 2016 from the cleaned data
eggdata16 <- eggdata141516 %>% filter(Year == 2016)

# Save the data
write.csv(eggdata16, "../data/eggdata16.csv", row.names = FALSE)
```

2 R Functions

This section contains functions written to assist with the analysis.

2.1 Function for Fitting Random Forests

The function below can be used to fit a random forest model given the name of a response variable (`resp`), the names of predictor variables (`preds`), and a data frame that contains the response and predictor variables. It returns a random forest as an object in a named list. The function fits the random forest using 1000 trees and uses a seed of 808. This is the same seed used by Camacho et al. 2019, so that his models can be reproduced.

```r
# Function for fitting a random forest model given a resonse
# variable, predictor variables, and a dataset
fit_rf <- function(resp, preds, data) {

  # Fit the random forest
  set.seed(808)
  rf <- randomForest(
    data %>% pull(resp) ~ .,
    data = data %>% select(all_of(preds)),
    importance = T,
    ntree = 1000
  )

  # Create name for the random forest and put model in a named list
  rf_name = ifelse(length(preds) == 17, resp, paste0(resp, "_reduced"))
  rf_list = list(rf)
  names(rf_list) = rf_name

  # Return the named list
  return(rf_list)

}
```

2.2 Metric Functions

The functions below (`compute_pa`, `compute_fpr`, and `compute_prec`) compute the metrics used in the paper (performance accuracy, false positive rate, and non-target taxa accuracy) given a name of a taxa class (`category`), a vector of observed response values (`obs`), and a vector of random forest predicted values that correspond to the observed response values (`pred`).

```
# Function for computing predictive accuracy
compute_pa <- function(category, obs, pred) {
  data.frame(obs, pred) %>%
    filter(obs == category) %>%
    summarise(pa = sum(obs == pred) / n()) %>%
    as.numeric()
}

# Function for computing false positive accuracy
compute_fpr <- function(category, obs, pred) {
  data.frame(obs, pred) %>%
    filter(obs != category) %>%
    summarise(sum(pred == category) / n()) %>%
    as.numeric()
}

# Function for computing non-target taxa accuracy
compute_prec <- function(category, obs, pred) {
 data.frame(obs, pred) %>%
    filter(pred == category) %>%
    summarise(sum(obs == pred) / n()) %>%
    as.numeric()
}
```

2.3 Function for Model Validation

The function below (`compute_val_metrics`) can be used to apply the metrics of prediction accuracy, false positive rate, and non-target taxa accuracy for model validation given a random forest model, the name of the random forest model returned from `fit_rf`, and the validation data in a data frame that includes the predictor variables and response variable. This function will be used for the validation of the 2014-2015 model. The function returns a data frame with the model name, the taxonomic class, and the metrics for each class.

```
# Function for computing validation metrics
compute_val_metrics <- function(model, model_name, newdata) {

  # Extract the resonse variable from the validation data
  obs = newdata %>% pull(str_remove(model_name, "_reduced"))

  # Apply the random forest model to the validation data
  # to obtain predictions
  pred = predict(object = model, newdata = newdata)

  # Identify the taxonomic classes based on the response
  # variable used to fit the random forest
  if ("ACGC" %in% model$classes) {
    classes = "ACGC"
  } else if ("Ctenopharyngodon" %in% model$classes &
             "Hypophthalmichthys" %in% model$classes) {
    classes = c("Ctenopharyngodon", "Hypophthalmichthys")
  } else {
    classes = c("Bighead Carp", "Grass Carp", "Silver Carp")
  }

  # Compute the metrics for validation for each class and put them
  # in a data frame
  data.frame(
    model = model_name,
    class = classes,
    pa = map_dbl(.x = classes, .f = compute_pa, obs = obs, pred = pred),
    fpr = map_dbl(.x = classes, .f = compute_fpr, obs = obs, pred = pred),
    prec = map_dbl(.x = classes, .f = compute_prec, obs = obs, pred = pred)
  ) %>%
    mutate_at(.vars = c("model", "class"), .funs = as.character)

}
```

2.4 Functions for Model Performance

The function below (`compute_perf_metrics`) can be used to apply the metrics of prediction accuracy, false positive rate, and non-target taxa accuracy for model performance given a random forest model and the name of the random forest model returned from `fit_rf`. This function will be used to compute the performance of the 2014-2015 and 2014-2016 models on the training data. The function returns a data frame with the model name, the taxonomic class, and the metrics for each class.

