R Code for Figures

Last Updated: February 15, 2021

Contents

This document contains the R code used to create figures for 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 figures from this document are available at https://github.com/goodekat/carp-egg-rf-validation.

```
# Load R packages
library(cowplot)
library(dplyr)
library(forcats)
library(ggplot2)
library(pdp)
library(purrr)
library(randomForest)
library(stringr)
library(tidyr)

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

```
# Specify the (base) font size (fs), font (ff), and dpi (fdpi) for all plots
fs = 10
ff = "Times"
fdpi = 500
```

1 Data

The data that will be used in this document is loaded below.

```
# Load the egg measurement datasets
eggdata16 <- read.csv("../results/eggdata16.csv")
eggdata1415 <- read.csv("../results/eggdata1415.csv")
eggdata141516 <- read.csv("../results/eggdata141516.csv")

# Load the predictions for the validation data
pred2016 <- read.csv("../results/pred2016.csv")

# Load the model validation metrics
val_metrics <- read.csv("../results/val_metrics.csv")
val_metrics_separate <- read.csv("../results/val_metrics_separate.csv")

# Load the model performance metrics
metrics1415 <- read.csv("../results/perf_metrics1415.csv")
metrics141516 <- read.csv("../results/perf_metrics141516.csv")

# Load the random forest models
rfs1415 <- readRDS("../results/rfs1415.rds")
rfs141516 <- readRDS("../results/rfs141516.rds")
```

Vectors containing the response and predictor variables are created below and used throughout the code in this document.

```
# 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(
  "Month",
  "Julian_Day",
  "Temperature",
  "Conductivity",
  "Larval_Length",
```

```
  "Membrane_Ave",
  "Membrane_SD",
  "Membrane_CV",
  "Yolk_to_Membrane_Ratio",
  "Yolk_Ave",
  "Yolk_SD",
  "Yolk_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",
  "Yolk_Ave",
  "Deflated",
  "Membrane_SD",
  "Temperature",
  "Pigment",
  "Julian_Day",
  "Yolk_to_Membrane_Ratio",
  "Membrane_CV",
  "Conductivity",
  "Yolk_SD"
)
```

The code below makes sure the data has the same structure as when the models were trained.

```
# Specify variable types for 2016 data
eggdata16 <- eggdata16 %>%
  mutate(Dataset = as.character(Dataset)) %>%
  mutate_at(
    .vars = c("Site", "River", "Egg_Stage", "Compact_Diffuse",
              "Pigment", "Sticky_Debris", "Deflated", all_of(vars_resp)),
    .funs = factor
  ) %>%
  mutate_at(
    .vars = c("Year", "Month", "Julian_Day"),
    .funs = as.integer
  ) %>%
  mutate_at(
    .vars = c("Temperature", "Conductivity", "Larval_Length", "Membrane_Ave",
              "Membrane_SD", "Membrane_CV", "Yolk_to_Membrane_Ratio", "Yolk_Ave",
              "Yolk_SD", "Yolk_CV"),
    .funs = as.double
  )

# Specify variable types for 2014-2015 data
```

```
eggdata1415 <- eggdata1415 %>%
  mutate(Dataset = as.character(Dataset)) %>%
  mutate_at(
    .vars = c("Site", "River", "Egg_Stage", "Compact_Diffuse",
              "Pigment", "Sticky_Debris", "Deflated", all_of(vars_resp)),
    .funs = factor
  ) %>%
  mutate_at(
    .vars = c("Year", "Month", "Julian_Day"),
    .funs = as.integer
  ) %>%
  mutate_at(
    .vars = c("Temperature", "Conductivity", "Larval_Length", "Membrane_Ave",
              "Membrane_SD", "Membrane_CV", "Yolk_to_Membrane_Ratio", "Yolk_Ave",
              "Yolk_SD", "Yolk_CV"),
    .funs = as.double
  )

# Specify variable types for 2014-2016 data
eggdata141516 <- eggdata141516 %>%
  mutate(Dataset = as.character(Dataset)) %>%
  mutate_at(
    .vars = c("Site", "River", "Egg_Stage", "Compact_Diffuse",
              "Pigment", "Sticky_Debris", "Deflated", all_of(vars_resp)),
    .funs = factor
  ) %>%
  mutate_at(
    .vars = c("Year", "Month", "Julian_Day"),
    .funs = as.integer
  ) %>%
  mutate_at(
    .vars = c("Temperature", "Conductivity", "Larval_Length", "Membrane_Ave",
              "Membrane_SD", "Membrane_CV", "Yolk_to_Membrane_Ratio", "Yolk_Ave",
              "Yolk_SD", "Yolk_CV"),
    .funs = as.double
  )
```

2 Validation of Original Models

This section contains the code corresponding to the visualizations in the paper for the validation of the original models.

2.1 Figure 3: Metrics for All Locations

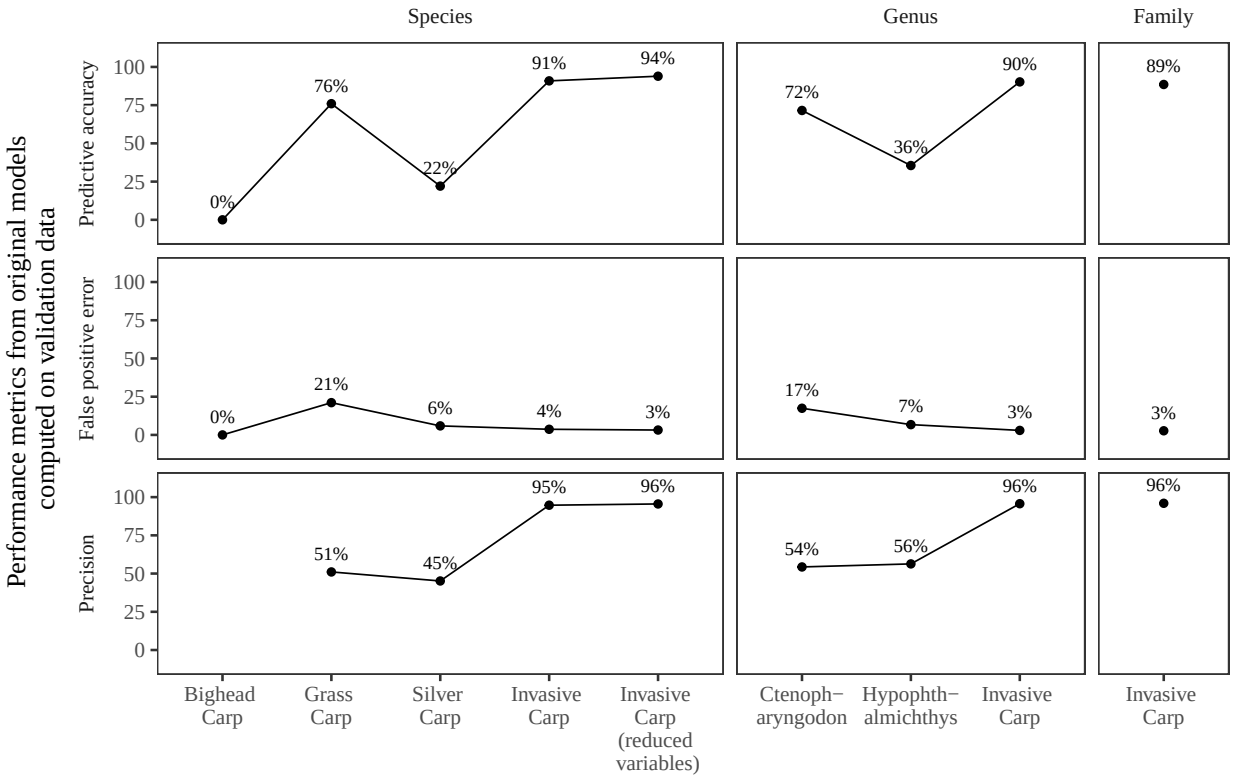
This section contains visualizations of the validation metrics. The code below loads and prepares the model validation metrics to be used for plotting.

