Code for Applying Augmented Models to New Data

Last Updated: September 15, 2020

Contents

This document accompanies the paper "Evaluation of a Random Forest Model to Identify Invasive Carp Eggs Based on Morphometric Features". It contains code for accessing the augmented models with invasive carp treated as one species. Users can use the function `carp_rf_predict` to input a set of their own morphometric variable values to obtain predictions from the models.

1 Augmented Random Forest Models

The code below loads the augmented random forest models (fit using data from 2014-2016), which are stored in the RDS object `rfs141516.rds` as a list. The four models where Bighead, Grass, and Silver Carp are treated as a single classification category (at the levels of species, family, and genus) are extracted and stored as separate objects. Note that ACGC is the category name for invasive carp.

```r
# Load the augmented random forest models
rfs141516 <- readRDS("../results/rfs141516.rds")

# Extract and separate the four models where invasive carp
# is one classification category
rf_species <- rfs141516$Common_Name_ACGC
rf_species_reduced <- rfs141516$Common_Name_ACGC_reduced
rf_genus <- rfs141516$Genus_ACGC
rf_family <- rfs141516$Family_ACGC
```

2 Structure of the Morphometric Variable Data Frame

In order to be able to use one of the random forest models for prediction, the new data must contain the morphometric predictor variables listed in the table below where each column in the data frame contains the entries for one of the variables. The variable names must be the same as those listed below (case sensitive), and the variable types also must be the same. If the objective is to use the random forest `rf_species_reduced`, then only the variables of `Membrane_Ave`, `Yolk_Ave`, `Deflated`, `Membrane_SD`, `Temperature`, `Pigment`, `Julian_Day`, `Yolk_to_Membrane_Ratio`, `Membrane_CV`, `Conductivity`, and `Yolk_SD` are needed.

Variable	Type	Description
`Month`	`int`	month when the egg was collected (specified as an integer where January is 1 and December is 12)
`Julian_Day`	`int`	Julian day on which the egg was collected
`Temperature`	`num`	temperature (C) of the water during the time of collection
`Larval_Length`	`num`	late stage embryo midline length (mm)
`Membrane_Ave`	`num`	average membrane diameter (mm)
`Membrane_SD`	`num`	membrane diameter standard deviation (mm)
`Membrane_CV`	`num`	membrane diameter coefficient of variation (mm)
`Yolk_to_Membrane_Ratio`	`num`	perivitelline space index
`Yolk_Ave`	`num`	average embryo diameter (mm)
`Yolk_SD`	`num`	embryo diameter standard deviation (mm)
`Yolk_CV`	`num`	embryo diameter coefficient of variation (mm)
`Egg_Stage`	`Factor`	development stage of the egg (1, 2, 3, 4, 5, 6, 7, 8, BROKEN, or D)
`Compact_Diffuse`	`Factor`	indicates whether the egg was compact or diffuse (C or D)
`Pigment`	`Factor`	indicates whether the egg had pigment or not (Y or N)
`Sticky_Debris`	`Factor`	indicates whether the egg had debris on it or not (Y or N)
`Deflated`	`Factor`	indicates whether the egg was deflated or not (Y or N)

An example of such a data frame is created below, and the structure is printed.

```r
# Create an example data frame
example_vars <-
  data.frame(
    Month = as.integer(c(4, 5)),
    Julian_Day = as.integer(c(120, 131)),
    Temperature = c(16, 16.8),
    Conductivity = c(435, 367),
    Larval_Length = c(0, 0),
    Membrane_Ave = c(1.39305, 1.31183),
    Membrane_SD = c(0.12109, 0.18596),
    Membrane_CV = c(0.08692, 0.14176),
    Yolk_to_Membrane_Ratio = c(0.57986, 0.14176),
    Yolk_Ave = c(0.80777, 1.06952),
    Yolk_SD = c(0.1685, 0.1955),
    Yolk_CV = c(0.2086, 0.1828),
    Egg_Stage = factor(c(1, 4)),
    Compact_Diffuse = factor(c("C", "C")),
    Pigment = factor(c("Y", "Y")),
    Sticky_Debris = factor(c("N", "N")),
    Deflated = factor(c("N", "N"))
  )

# Print the structure of the data frame
str(example_vars)
```

```
## 'data.frame':    2 obs. of  17 variables:
##  $ Month                 : int  4 5
##  $ Julian_Day            : int  120 131
##  $ Temperature           : num  16 16.8
##  $ Conductivity          : num  435 367
##  $ Larval_Length         : num  0 0
##  $ Membrane_Ave          : num  1.39 1.31
##  $ Membrane_SD           : num  0.121 0.186
##  $ Membrane_CV           : num  0.0869 0.1418
##  $ Yolk_to_Membrane_Ratio: num  0.58 0.142
##  $ Yolk_Ave              : num  0.808 1.07
##  $ Yolk_SD               : num  0.169 0.196
##  $ Yolk_CV               : num  0.209 0.183
##  $ Egg_Stage             : Factor w/ 2 levels "1","4": 1 2
##  $ Compact_Diffuse       : Factor w/ 1 level "C": 1 1
##  $ Pigment               : Factor w/ 1 level "Y": 1 1
##  $ Sticky_Debris         : Factor w/ 1 level "N": 1 1
##  $ Deflated              : Factor w/ 1 level "N": 1 1
```