```
# Function for computing performance metrics
compute_perf_metrics <- function(model, model_name) {

  # Extract the observed response variable
  obs = model$y

  # Obtain the training data model predictions
  pred = predict(model)

  # Identify the taxonomic classes based on the response
  # variable used to fit the random forest
  if ("ACGC" %in% model$classes) {
    classes = "ACGC"
  } else if ("Ctenopharyngodon" %in% model$classes &
             "Hypophthalmichthys" %in% model$classes) {
    classes = c("Ctenopharyngodon", "Hypophthalmichthys")
  } else {
    classes = c("Bighead Carp", "Grass Carp", "Silver Carp")
  }

  # Compute the model performance metrics for each class
  # and put them in a data frame
  data.frame(
    model = model_name,
    class = classes,
    pa = map_dbl(.x = classes, .f = compute_pa, obs = obs, pred = pred),
    fpr = map_dbl(.x = classes, .f = compute_fpr, obs = obs, pred = pred),
    prec = map_dbl(.x = classes, .f = compute_prec, obs = obs, pred = pred)
  ) %>%
    mutate_at(.vars = c("model", "class"), .funs = as.character)

}
```

3 Validation of Original Models

The validation of the 2014-2015 models is done in this section using the data from 2016.

3.1 Validation Data

This section includes the code used for computing summary statistics associated with the 2016 data.

The total number of observations per year in the data with all observations:

```
# Determine the number of observations per year in the raw data
eggdata_raw %>% count(Year)
```

```
##   Year    n
## 1 2014 1294
## 2 2015  767
## 3 2016  839
```

The total number of observations per year in the data with only successful genetic identifications:

```
# Determine the number of observations per year in the cleaned data
eggdata141516 %>% count(Year)
```

```
##   Year   n
## 1 2014 734
## 2 2015 541
## 3 2016 703
```

The number of observations and percent of the total observations per species:

```
# Determine the number and percent of observations per species
eggdata16 %>%
  count(Common_Name) %>%
  mutate(percent = round((n / sum(n))*100, 2)) %>%
  arrange(desc(percent))
```

```
##        Common_Name   n percent
## 1 Freshwater Drum 309   43.95
## 2      Grass Carp 158   22.48
## 3     Silver Carp 127   18.07
## 4  Emerald Shiner  44    6.26
## 5      Shiner sp.  37    5.26
## 6   Speckled Chub  13    1.85
## 7    Bighead Carp  11    1.56
## 8    River Shiner   3    0.43
## 9         Goldeye   1    0.14
```

3.2 Original Models (2014-2015)

The code below fits the random forest models to all response variables using the 2014-2015 data. These models are reproductions of the models from Camacho et al. 2019. By using the function `map` from the `purrr` R package to apply the `fit_rf` function, the models are returned in a list, which is saved as the file `rfs1415.rds`.