```
# Prepare model validation data for plotting
val_plot_data <-
```

```
val_metrics %>%
pivot_longer(
  names_to = "metric",
  values_to = "value",
  cols = c("pa", "fpr", "prec")
) %>%
mutate(
  model_class = factor(class):factor(model),
  model_type = ifelse(
    str_detect(model_class, "ACGC"),
    "Grouped",
    "Separate"
  )
) %>%
mutate(
  taxa = fct_recode(
    model,
    "Genus" = "Genus",
    "Species" = "Common_Name",
    "Family" = "Family_ACGC",
    "Genus" = "Genus_ACGC",
    "Species" = "Common_Name_ACGC",
    "Species" = "Common_Name_ACGC_reduced"
  ),
  model_class = fct_recode(
    model_class,
    "Bighead \nCarp" = "Bighead Carp:Common_Name",
    "Grass \nCarp" = "Grass Carp:Common_Name",
    "Silver \nCarp" = "Silver Carp:Common_Name",
    "Ctenoph-\naryngodon" = "Ctenopharyngodon:Genus",
    "Hypophth-\nalmichthys" = "Hypophthalmichthys:Genus",
    "Invasive \nCarp" = "ACGC:Common_Name_ACGC",
    "Invasive \nCarp \n(reduced \nvariables)" =
      "ACGC:Common_Name_ACGC_reduced",
    "Invasive \nCarp" = "ACGC:Genus_ACGC",
    "Invasive \nCarp" = "ACGC:Family_ACGC"
  ),
  metric = fct_recode(
    metric,
    "Predictive accuracy" = "pa",
    "False positive error" = "fpr",
    "Precision" = "prec"
  )
) %>%
mutate(
  metric = fct_relevel(
    metric,
    "Predictive accuracy",
    "False positive error",
    "Precision",
  ),
  taxa = fct_relevel(taxa, "Species", "Genus", "Family"),
  model_class = fct_relevel(
    model_class,
```

```
      "Bighead \nCarp",
      "Grass \nCarp",
      "Silver \nCarp",
      "Ctenoph-\naryngodon",
      "Hypophth-\nalmichthys",
      "Invasive \nCarp",
      "Invasive \nCarp \n(reduced \nvariables)"
    ),
    model_type =
      fct_relevel(
        model_type,
        "Separate",
        "Grouped"
        )
  )



# Create plot of validation metrics faceted by metric and taxonomic levels
val_metric_plot <-
  ggplot(val_plot_data,
       aes(
         x = model_class,
         y = value * 100,
         group = taxa
       )) +
  geom_point(size = 1) +
  geom_line(size = 0.3) +
  geom_text(
    aes(label = paste0(round(value * 100, 0), "%")),
    nudge_y = 12,
    family = ff,
    size = 2.5
  ) +
  facet_grid(
    metric ~ taxa,
    switch = "y",
    scales = "free",
    space = "free_x"
  ) +
  theme_bw(base_size = fs, base_family = ff) +
  theme(
    strip.placement = "outside",
    strip.background = element_rect(color = "white", fill = "white"),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  ) +
  labs(
    x = "Model response variable (taxonomic level; top labels) and \nclassification category (bottom lab
    y = "Performance metrics from original models \ncomputed on validation data",
    color = "Taxonomic level"
  ) +
  scale_y_continuous(breaks = seq(0, 100, 25), limits = c(-10, 110))
```

```
# Print the figure
val_metric_plot
```




```
# Save the figure
ggsave(
  filename = "../figures/figure03.png",
  plot = val_metric_plot,
  height = 4.25,
  width = 6.5,
  dpi = fdpi
)
```

2.2 Figure 4: Confusion Matrices

The code below contains a function `plot_confus` that creates a plot of a confusion matrix given a vector of observed response variable values (`obs`) and a corresponding vector of model predictions (`pred`).

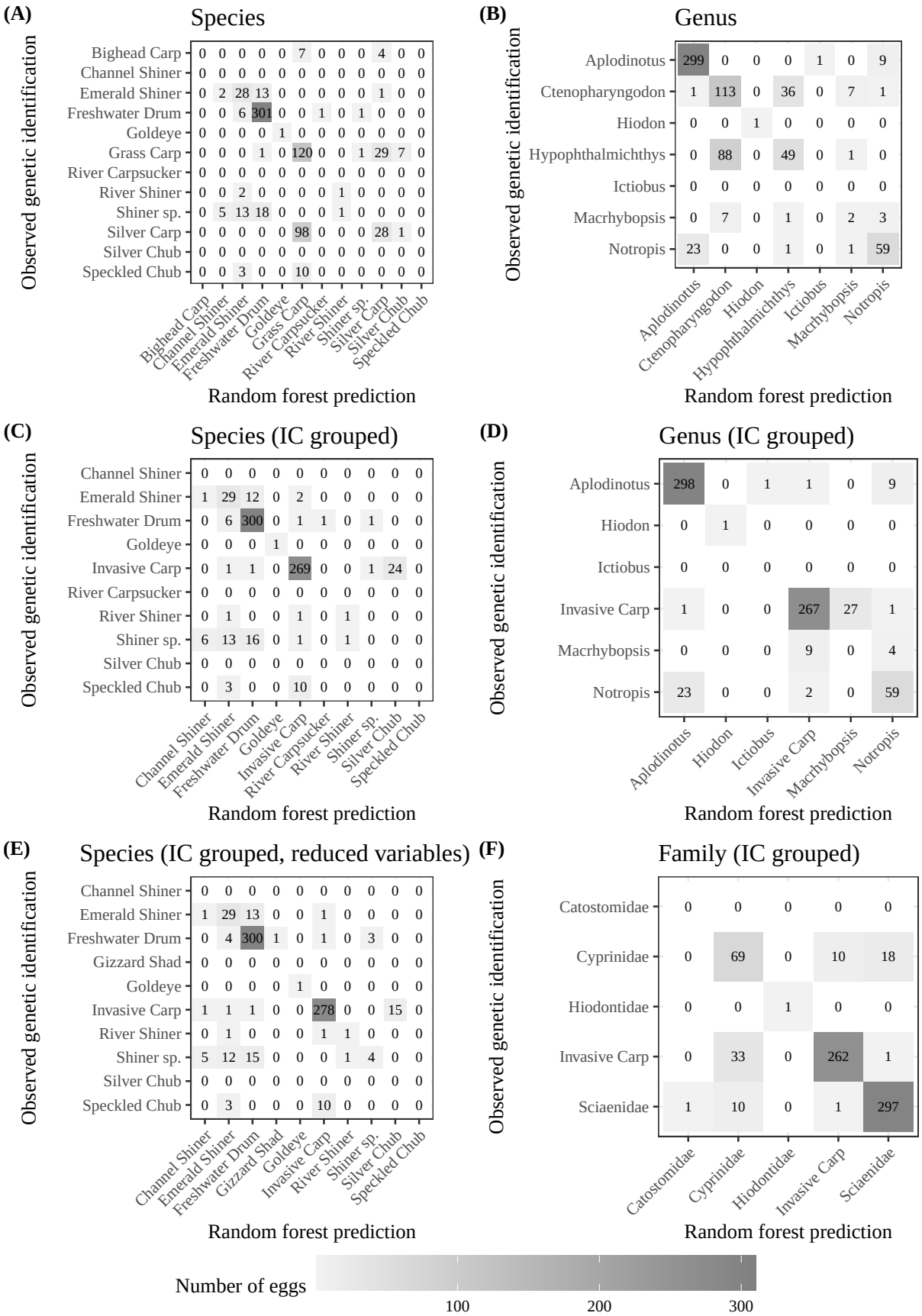
```
# Function for creating a plot of a confusion matrix
plot_confus <- function(obs, pred) {

  # Convert obs and pred to character strings
  obs = as.character(obs)
```

```
  pred = as.character(pred)

  # Change ACGC to IC for plotting
  obs = ifelse(obs == "ACGC", "Invasive Carp", obs)
  pred = ifelse(pred == "ACGC", "Invasive Carp", pred)

  # Identify the factor levels
  lev = unique(c(obs, pred))

  # Create the plot
  data.frame(obs, pred) %>%
    count(obs, pred) %>%
    right_join(expand.grid(obs = lev, pred = lev), by = c("obs", "pred")) %>%
    mutate(n = ifelse(is.na(n), 0, n)) %>%
    mutate(obs = factor(obs, levels = rev(sort(lev))),
           pred = factor(pred, levels = sort(lev))) %>%
    ggplot(aes(
      x = pred,
      y = obs,
      fill = n,
      label = n
    )) +
    geom_tile() +
    geom_text(alpha = 1, family = ff, size = 2.5) +
    theme_bw(base_size = fs, base_family = ff) +
    scale_fill_gradient(
      low = "grey95",
      high = "grey50",
      limits = c(1, 310),
      na.value = 'white'
    ) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          aspect.ratio = 1) +
    labs(x = "Random forest prediction",
         y = "Observed genetic identification",
         fill = "Number of eggs")

}
```

The code below applies the function `plot_confus` to create a visualization containing a grid of confusion matrices corresponding to the validation predictions from the six original models.

