From Data to Insights: Analyzing Stream Fish Functional Traits With Neural Networks

Fabricio dos Anjos Santa Rosa

OVERVIEW

This code process generates neural networks based on 29 functional traits of stream fishes. Functional traits offer insights into the ecological roles and adaptations of organisms, facilitating the understanding of community dynamics and ecosystem functioning. The code computes a distance matrix based on the Euclidean approach and conducts a non-metric multidimensional scaling (nMDS) ordination analysis. Visualizations of the nMDS stress metric and coordinates are produced, alongside data wrangling for plot generation. The subsequent Python section involves importing packages, loading the dataset, preprocessing, normalizing data, and partitioning it for training and testing. Finally, a neural network is assembled, trained, and evaluated, with metrics visualized using ggplot in R.

• Install and/or Load packages:

```
pack <- c('vegan', 'dplyr','ggplot2','plyr', 'tidyr', 'tibble','reticulate')
vars <- pack[!(pack %in% installed.packages()[, "Package"])]
if (length(vars != 0)) {
  install.packages(vars, dependencies = TRUE)
}
sapply(pack, require, character.only = TRUE)
```

• Set the directory:

```
knitr::opts_knit$set(root.dir = "C:/Users/Administrador/Downloads")
```

• Load the dataset in R environment:

```
Species <- read.csv("Species.csv")
```

• Create a distance matrix based on euclidean approach:

```
distances <- dist(Species[,2:ncol(Species)], method = "euclidean")
```

• Run an ordination analysis - nMDS:

```
nmds <- metaMDS(distances, distance = "euclidean")
```

• See the nMDS stress metric:

```
nmds$stress
```

```
## [1] 0.0009538131
```

• Extract NMDS coordinates:

```
nmds_coords <- as.data.frame(scores(nmds))
```

• Add a column with the grouping variable (species in this case):

```
nmds_coords <- data.frame(Specie = Species$Specie, nmds_coords)
```

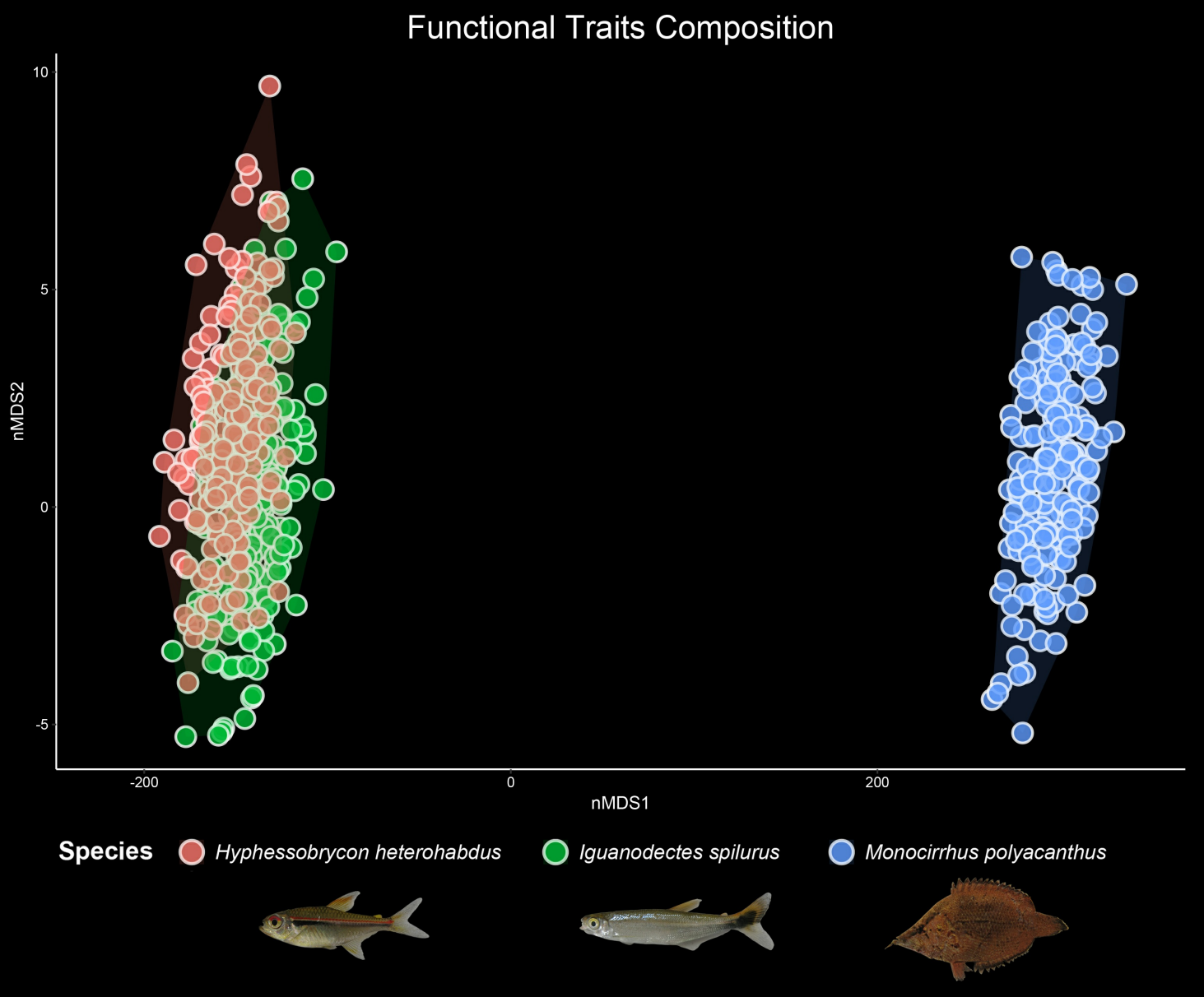
• Wrangling the data to get the points outline:

```r
composition <- nmds_coords %>%
  group_by(Specie) %>%
  mutate_at(vars(NMDS1, NMDS2), list(~ . + jitter(1.00)))

find_hull <- function(composition) composition[chull(composition$NMDS1, composition$NMDS2), ]
hulls <- ddply(composition, "Specie", find_hull)
```