```
# Make a list of file paths for the 2014-2015 models
rfs1415_file_path = "../results/rfs1415.rds"

# Check if the 2014-2015 random forest models have been fit
if (!file.exists(rfs1415_file_path)) {

  # If not, fit the models...

  # Models with all predictor variables
  rfs1415_full = map(
    .x = vars_resp,
    .f = fit_rf,
    preds = vars_pred,
    data = eggdata1415
  ) %>% flatten()

  # Model with reduced predictor variables
  rf1415_reduced =
    fit_rf(resp = "Common_Name_ACGC",
           preds = vars_pred_reduced,
           data = eggdata1415)

  # Join the models
  rfs1415 = append(rfs1415_full, rf1415_reduced)
  # ...and save them
  saveRDS(rfs1415, rfs1415_file_path)

}

# Load the 2014-2015 models
rfs1415 = readRDS(rfs1415_file_path)
```

3.3 Predictions

The code below computes predictions for the 2016 data using the 2014-2015 models.

```
# Create a list of new datasets for making predictions on
new_data <- list(eggdata16 %>% select(all_of(vars_pred)))
new_data <- c(rep(new_data[1], 5), list(eggdata16 %>% select(all_of(vars_pred_reduced))))

# Apply the compute_val_metrics function to the
# 2014-2015 random forests and 2016 validation data
pred2016 <- pmap_dfc(
  .l = list(rfs1415, new_data),
  .f = predict,
  newdata = ,
  type = "response"
)

# Save the 2016 data RF predictions
write.csv(pred2016, "../results/pred2016.csv", row.names = FALSE)
```

3.4 Metric Computations

3.4.1 All Locations

The code below applies the function `compute_val_metrics` to the random forest models fit using the 2014-2015 data (`rfs1415`) and computes the validation metrics using the 2016 data (`eggdata16`). The results are also saved as a csv file.

```
# Apply the compute_val_metrics function to the
# 2014-2015 random forests and 2016 validation data
val_metrics <- map2_df(
  .x = rfs1415,
  .y = names(rfs1415),
  .f = compute_val_metrics,
  newdata = eggdata16
)
```

```r
# Save the validation metrics
write.csv(val_metrics, "../results/val_metrics.csv", row.names = FALSE)
```

The validation metrics are printed in the table below. The metrics were converted to percentages and rounded to 2 decimal points before being printed.

```r
# Print a table of the model validation metrics
val_metrics %>%
  mutate_if(.predicate = is.double, .funs = function(.) round(. * 100, 2)) %>%
  knitr::kable()
```

model	class	pa	fpr	prec
Genus	Ctenopharyngodon	71.52	17.43	54.33
Genus	Hypophthalmichthys	35.51	6.73	56.32
Common_Name	Bighead Carp	0.00	0.00	NaN
Common_Name	Grass Carp	75.95	21.10	51.06
Common_Name	Silver Carp	22.05	5.90	45.16
Family_ACGC	ACGC	88.51	2.70	95.97
Genus_ACGC	ACGC	90.20	2.95	95.70
Common_Name_ACGC	ACGC	90.88	3.69	94.72
Common_Name_ACGC_reduced	ACGC	93.92	3.19	95.53

3.4.2 Old and New Locations Separated

The code below identifies locations (`Site` and `River` combinations) that were only sampled in 2016.

```r
new2016sites <-
  eggdata141516 %>%
  count(Site, River, Year) %>%
  pivot_wider(names_from = Year, values_from = n) %>%
  mutate(new2016 = ifelse(is.na(`2014`) + is.na(`2015`) == 2, TRUE, FALSE)) %>%
  filter(new2016 == TRUE) %>%
  select(Site, River)
```

The following code computes the validation metrics using the function `compute_val_metrics` for the locations in the 2016 data that were sampled previously (`val_metrics_old_sites`) and locations first sampled in 2016 (`val_metrics_old_sites`). These metrics are joined and saved as a csv file.