```
# Extract the observed values
obs2016 <- eggdata16 %>% select(all_of(vars_resp))

# Join the plots
confus_matrix_grid <-
  plot_grid(
    plot_confus(obs = obs2016$Common_Name, pred = pred2016$Common_Name) +
      theme(legend.position = "none") +
      labs(title = "Species"),
```

```
    plot_confus(obs = obs2016$Genus, pred = pred2016$Genus) +
      theme(legend.position = "none") +
      labs(title = "Genus"),
    plot_confus(
      obs = obs2016$Common_Name_ACGC,
      pred = pred2016$Common_Name_ACGC
    ) +
      theme(legend.position = "none") +
      labs(title = "Species (IC grouped)"),
    plot_confus(obs = obs2016$Genus_ACGC, pred = pred2016$Genus_ACGC) +
      theme(legend.position = "none") +
      labs(title = "Genus (IC grouped)"),
    plot_confus(
      obs = obs2016$Common_Name_ACGC,
      pred = pred2016$Common_Name_ACGC_reduced
    ) +
      theme(legend.position = "none",
            plot.title = element_text(hjust = 0.75)) +
      labs(title = "Species (IC grouped, reduced variables)"),
    plot_confus(obs = obs2016$Family_ACGC, pred = pred2016$Family_ACGC) +
      theme(legend.position = "none") +
      labs(title = "Family (IC grouped)"),
    align = "v",
    ncol = 2,
    labels = c('(A)', '(B)', '(C)', '(D)', '(E)', '(F)'),
    label_fontfamily = ff,
    label_size = fs
  ) %>%
  plot_grid(
    get_legend(
      plot_confus(obs = obs2016$Genus, pred = pred2016$Genus) +
        theme(legend.position = "bottom") +
        guides(fill = guide_colourbar(barwidth = 15))
    ),
    rel_heights = c(0.95, 0.05),
    ncol = 1
  )

# Print the figure
confus_matrix_grid
```


(A)
Species
Observed genetic identification
Bighead Carp 0 0 0 0 0 7 0 0 0 4 0 0
Channel Shiner 0 0 0 0 0 0 0 0 0 0 0 0
Emerald Shiner 0 2 28 13 0 0 0 0 0 1 0 0
Freshwater Drum 0 0 6 301 0 0 1 0 1 0 0 0
Goldeye 0 0 0 0 1 0 0 0 0 0 0 0
Grass Carp 0 0 0 1 0 120 0 0 1 29 7 0
River Carpsucker 0 0 0 0 0 0 0 0 0 0 0 0
River Shiner 0 0 2 0 0 0 0 1 0 0 0 0
Shiner sp. 0 5 13 18 0 0 0 1 0 0 0 0
Silver Carp 0 0 0 0 0 98 0 0 0 28 1 0
Silver Chub 0 0 0 0 0 0 0 0 0 0 0 0
Speckled Chub 0 0 3 0 0 10 0 0 0 0 0 0
Bighead Carp
Channel Shiner
Emerald Shiner
Freshwater Drum
Goldeye
Grass Carp
River Carpsucker
River Shiner
Shiner sp.
Silver Carp
Silver Chub
Speckled Chub
Random forest prediction
(B)
Genus
Observed genetic identification
Aplodinotus 299 0 0 0 1 0 9
Ctenopharyngodon 1 113 0 36 0 7 1
Hiodon 0 0 1 0 0 0 0
Hypophthalmichthys 0 88 0 49 0 1 0
Ictiobus 0 0 0 0 0 0 0
Macrhybopsis 0 7 0 1 0 2 3
Notropis 23 0 0 1 0 1 59
Aplodinotus
Ctenopharyngodon
Hiodon
Hypophthalmichthys
Ictiobus
Macrhybopsis
Notropis
Random forest prediction
(C)
Species (IC grouped)
Observed genetic identification
Channel Shiner 0 0 0 0 0 0 0 0 0 0
Emerald Shiner 1 29 12 0 2 0 0 0 0 0
Freshwater Drum 0 6 300 0 1 1 0 1 0 0
Goldeye 0 0 0 1 0 0 0 0 0 0
Invasive Carp 0 1 1 0 269 0 0 1 24 0
River Carpsucker 0 0 0 0 0 0 0 0 0 0
River Shiner 0 1 0 0 1 0 1 0 0 0
Shiner sp. 6 13 16 0 1 0 1 0 0 0
Silver Chub 0 0 0 0 0 0 0 0 0 0
Speckled Chub 0 3 0 0 10 0 0 0 0 0
Channel Shiner
Emerald Shiner
Freshwater Drum
Goldeye
Invasive Carp
River Carpsucker
River Shiner
Shiner sp.
Silver Chub
Speckled Chub
Random forest prediction
(D)
Genus (IC grouped)
Observed genetic identification
Aplodinotus 298 0 1 1 0 9
Hiodon 0 1 0 0 0 0
Ictiobus 0 0 0 0 0 0
Invasive Carp 1 0 0 267 27 1
Macrhybopsis 0 0 0 9 0 4
Notropis 23 0 0 2 0 59
Aplodinotus
Hiodon
Ictiobus
Invasive Carp
Macrhybopsis
Notropis
Random forest prediction
(E)
Species (IC grouped, reduced variables)
Observed genetic identification
Channel Shiner 0 0 0 0 0 0 0 0 0 0
Emerald Shiner 1 29 13 0 0 1 0 0 0 0
Freshwater Drum 0 4 300 1 0 1 0 3 0 0
Gizzard Shad 0 0 0 0 0 0 0 0 0 0
Goldeye 0 0 0 0 1 0 0 0 0 0
Invasive Carp 1 1 1 0 0 278 0 0 15 0
River Shiner 0 1 0 0 0 1 1 0 0 0
Shiner sp. 5 12 15 0 0 0 1 4 0 0
Silver Chub 0 0 0 0 0 0 0 0 0 0
Speckled Chub 0 3 0 0 0 10 0 0 0 0
Channel Shiner
Emerald Shiner
Freshwater Drum
Gizzard Shad
Goldeye
Invasive Carp
River Shiner
Shiner sp.
Silver Chub
Speckled Chub
Random forest prediction
(F)
Family (IC grouped)
Observed genetic identification
Catostomidae 0 0 0 0 0
Cyprinidae 0 69 0 10 18
Hiodontidae 0 0 1 0 0
Invasive Carp 0 33 0 262 1
Sciaenidae 1 10 0 1 297
Catostomidae
Cyprinidae
Hiodontidae
Invasive Carp
Sciaenidae
Random forest prediction
Number of eggs
100
200
300