3 Factor Levels

In order to be able to use one of the random forest models to make predictions, the factor variables in the data frame must have matching levels as those used to train the model. The function below adjusts the factor levels in the data frame to match those of the data used to train the model. If a level is in one of the factors in the new data that was not in the training data, an error will be returned.

```
# Function for adjusting the factor levels of the data frame with the
# morphometric variables as needed
adjust_factor_levels <- function(df) {

  # Create factors with levels
  es_levels = c("1", "2", "3", "4", "5", "6", "7", "8", "BROKEN", "D")
  cd_levels = c("C", "D")
  yn_levels = c("N", "Y")

  # Check if the levels are correct and change if not (stop if a
  # level that is not in the training data is found)
  if (sum(levels(df$Egg_Stage) != es_levels) > 0) {
    if (!(sum(levels(df$Egg_Stage) %in% es_levels))) {
      stop ("Level in Egg_Stage that is not in training data.")
    }
    df$Egg_Stage <- factor(df$Egg_Stage,levels = es_levels)
  }
  if (sum(levels(df$Compact_Diffuse) != cd_levels) > 0) {
    if (!(sum(levels(df$Compact_Diffuse) %in% cd_levels))) {
      stop ("Level in Compact_Diffuse that is not in training data.")
    }
    df$Compact_Diffuse <- factor(df$Compact_Diffuse, levels = cd_levels)
  }
  if (sum(levels(df$Pigment) != yn_levels) > 0) {
    if (!(sum(levels(df$Pigment) %in% yn_levels))) {
      stop ("Level in Pigment that is not in training data.")
    }
    df$Pigment <- factor(df$Pigment, levels = yn_levels)
  }
  if (sum(levels(df$Sticky_Debris) != yn_levels) > 0) {
    if (!(sum(levels(df$Sticky_Debris) %in% yn_levels))) {
     stop ("Level in Sticky_Debris that is not in training data.")
    }
    df$Sticky_Debris <- factor(df$Sticky_Debris, levels = yn_levels)
  }
  if (sum(levels(df$Deflated) != yn_levels) > 0) {
    if (!(sum(levels(df$Deflated) %in% yn_levels))) {
      stop ("Level in Deflated that is not in training data.")
    }
    df$Deflated <- factor(df$Deflated, levels = yn_levels)
  }

  # Return the data frame
  return(df)

}
```

Input the data frame with the predictor variables into this function to return a data frame with the correct factor levels as is done in the code below.

```
# Use the function to adjust the factor variable levels as needed
example_vars_adj <- adjust_factor_levels(example_vars)
```

4 Predictions

Finally, the `predict` function from the `randomForest` package can be used to make predictions given one of the augmented models and the data frame with morphometric variables. All of the options in the `predict` function can be utilized. Several example are included below.

```
# Load the randomForest package
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```
# Predict the species
predict(rf_species, example_vars_adj)

##               1               2
## Freshwater Drum Freshwater Drum
## 27 Levels: ACGC Banded Darter Bigmouth Buffalo Black Buffalo ... White Bass
```

```
# Return the random forest probability associated with each species
# for both observations (the column ACGC contains the probabilities
# that the egg is an invasive carp)
predict(rf_species, example_vars_adj, type = "prob")

##   ACGC Banded Darter Bigmouth Buffalo Black Buffalo Buffalo sp. Carpsuckers sp.
## 1 0.01             0            0.016         0.001       0.005               0
## 2 0.05             0            0.000         0.000       0.000               0
##   Channel Shiner Common Logperch Common Shiner Emerald Shiner Fathead Minnow
## 1          0.001            0.01             0          0.049          0.000
## 2          0.000            0.00             0          0.088          0.006
##   Freshwater Drum Gizzard Shad Goldeye Quillback River Carpsucker River Shiner
## 1           0.806            0       0         0            0.039        0.000
## 2           0.727            0       0         0            0.044        0.002
##   Sand Shiner Shiner sp. Silver Chub Skipjack Shad Smallmouth Buffalo
## 1           0      0.021       0.001         0.003                  0
## 2           0      0.033       0.013         0.008                  0
##   Speckled Chub Spotfin Shiner Striped Bass Walleye White Bass
## 1         0.005              0        0.002   0.031      0.000
## 2         0.026              0        0.002   0.000      0.001
## attr(,"class")
## [1] "matrix" "array"  "votes"
```

```
# Predict the species using the reduced variables model
predict(rf_species_reduced, example_vars_adj)
```

```
##               1               2
## Freshwater Drum Freshwater Drum
## 27 Levels: ACGC Banded Darter Bigmouth Buffalo Black Buffalo ... White Bass
```

```
# Predict the genus and return the random forest probabilities
# for each species (again, ACGC represents the invasive carp category)
predict(rf_genus, example_vars_adj, type = "prob")
```

```
##    ACGC Alosa Aplodinotus Carpiodes Cyprinella Dorosoma Etheostoma Hiodon
## 1 0.008 0.001       0.829     0.044          0        0          0  0.001
## 2 0.047 0.007       0.750     0.038          0        0          0  0.000
##   Ictiobus Luxilus  Macrhybopsis Morone Notropis Percina Pimephales Sander
## 1    0.017        0        0.003  0.001    0.063   0.016      0.000  0.017
## 2    0.000        0        0.051  0.002    0.090   0.001      0.014  0.000
## attr(,"class")
## [1] "matrix" "array"  "votes"
```

```
# Predict the family and return the random forest probabilities
predict(rf_family, example_vars_adj, type = "prob")
```

```
##    ACGC Catostomidae Clupeidae Cyprinidae Hiodontidae Moronidae Percidae
## 1 0.011        0.051     0.000      0.060       0.002     0.002    0.031
## 2 0.044        0.042     0.008      0.131       0.001     0.003    0.001
##   Sciaenidae
## 1      0.843
## 2      0.770
## attr(,"class")
## [1] "matrix" "array"  "votes"
```