```r
# Separate the old and new sites
eggdata16_old_sites <- eggdata16 %>% anti_join(new2016sites, by = c("Site", "River"))
eggdata16_new_sites <- eggdata16 %>% right_join(new2016sites, by = c("Site", "River"))
```

```r
# Compute validation metrics for previously sampled locations
val_metrics_old_sites <- map2_df(
  .x = rfs1415,
  .y = names(rfs1415),
  .f = compute_val_metrics,
  newdata = eggdata16_old_sites
)

# Compute validation metrics for new locations in 2016
val_metrics_new_sites <- map2_df(
  .x = rfs1415,
  .y = names(rfs1415),
  .f = compute_val_metrics,
  newdata = eggdata16_new_sites
)

# Join the metrics with a variable site_type indicating whether the
# location is 'new' or 'old'
val_metrics_separate <-
  val_metrics_new_sites %>%
  mutate(site_type = "new") %>%
  bind_rows(val_metrics_old_sites %>% mutate(site_type = "old")) %>%
  select(site_type, everything())

# Save the validation metrics
write.csv(val_metrics_separate, "../results/val_metrics_separate.csv", row.names = FALSE)
```

The two sets of validation metrics are printed in the table below. The metrics were converted to percentages and rounded to 2 decimal points before being printed.

```r
# Print a table of the model validation metrics
val_metrics_separate %>%
  mutate_if(.predicate = is.double, .funs = function(.) round(. * 100, 2)) %>%
  knitr::kable()
```

site_type	model	class	pa	fpr	prec
new	Genus	Ctenopharyngodon	43.48	1.44	83.33
new	Genus	Hypophthalmichthys	NaN	7.41	0.00
new	Common_Name	Bighead Carp	NaN	0.00	NaN
new	Common_Name	Grass Carp	43.48	1.44	83.33
new	Common_Name	Silver Carp	NaN	7.41	0.00
new	Family_ACGC	ACGC	8.70	2.16	40.00
new	Genus_ACGC	ACGC	21.74	2.16	62.50
new	Common_Name_ACGC	ACGC	30.43	2.88	63.64
new	Common_Name_ACGC_reduced	ACGC	69.57	2.16	84.21
old	Genus	Ctenopharyngodon	76.30	22.91	52.55
old	Genus	Hypophthalmichthys	35.51	6.45	65.33
old	Common_Name	Bighead Carp	0.00	0.00	NaN
old	Common_Name	Grass Carp	81.48	27.83	49.33

site_type	model	class	pa	fpr	prec
old	Common_Name	Silver Carp	22.05	5.31	56.00
old	Family_ACGC	ACGC	95.24	2.99	97.01
old	Genus_ACGC	ACGC	95.97	3.36	96.68
old	Common_Name_ACGC	ACGC	95.97	4.10	95.97
old	Common_Name_ACGC_reduced	ACGC	95.97	3.73	96.32

4 Augmented Models

This section contains the code that fit computes performance metrics for the 2014-2015 models and 2014-2016 models.

4.1 Augmented Models (2014-2016)

The code below fits the random forest models to all response variables using the 2014-2016 data. Again, the models are returned in a list, which is saved as the file `rfs141516.rds`.

```
# Make a list of file paths for the 2014-2015 models
rfs141516_file_path = "../results/rfs141516.rds"

# Check if the 2014-2015 random forest models have been fit
if (!file.exists(rfs141516_file_path)) {

  # If not, fit the models...

  # Models with all predictor variables
  rfs141516_full = map(
    .x = vars_resp,
    .f = fit_rf,
    preds = vars_pred,
    data = eggdata141516
  ) %>% flatten()

  # Model with reduced predictor variables
  rf141516_reduced =
    fit_rf(resp = "Common_Name_ACGC",
           preds = vars_pred_reduced,
           data = eggdata141516)

  # Join the models
  rfs141516 = append(rfs141516_full, rf141516_reduced)

  # ...and save them
  saveRDS(rfs141516, rfs141516_file_path)

}

# Load the 2014-2015 models
rfs141516 = readRDS(rfs141516_file_path)
```

4.2 Metric Computations

The code below applies the function `compute_perf_metrics` to the 2014-2015 random forest models to compute the performance metrics. The results are saved in a csv file.