```
# Save the figure
ggsave(
  filename = "../figures/figure04.png",
  plot = confus_matrix_grid,
  height = 8.75,
  width = 6.5,
  dpi = fdpi
)
```

2.3 Figure 5: Metrics Separately by New and Old Locations

The code below prepares the validation metrics separated by old and new sites for plotting.

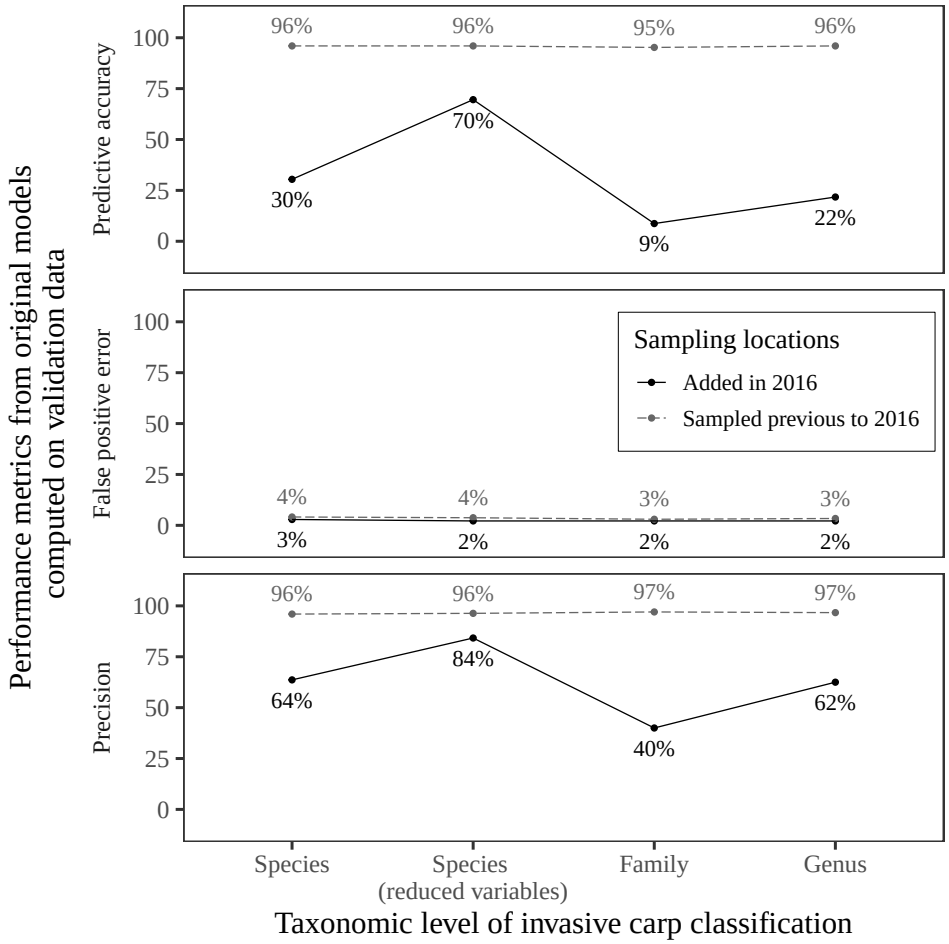
```
# Prepare the data for the plot
val_compare_plot_data <-
  val_metrics_separate %>%
  pivot_longer(names_to = "metric", values_to = "value", cols = pa:prec) %>%
  filter(model %in%
           c("Genus_ACGC", "Family_ACGC", "Common_Name_ACGC", "Common_Name_ACGC_reduced")) %>%
  mutate(
    model = fct_recode(
      model,
      "Family" = "Family_ACGC",
      "Genus" = "Genus_ACGC",
      "Species" = "Common_Name_ACGC",
      "Species \n(reduced variables)" = "Common_Name_ACGC_reduced"
    ),
    metric = fct_recode(
      metric,
      "Predictive accuracy" = "pa",
      "False positive error" = "fpr",
      "Precision" = "prec"
    ),
    site_type = fct_recode(
      site_type,
      "Added in 2016" = "new",
      "Sampled previous to 2016" = "old"
    )
  ) %>%
  mutate(
    metric = fct_relevel(
      metric,
      "Predictive accuracy",
      "False positive error",
      "Precision",
    ),
    model = fct_relevel(
      model,
      "Species",
      "Species \n(reduced variables)",
```

```
      "Family",
      "Genus"
    )
  )
```

A plot comparing the validation metrics between the old and new sites is created below.

```
# Create the plot
val_metric_compare_plot <-
  ggplot(val_compare_plot_data, aes(
         x = model,
         y = value * 100,
         group = site_type,
         linetype = site_type,
         color = site_type
       )) +
  geom_point(size = 1) +
  geom_line(size = 0.3) +
  geom_text(
    data = val_compare_plot_data  %>% filter(site_type == "Sampled previous to 2016"),
    aes(label = paste0(round(value * 100, 0), "%")),
    nudge_y = 10,
    show.legend = FALSE,
    family = ff,
    size = 4
  ) +
  geom_text(
    data = val_compare_plot_data %>% filter(site_type == "Added in 2016"),
    aes(label = paste0(round(value * 100, 0), "%")),
    nudge_y = -10,
    show.legend = FALSE,
    family = ff,
    size = 4
  ) +
  facet_grid(
    metric ~ .,
    switch = "y",
    scales = "free",
    space = "free_x"
  ) +
  theme_bw(base_size = fs*1.5, base_family = ff) +
  theme(
    strip.placement = "outside",
    strip.background = element_rect(color = "white", fill = "white"),
    legend.position = c(.78, 0.55),
    legend.background = element_rect(linetype = "solid", color = "black", size = 0.2),
    legend.text = element_text(size = (fs*1.5)-4),
    legend.title = element_text(size = (fs*1.5)-2),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  ) +
```

```
  labs(
    x = "Taxonomic level of invasive carp classification",
    y = "Performance metrics from original models \ncomputed on validation data",
    color = "Sampling locations",
    linetype = "Sampling locations"
  ) +
  scale_color_manual(values = c("black", "grey40")) +
  scale_linetype_manual(values = c("solid", "longdash")) +
  scale_y_continuous(breaks = seq(0, 100, 25), limits = c(-10, 110)) +
  guides(linetype = guide_legend(ncol = 2))

# Print the figure
val_metric_compare_plot
```




```
# Save the figure
ggsave(
  filename = "../figures/figure05.png",
  plot = val_metric_compare_plot,
  height = 5.5,
  width = 6.5,
  dpi = fdpi
)
```

3 Augmented Models

This section contains the code corresponding to the visualizations in the paper for the augmented models.

3.1 Figure 6: Performance Metrics

The code below prepares the data for the visualization to compare the 2014-2015 and 2014-2016 model performance metrics.

```
# 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)
  )

# Prepare the model performance metrics for plotting
perf_plot_data <-
  metrics_joined %>%
  pivot_longer(names_to = "metric",
               values_to = "value",
               cols = -c(model, class)) %>%
  filter(str_detect(metric, ".paper", negate = TRUE)) %>%
  separate(metric, c("metric", "years"), "_") %>%
  mutate(model_class = factor(class):factor(model)) %>%
  mutate(
    model_class = factor(class):factor(model),
    model_type = ifelse(
      str_detect(model_class, "ACGC"),
      "Grouped",
      "Separate"
    )
  ) %>%
  mutate(
    taxa = fct_recode(
      model,
      "Genus" = "Genus",
      "Species" = "Common_Name",
      "Family" = "Family_ACGC",
```

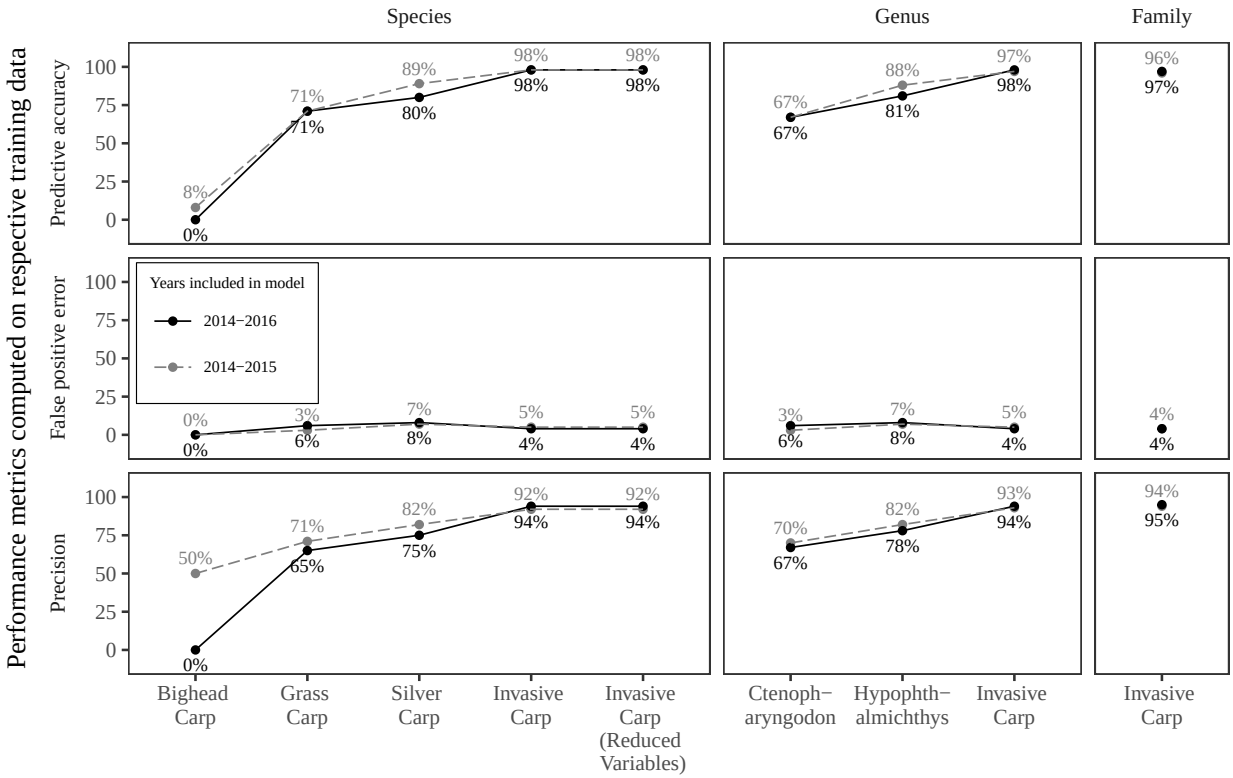
```
    "Genus" = "Genus_ACGC",
    "Species" = "Common_Name_ACGC",
    "Species" = "Common_Name_ACGC_reduced"
  ),
  model_class = fct_recode(
    model_class,
    "Bighead \nCarp" = "Bighead Carp:Common_Name",
    "Grass \nCarp" = "Grass Carp:Common_Name",
    "Silver \nCarp" = "Silver Carp:Common_Name",
    "Ctenoph-\naryngodon" = "Ctenopharyngodon:Genus",
    "Hypophth-\nalmichthys" = "Hypophthalmichthys:Genus",
    "Invasive \nCarp" = "ACGC:Common_Name_ACGC",
    "Invasive \nCarp \n(Reduced \nVariables)" = "ACGC:Common_Name_ACGC_reduced",
    "Invasive \nCarp" = "ACGC:Genus_ACGC",
    "Invasive \nCarp" = "ACGC:Family_ACGC"
  ),
  metric = fct_recode(
    metric,
    "Predictive accuracy" = "pa",
    "False positive error" = "fpr",
    "Precision" = "prec"
  ),
  years = fct_recode(
    years,
    "2014-2015" = "1415",
    "2014-2016" = "141516"
  )
) %>%
mutate(
  metric = fct_relevel(
    metric,
    "Predictive accuracy",
    "False positive error",
    "Precision",
  ),
  taxa = fct_relevel(taxa, "Species", "Genus", "Family"),
  model_class = fct_relevel(
    model_class,
    "Bighead \nCarp",
    "Grass \nCarp",
    "Silver \nCarp",
    "Ctenoph-\naryngodon",
    "Hypophth-\nalmichthys",
    "Invasive \nCarp",
    "Invasive \nCarp \n(Reduced \nVariables)"
  ),
  model_type =
    fct_relevel(
      model_type,
      "Separate",
      "Grouped"
      ),
  years = fct_relevel(
    years,
```

```
      "2014-2016",
      "2014-2015"
    )
  )
```

The visualization comparing the 2014-2015 and 2014-2016 model performance metrics is included below.

```
perf_metric_plot <-
  ggplot(perf_plot_data, aes(
    x = model_class,
    y = value,
    group = years,
    color = years,
    linetype = years
  )) +
  geom_point(size = 1) +
  geom_line(size = 0.3) +
  geom_text(
    data = perf_plot_data %>% filter(years == "2014-2015"),
    aes(label = paste0(round(value, 0), "%")),
    nudge_y = 10,
    show.legend = FALSE,
    family = ff,
    size = 2.5
  ) +
  geom_text(
    data = perf_plot_data %>% filter(years == "2014-2016"),
    aes(label = paste0(round(value, 0), "%")),
    nudge_y = -10,
    show.legend = FALSE,
    family = ff,
    size = 2.5
  ) +
  facet_grid(metric ~ taxa,
    switch = "y",
    scales = "free",
    space = "free_x"
  ) +
  theme_bw(base_size = fs, base_family = ff) +
  theme(
    strip.placement = "outside",
    strip.background = element_rect(color = "white", fill = "white"),
    legend.position = c(0.09, 0.54),
    legend.background = element_rect(linetype = "solid", color = "black", size = 0.2),
    legend.text = element_text(size = fs-4),
    legend.title = element_text(size = fs-4),
    legend.box = "horizontal",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  ) +
  scale_color_manual(values = c("black", "grey50")) +
```

```
  scale_linetype_manual(values = c("solid", "longdash")) +
  labs(
    x = "Model response variable (taxonomic level; top labels) and \nclassification category (bottom lab
    y = "Performance metrics computed on respective training data",
    color = "Years included in model",
    linetype = "Years included in model"
  ) +
  scale_y_continuous(breaks = seq(0, 100, 25), limits = c(-10, 110))

# Print the figure
perf_metric_plot
```




```
# Save the figure
ggsave(
  filename = "../figures/figure06.png",
  plot = perf_metric_plot,
  height = 4.5,
  width = 6.5,
  dpi = fdpi
)
```

3.2 Figure 7: Variable Importance

The code below computes the variable importance for the original and augmented models.

```
# Compute variable importance for the original models
vi1415 <- map_df(
  .x = rfs1415,
  .f = function(model) {
    data.frame(model$importance) %>%
      select(MeanDecreaseGini) %>%
      mutate(variable = rownames(model$importance))
  },
  .id = "model"
) %>%
  group_by(model) %>%
  mutate(rank = factor(rank(-MeanDecreaseGini), levels = 17:1)) %>%
  arrange(model, rank)

# Compute variable importance for the augmented models
vi141516 <- map_df(
  .x = rfs141516,
  .f = function(model) {
    data.frame(model$importance) %>%
      select(MeanDecreaseGini) %>%
      mutate(variable = rownames(model$importance))
  },
  .id = "model"
) %>%
  group_by(model) %>%
  mutate(rank = factor(rank(-MeanDecreaseGini), levels = 17:1)) %>%
  arrange(model, rank)
```

The variable importance values are joined in a dataframe below in preparation for plotting.

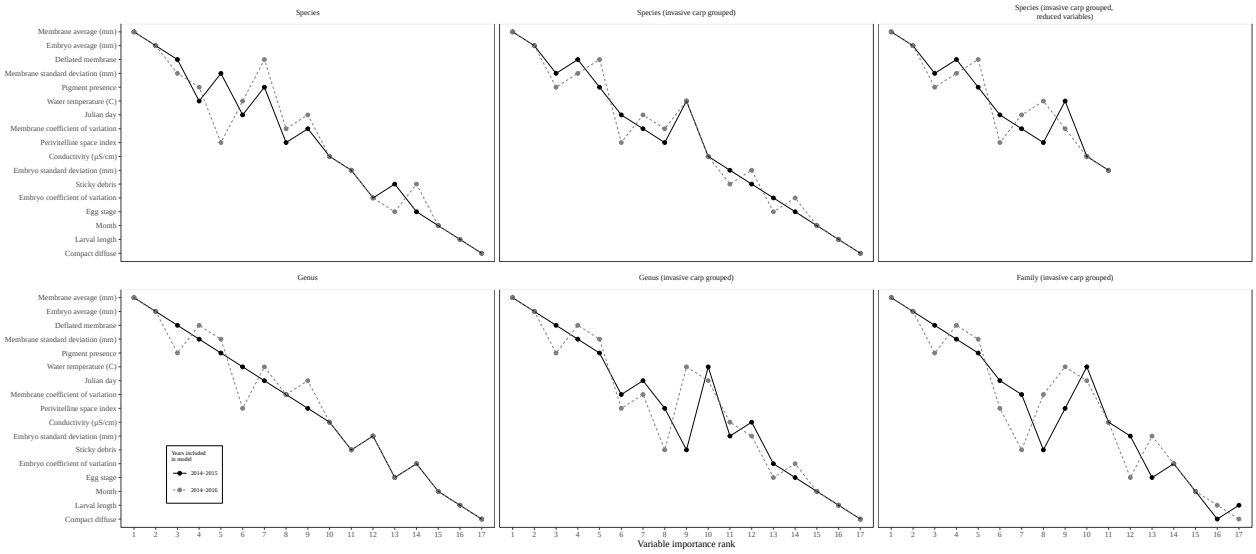
```
vi_plot_data <-
  vi1415 %>%
  mutate(year = "2014-2015") %>%
  bind_rows(vi141516 %>% mutate(year = "2014-2016")) %>%
  rename(vi = MeanDecreaseGini) %>%
  select(model, year, variable, vi, rank) %>%
  mutate_at(
    .vars = vars(model, variable),
    .funs = function(.)
      str_replace_all(., "_", " ") %>% str_replace("ACGC", "(invasive carp grouped)")
  ) %>%
  mutate(model = factor(model)) %>%
  mutate(
    model = fct_relevel(
      model,
      "Common Name",
      "Common Name (invasive carp grouped)",
      "Common Name (invasive carp grouped) reduced",
      "Genus",
      "Genus (invasive carp grouped)",
      "Family (invasive carp grouped)"
```