```
# Compute the model performance metrics for the 2014-2015 models
metrics1415 <- map2_df(.x = rfs1415, .y = names(rfs1415), .f = compute_perf_metrics)

# Save the 2014-2015 model performance metrics
write.csv(metrics1415, "../results/perf_metrics1415.csv", row.names = FALSE)
```

The following below applies the function `compute_perf_metrics` to the 2014-2016 random forest models to compute the performance metrics. The results are saved in a csv file.

```
# Compute the model performance metrics for the 2014-2016 models
metrics141516 <- map2_df(.x = rfs141516, .y = names(rfs141516), .f = compute_perf_metrics)

# Save the 2014-2016 model performance metrics
write.csv(metrics141516, "../results/perf_metrics141516.csv", row.names = FALSE)
```

4.3 Camacho Paper Metrics

Below, a data frame is created that contains the model performance metrics directly copied from Camacho et al. 2019 to check that the 2014-2015 models fit in this document accurately reproduce those from Camacho et al. 2019.

```
# Create a dataframe with the performance metrics from the Camacho et al. 2019 paper
metrics1415_paper <-
  data.frame(
    model = c(
      "Genus",
      "Genus",
      "Common_Name",
      "Common_Name",
      "Common_Name",
      "Family_ACGC",
      "Genus_ACGC",
      "Common_Name_ACGC",
      "Common_Name_ACGC_reduced"
    ),
    class = c(
      "Ctenopharyngodon",
      "Hypophthalmichthys",
      "Bighead Carp",
      "Grass Carp",
      "Silver Carp",
      "ACGC",
      "ACGC",
      "ACGC",
      "ACGC"
    ),
    pa_paper = c(68, 88, 8, 73, 88, 97, 97, 97, 98),
    fpr_paper = c(3, 7, 0, 3, 7, 4, 4, 5, 5),
    prec_paper = c(72, 83, 50, 71, 82, 94, 93, 93, 93)
  ) %>%
  mutate_at(.vars = c("model", "class"), .funs = as.character)
```

4.4 Metric Table

The code below joins the model performance metrics from the 2014-2015 and 2014-2016 models computed in this paper and the metrics from the Camacho et al. 2019 paper.

```
# Join the model performance metrics into one data frame
metrics_joined <- metrics1415 %>%
  rename("pa_1415" = "pa",
         "fpr_1415" = "fpr",
         "prec_1415" = "prec") %>%
  left_join(
    metrics141516 %>%
      rename(
        "pa_141516" = "pa",
        "fpr_141516" = "fpr",
        "prec_141516" = "prec"
      ),
    by = c("model", "class")
  ) %>%
  mutate_if(
    .predicate = is.double,
    .funs = function(.)
      round(. * 100, 0)
  ) %>%
  left_join(
    metrics1415_paper %>%
      rename(
        "pa_1415.paper" = "pa_paper",
        "fpr_1415.paper" = "fpr_paper",
        "prec_1415.paper" = "prec_paper"
      ),
    by = c("model", "class")
  )
```

The code below prints tables of the model performance metrics separated by metric.

```
# Print the prediction accuracy model performance metrics
metrics_joined %>%
  select(model, class, pa_1415.paper, pa_1415, pa_141516) %>%
  knitr::kable()
```

model	class	pa_1415.paper	pa_1415	pa_141516
Genus	Ctenopharyngodon	68	67	67
Genus	Hypophthalmichthys	88	88	81
Common_Name	Bighead Carp	8	8	0
Common_Name	Grass Carp	73	71	71
Common_Name	Silver Carp	88	89	80
Family_ACGC	ACGC	97	96	97
Genus_ACGC	ACGC	97	97	98
Common_Name_ACGC	ACGC	97	98	98
Common_Name_ACGC_reduced	ACGC	98	98	98