```
    )
  ) %>%
  mutate(
    model = fct_recode(
      model,
      "Species" = "Common Name",
      "Species (invasive carp grouped)" = "Common Name (invasive carp grouped)",
      "Species (invasive carp grouped, \nreduced variables)" =
        "Common Name (invasive carp grouped) reduced",
      "Genus (invasive carp grouped)" = "Genus (invasive carp grouped)",
      "Family (invasive carp grouped)" = "Family (invasive carp grouped)"
    )
  ) %>%
  mutate(
    variable = fct_recode(
      factor(variable),
      "Pigment presence" = "Pigment",
      "Deflated membrane" = "Deflated",
      "Embryo average (mm)" = "Yolk Ave",
      "Embryo standard deviation (mm)" = "Yolk SD",
      "Embryo coefficient of variation" = "Yolk CV",
      "Membrane average (mm)" = "Membrane Ave",
      "Membrane standard deviation (mm)" = "Membrane SD",
      "Membrane coefficient of variation" = "Membrane CV",
      "Perivitelline space index" = "Yolk to Membrane Ratio",
      "Water temperature (C)" = "Temperature",
      "Conductivity (\U00B5S/cm)" = "Conductivity",
      "Sticky debris" = "Sticky Debris",
      "Julian day" = "Julian Day",
      "Egg stage" = "Egg Stage",
      "Larval length" = "Larval Length",
      "Compact diffuse" = "Compact Diffuse"
    )
  )

vi_ave_order <-
  vi_plot_data %>%
  filter(year == "2014-2015") %>%
  group_by(variable) %>%
  summarise(mean_vi = mean(vi), .groups = "drop") %>%
  arrange(mean_vi) %>%
  pull(variable)
```

A visualization comparing the variable importance values is created in the following code.

```
vi_plot <-
  vi_plot_data %>%
  ggplot(aes(
    y = factor(variable, levels = vi_ave_order),
    x = factor(rank, levels = 1:17),
    color = year,
```

```
    group = year,
    linetype = year
  )) +
  geom_point(size = 1.5) +
  geom_line(size = 0.3) +
  facet_wrap(. ~ model, ncol = 3) +
  scale_color_manual(values = c("black", "grey50")) +
  scale_linetype_manual(values = c("solid", "dashed")) +
  theme_bw(base_size = fs, base_family = ff) +
  theme(
    strip.background = element_rect(color = "white", fill = "white"),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    legend.position = c(0.065, 0.105),
    legend.background = element_rect(linetype = "solid", color = "black", size = 0.2),
    legend.text = element_text(size = fs-4),
    legend.title = element_text(size = fs-4),
    axis.title.y = element_blank()
  ) +
  labs(
    x = "Variable importance rank",
    color = "Years included \nin model",
    linetype = "Years included \nin model"
  )

# Print the figure
vi_plot
```




```
# Save the figure
ggsave(
  filename = "../figures/figure07.png",
  plot = vi_plot,
  height = 5,
  width = 8.5,
  dpi = fdpi
)
```

3.3 Figure 8: Partial Dependence Plots

The code below computes the values for the partial dependence plots for the original and augmented models with reduced variables for all predictor variables. The values are put in dataframes and saved. If dataframes have been run previously, then they are loaded.

```
# Compute and save (or load) individual conditional expectation (ICE)
# curves 2014-2015 models
ice1415_file_name = "../results/ice1415.csv"
if (file.exists(ice1415_file_name)) {
  ice1415 <- read.csv(ice1415_file_name)
} else {
  ice1415 <-
    map_dfr(
    .x = vars_pred_reduced,
    .f = function(var) {
      pdp::partial(
        object = rfs1415$Common_Name_ACGC_reduced,
        pred.var = var,
        train = eggdata1415,
        which.class = 1,
        type = "classification",
        prob = TRUE,
        ice = TRUE
      ) %>%
        data.frame() %>%
        mutate(variable = all_of(var)) %>%
        rename(value = all_of(var)) %>%
        mutate(value = as.character(value))
    }
  )
  write.csv(ice1415, ice1415_file_name, col.names = FALSE)
}

# Compute and save (or load) individual conditional expectation (ICE)
# curves 2014-2016 models
ice141516_file_name = "../results/ice141516.csv"
if (file.exists(ice141516_file_name)) {
  ice141516 <- read.csv(ice141516_file_name)
} else {
  ice141516 <-
    map_dfr(
    .x = vars_pred_reduced,
    .f = function(var) {
      pdp::partial(
        object = rfs141516$Common_Name_ACGC_reduced,
        pred.var = all_of(var),
        train = eggdata141516,
        which.class = 1,
        type = "classification",
        prob = TRUE,
        ice = TRUE
      ) %>%
```

```
        data.frame() %>%
        mutate(variable = all_of(var)) %>%
        rename(value = all_of(var)) %>%
        mutate(value = as.character(value))
    }
  )
  write.csv(ice141516, ice141516_file_name, col.names = FALSE)
}
```

The partial dependence curves (and their standard deviations) for the original and augmented model are computed below based on the ICE curves.

```
# Compute partial dependence curves for the original model
pd1415 <-
  ice1415 %>%
  group_by(variable, value) %>%
  summarise(
    mean = mean(yhat),
    sd = sd(yhat),
    min = min(yhat),
    max = max(yhat),
    .groups = "drop"
  )

# Compute partial dependence curves for the augmented model
pd141516 <-
  ice141516 %>%
  group_by(variable, value) %>%
  summarise(
    mean = mean(yhat),
    sd = sd(yhat),
    min = min(yhat),
    max = max(yhat),
    .groups = "drop"
  )
```

The function below (`create_pd_plot`) creates a partial dependence plot for the augmented model given the name of a predictor variable and whether or not the partial dependence curve from the original models should be included.

```
# Function for creating a partial dependence plot given a variable
create_pd_plot <- function(var, include_1415 = FALSE) {

  # Improve variable name for plot
  var_clean <- data.frame(
    name = vars_pred_reduced,
    name_clean = c(
      "Membrane average (mm)",
```

```
    "Embryo average (mm)",
    "Deflated membrane",
    "Membrane standard deviation (mm)",
    "Water temperature (C)",
    "Pigment presence",
    "Julian day",
    "Perivitelline space index",
    "Membrane coefficient of variation",
    "Conductivity (\U00B5S/cm)",
    "Embryo standard deviation (mm)"
  )
) %>%
  filter(name == var) %>%
  pull(name_clean)

# Extract plot data
pd_plot_data <-
  pd141516 %>%
  filter(variable == var) %>%
  mutate(variable = var_clean)

# Extract plot data for original models (if requested)
if (include_1415 == TRUE) {
  pd1415_plot_data <-
    pd1415 %>%
    filter(variable == var) %>%
    mutate(variable = var_clean)
}

# Determine if var is categorical
if (var %in% c("Deflated", "Pigment")) cat = TRUE else cat = FALSE

if (cat == FALSE) {
  pd_plot_data <- pd_plot_data %>% mutate(value = as.numeric(as.character(value)))
  if (include_1415 == TRUE) {
    pd1415_plot_data <-
      pd1415_plot_data %>% mutate(value = as.numeric(as.character(value)))
  }
}