```
# Print the false positive rate model performance metrics
metrics_joined %>%
  select(model, class, fpr_1415.paper, fpr_1415, fpr_141516) %>%
  knitr::kable()
```

model	class	fpr_1415.paper	fpr_1415	fpr_141516
Genus	Ctenopharyngodon	3	3	6
Genus	Hypophthalmichthys	7	7	8
Common_Name	Bighead Carp	0	0	0
Common_Name	Grass Carp	3	3	6
Common_Name	Silver Carp	7	7	8
Family_ACGC	ACGC	4	4	4
Genus_ACGC	ACGC	4	5	4
Common_Name_ACGC	ACGC	5	5	4
Common_Name_ACGC_reduced	ACGC	5	5	4

```
# Print the non-target taxa accuracy model performance metrics
metrics_joined %>%
  select(model, class, prec_1415.paper, prec_1415, prec_141516) %>%
  knitr::kable()
```

model	class	prec_1415.paper	prec_1415	prec_141516
Genus	Ctenopharyngodon	72	70	67
Genus	Hypophthalmichthys	83	82	78
Common_Name	Bighead Carp	50	50	0
Common_Name	Grass Carp	71	71	65
Common_Name	Silver Carp	82	82	75
Family_ACGC	ACGC	94	94	95
Genus_ACGC	ACGC	93	93	94
Common_Name_ACGC	ACGC	93	92	94
Common_Name_ACGC_reduced	ACGC	93	92	94

5 Summary Statistics

The following code computes the average and standard error for the variables of average membrane diameter and average embryo diameter for the species of Bighead, Grass, and Silver Carp.

```
eggdata141516 %>%
  filter(Common_Name %in% c("Bighead Carp", "Grass Carp", "Silver Carp")) %>%
  group_by(Common_Name) %>%
  summarise(mean_mem_ave = round(mean(Membrane_Ave), 1),
            se_mem_ave = round(sd(Membrane_Ave) / n(), 3),
            mean_embryo_ave = round(mean(Embryo_Ave), 1),
            se_embryo_ave = round(sd(Embryo_Ave) / n(), 3))
```

```
## # A tibble: 3 x 5
##   Common_Name  mean_mem_ave se_mem_ave mean_embryo_ave se_embryo_ave
##   <fct>               <dbl>      <dbl>           <dbl>         <dbl>
## 1 Bighead Carp          3.7      0.03              2           0.021
## 2 Grass Carp            3.6      0.002             1.6         0.001
## 3 Silver Carp           3.1      0.002             1.7         0.001
```

6 Session Info

The computer platform, R version, and R package version used to perform this analysis are printed below.

```
sessionInfo()
```

```
## R version 4.0.4 (2021-02-15)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] tools     stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] tidyselect_1.1.0    tidyr_1.1.3         stringr_1.4.0
## [4] randomForest_4.6-14 purrr_0.3.4         MLmetrics_1.1.1
## [7] ggplot2_3.3.3       forcats_0.5.1       dplyr_1.0.5
##
## loaded via a namespace (and not attached):
##  [1] highr_0.8         pillar_1.5.1      compiler_4.0.4    digest_0.6.27
##  [5] evaluate_0.14     lifecycle_1.0.0   tibble_3.1.0      gtable_0.3.0
##  [9] pkgconfig_2.0.3   rlang_0.4.10      rstudioapi_0.13   cli_2.3.1
## [13] DBI_1.1.1         yaml_2.2.1        xfun_0.22         withr_2.4.1
## [17] knitr_1.31        generics_0.1.0    vctrs_0.3.7       grid_4.0.4
## [21] glue_1.4.2        R6_2.5.0          fansi_0.4.2       rmarkdown_2.7
## [25] magrittr_2.0.1    scales_1.1.1      ellipsis_0.3.1    htmltools_0.5.1.1
## [29] assertthat_0.2.1  colorspace_2.0-0  utf8_1.2.1        stringi_1.5.3
## [33] munsell_0.5.0     crayon_1.4.1
```