# Change variable to continuous and extract rug data if continuous
if (cat == FALSE) {
  pd_plot_data <-
    pd_plot_data %>% mutate(value = as.numeric(as.character(value)))
  rug_plot_data <-
    eggdata141516 %>%
    select(all_of(var)) %>%
    pivot_longer(names_to = "variable", cols = everything()) %>%
    mutate(variable = var_clean)
}

# Start plot (based on categorical or not)
pd_plot <- ggplot(pd_plot_data)
```

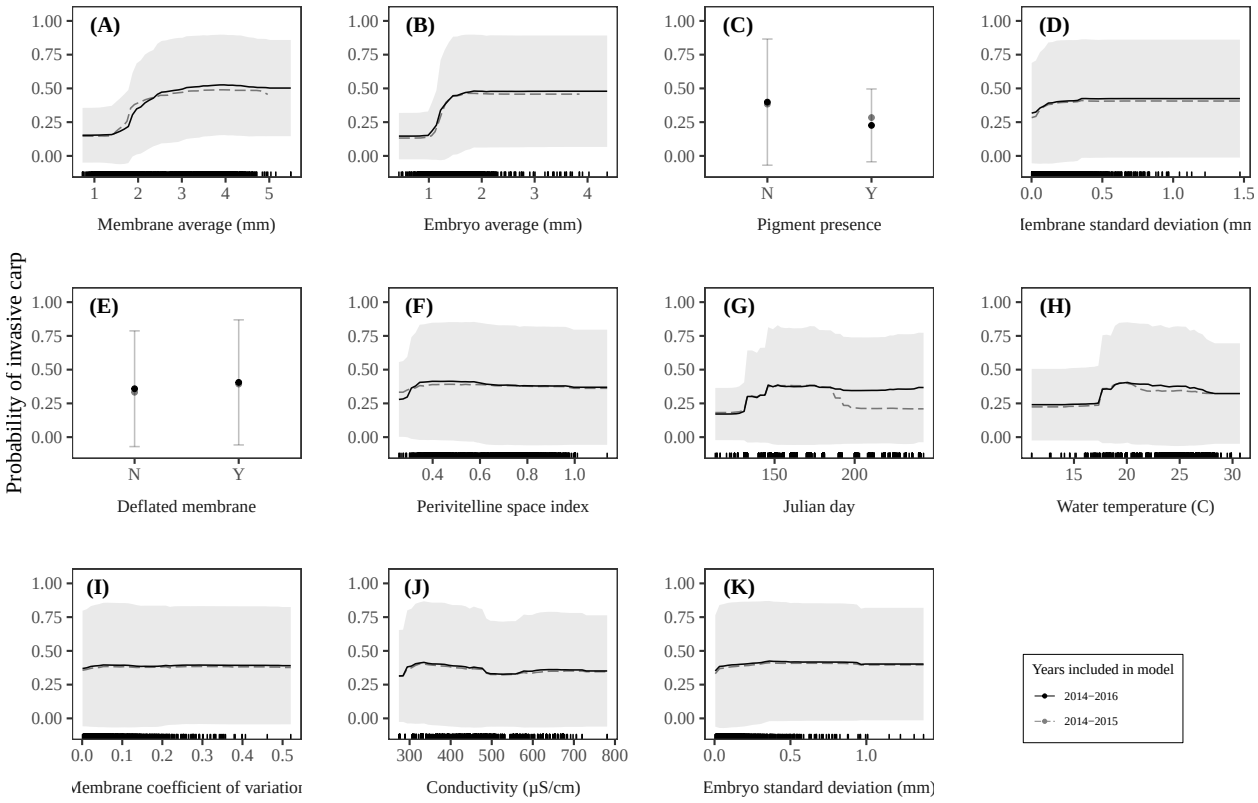
```
# Add line/point for 1415 model (if requested)
if (include_1415 == TRUE) {
  if (cat == TRUE) {
    pd_plot <- pd_plot +
    geom_point(data = pd1415_plot_data,
               aes(
                 x = value,
                 y = mean,
                 group = variable,
                 color = "Original model"
               )) +
    scale_color_manual(values = "grey50")
  } else {
   pd_plot <- pd_plot +
    geom_line(data = pd1415_plot_data,
               aes(
                 x = value,
                 y = mean,
                 group = variable,
                 linetype = "Original model"
               ), color = "grey50") +
    scale_linetype_manual(values = "longdash")
  }
}

# Add lines/points for pdp of 2014-2016 models
if (cat == TRUE) {
  pd_plot <-
    pd_plot +
    geom_point(aes(x = value, y = mean, group = variable)) +
    geom_errorbar(aes(
      x = value,
      ymin = mean - sd,
      ymax = mean + sd
    ),
    width = 0.1,
    alpha = 0.25)
} else {
  pd_plot <-
    pd_plot +
    geom_line(aes(x = value, y = mean, group = variable)) +
    geom_ribbon(aes(
      x = value,
      ymin = mean - sd,
      ymax = mean + sd
    ), alpha = 0.1) +
    geom_rug(data = rug_plot_data, aes(x = value))
}

# Add facets and theme
pd_plot <- pd_plot +
  facet_wrap(. ~ variable, scales = "free_x", strip.position = "bottom") +
  ylim(-0.1, 1.05) +
  theme_bw(base_size = fs*1.5, base_family = ff) +
```

```
    theme(
      strip.background = element_rect(color = "white", fill = "white"),
      strip.placement = "outside",
      legend.background = element_rect(
        linetype = "solid",
        color = "black",
        size = 0.2
      ),
      legend.text = element_text(size = (fs*1.5)-2),
      legend.title = element_text(size = fs*1.5),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank()
    )

  # Finish plot by adding labels
  if (var == "Deflated") {
    pd_plot + labs(x = "", y = "Probability of invasive carp") +
      theme(axis.title = element_text(size = fs*1.5))
  } else {
    pd_plot + labs(x = "", y = "")
  }

}
```

A grid of partial dependence plots for the augmented model with partial dependence curves from the original model is created below.

```
# Determine order of reduced variables based on VI from augmented model
vars_pred_reduced_vi <-
  vi141516 %>%
  filter(model == "Common_Name_ACGC_reduced") %>%
  arrange(desc(MeanDecreaseGini)) %>%
  pull(variable)

# Create the pdp plots
pd_plots <-
  map(
    .x = vars_pred_reduced_vi,
    .f = function(var) create_pd_plot(var, include_1415 = TRUE) + theme(legend.position = "none")
  )

# Use the legend from the performance metric plot for the pd plots
pd_plot_legend <-
  get_legend(
    perf_metric_plot +
      theme_bw(base_size = 12, base_family = ff) + theme(
        legend.position = "right",
        legend.background = element_rect(
          linetype = "solid",
          color = "black",
          size = 0.2
```

```
        ),
        legend.text = element_text(size = 8),
        legend.title = element_text(size = 10)
      )
  ) %>% list()

# Create the panel of pd plots
pd_plot <-
  plot_grid(
    plotlist = c(pd_plots, pd_plot_legend),
    labels = c('(A)', '(B)', '(C)', '(D)', '(E)', '(F)', '(G)', '(H)', '(I)', '(J)', '(K)'),
    label_fontfamily = ff,
    label_size = fs*1.5, label_x = 0.25, label_y = 0.95
  )

# Print the figure
pd_plot
```




```
# Save the figure
ggsave(
  filename = "../figures/figure08.png",
  plot = pd_plot,
  height = 8,
  width = 13
)
```

The following code computes the change in average probability of classification of an invasive carp based on

a specified change in average membrane diameter and average yolk diameter.

```
# Change in membrane average
pd141516 %>%
  filter(variable == "Membrane_Ave") %>%
  mutate(value = as.numeric(as.character(value))) %>%
  filter(value >= 1.5, value <= 2.5) %>%
  summarise(diff = max(mean) - min(mean)) %>%
  pull(diff)
```

```
## [1] 0.2740814
```

```
# Change in yolk average
pd141516 %>%
  filter(variable == "Yolk_Ave") %>%
  mutate(value = as.numeric(as.character(value))) %>%
  filter(value >= 1, value <= 2) %>%
  summarise(diff = max(mean) - min(mean)) %>%
  pull(diff)
```

```
## [1] 0.2935774
```

3.4 Figure 9: Correlation Heatmap

Below, the correlations between the continuous predictor variables in the reduced set are computed.

```
# Compute the correlation between continuous reduced variables
cor_red_vars <-
  eggdata141516 %>%
  select(all_of(vars_pred_reduced), -Pigment, -Deflated) %>%
  cor() %>%
  data.frame() %>%
  mutate(vars_rows = rownames(.)) %>%
  pivot_longer(names_to = "vars_cols", values_to = "cor", cols = -vars_rows) %>%
  mutate(cor = round(cor, 2))
```

The ten pairs of variables with the largest magnitudes of correlation are printed below.

```
# Identify variables with the highest correlations
cor_red_vars %>%
  filter(vars_rows != vars_cols) %>%
  arrange(desc(abs(cor))) %>%
  slice(1:10)
```

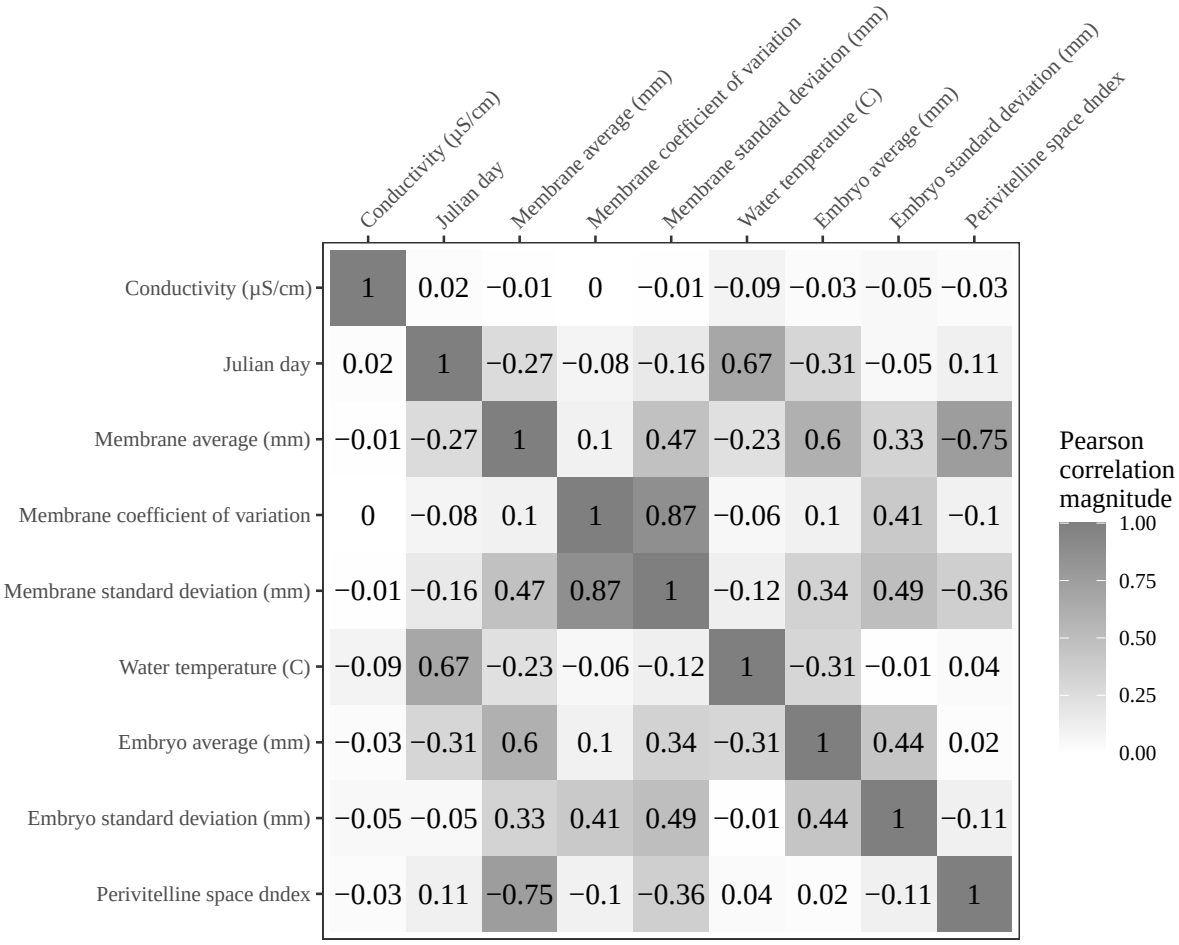
```
## # A tibble: 10 x 3
##    vars_rows              vars_cols               cor
```

```
##    <chr>                  <chr>                  <dbl>
##  1 Membrane_SD            Membrane_CV             0.87
##  2 Membrane_CV            Membrane_SD             0.87
##  3 Membrane_Ave           Yolk_to_Membrane_Ratio -0.75
##  4 Yolk_to_Membrane_Ratio Membrane_Ave           -0.75
##  5 Temperature            Julian_Day              0.67
##  6 Julian_Day             Temperature             0.67
##  7 Membrane_Ave           Yolk_Ave                0.6
##  8 Yolk_Ave               Membrane_Ave            0.6
##  9 Membrane_SD            Yolk_SD                 0.49
## 10 Yolk_SD                Membrane_SD             0.49
```

A heatmap of the correlations is printed below.

```
# Plot the correlations
cor_heatmap <-
  cor_red_vars %>%
  mutate(vars_rows = factor(vars_rows, levels = rev(levels(
    factor(cor_red_vars$vars_rows)
  )))) %>%
  mutate_at(
    .vars = c("vars_rows", "vars_cols"),
    .funs = function(.) {
      fct_recode(
        .,
        "Membrane average (mm)" = "Membrane_Ave",
        "Embryo average (mm)" = "Yolk_Ave",
        "Membrane standard deviation (mm)" = "Membrane_SD",
        "Water temperature (C)" = "Temperature",
        "Julian day" = "Julian_Day",
        "Perivitelline space dndex" = "Yolk_to_Membrane_Ratio",
        "Membrane coefficient of variation" = "Membrane_CV",
        "Conductivity (\U00B5S/cm)" = "Conductivity",
        "Embryo standard deviation (mm)" = "Yolk_SD"
      )
    }
  ) %>%
  ggplot(aes(x = vars_cols, y = vars_rows, fill = abs(cor))) +
  geom_tile() +
  geom_text(aes(label = cor), family = ff) +
  scale_fill_gradient(low = "white", high = "grey50", limits = c(0, 1)
  ) +
  theme_bw(base_size = fs, base_family = ff) +
  scale_x_discrete(position = "top") +
  theme(
    panel.grid = element_blank(),
    aspect.ratio = 1,
    axis.text.x = element_text(angle = 45, hjust = 0)
  ) +
  labs(x = "", y = "", fill = "Pearson \ncorrelation \nmagnitude")
```

```
# Print the figure
cor_heatmap
```




```
# Save the figure
ggsave(
  filename = "../figures/figure09.png",
  plot = cor_heatmap,
  height = 6.5,
  width = 6.5,
  dpi = fdpi
)
```

4 Session Info

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

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## 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] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] tidyr_1.1.2         stringr_1.4.0       randomForest_4.6-14
## [4] purrr_0.3.4         pdp_0.7.0           ggplot2_3.3.3.9000
## [7] forcats_0.5.0       dplyr_1.0.3         cowplot_1.1.0
##
## loaded via a namespace (and not attached):
##  [1] pillar_1.4.7     compiler_4.0.3   tools_4.0.3      digest_0.6.27
##  [5] evaluate_0.14    lifecycle_0.2.0  tibble_3.0.5     gtable_0.3.0
##  [9] lattice_0.20-41  pkgconfig_2.0.3  rlang_0.4.10     cli_2.2.0
## [13] DBI_1.1.1        yaml_2.2.1       xfun_0.20        gridExtra_2.3
## [17] withr_2.3.0      knitr_1.30       generics_0.1.0   vctrs_0.3.6
## [21] grid_4.0.3       tidyselect_1.1.0 glue_1.4.2       R6_2.5.0
## [25] fansi_0.4.2      rmarkdown_2.6    farver_2.0.3     magrittr_2.0.1
## [29] scales_1.1.1     ellipsis_0.3.1   htmltools_0.5.1  assertthat_0.2.1
## [33] colorspace_2.0-0 labeling_0.4.2   utf8_1.1.4       stringi_1.5.3
## [37] munsell_0.5.0    crayon_1.3.4
```