• **Plot in R + Adjusts in Photoshop:**

```r
ggplot(composition, aes(x = NMDS1, y = NMDS2, fill = Specie)) +
  xlim(-220,340)+
  geom_point(shape = 21, color = "White", size = 5, stroke = 1.2, alpha = 0.7) +
  geom_polygon(data = hulls, aes(color = Specie, fill = Specie), alpha = 0.15, size = 3, color = "NA") +
  labs(title = "Functional Traits Composition", x = "nMDS1", y = "nMDS2", fill = "Species") +
  scale_fill_discrete(labels = function(x) gsub("_", " ", x)) + # Modify legend labels
  theme(
    plot.background = element_rect(fill = "black"),
    panel.background = element_rect(fill = "black"),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(color = "white"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", hjust = 0.5, size = 20),
    legend.background = element_rect(fill = "black", color = NA),
    legend.text = element_text(color = "white", face = "italic", size = 11),
    legend.title = element_text(color = "white", face = "bold") #13x7
  )
```



• **Define a vector of Python packages to install:**

```
packages <- c("tensorflow", "pandas", "numpy", "scikit-learn", "matplotlib", "seaborn")

# Install packages using a loop
for (package in packages) {
  py_install(package)
}
```

• Define a Python path:

```
use_python("C:/ProgramData/anaconda3/")
```

• Define a vector of Python packages to load:

```
# Define a vector of Python package names
package_names <- c("tensorflow", "pandas", "numpy", "sklearn", "matplotlib.pyplot", "seaborn")

imported_packages <- list()

for (package_name in package_names) {
  # Import the package and store it in the list
  imported_packages[[package_name]] <- import(package_name)
}
```

• Load the dataset in Phyton environment:

```
py_run_string("
import pandas as pd

dataset = pd.read_csv('Species.csv')
")
```

• Let's see the head of the six first columns of our fish dataset:

```
head(py$dataset[,1:5])
```

```
##                Specie   L_boca  L_corpo L_pedunculo L_cabeca
## 1 Iguanodectes_spilurus 2.118545 3.193178   0.4928050 3.588708
## 2 Iguanodectes_spilurus 2.097996 2.908360   0.4997328 3.643324
## 3 Iguanodectes_spilurus 1.947814 3.241776   0.4901531 3.554564
## 4 Iguanodectes_spilurus 2.090229 3.695870   0.5257104 3.767944
## 5 Iguanodectes_spilurus 1.905121 2.969604   0.4962976 3.660668
## 6 Iguanodectes_spilurus 1.971892 3.251269   0.5033700 3.834139
```

• Here we select only the functional characters to insert in the neural network:

```
py_run_string("
X = dataset.iloc[:, 1:29].values
")
```

• Here we wrangling and select only the labels (species names) to insert in the neural network:

```
py_run_string("
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Assuming y contains the names of the three classes
y_n = dataset.iloc[:, 0].values

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Fit label encoder and transform the target labels
y = label_encoder.fit_transform(y_n)
")
```

• Data normalization (z-scores) of functional characters:

```
py_run_string("
scaler = StandardScaler()
X = scaler.fit_transform(X)
")
```

• Separate the dataset in train and test:

```
py_run_string("
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
")
```

• **Neural network assembly and training:**

```
py_run_string("import tensorflow as tf
import numpy as np
model = tf.keras.models.Sequential([
  tf.keras.layers.Dense(units=15, activation='relu', input_shape=(28,)),
  tf.keras.layers.Dense(units=15, activation='relu'),
  tf.keras.layers.Dense(units=15, activation='relu'),
  tf.keras.layers.Dense(units=np.max(y)+1, activation='softmax')
])")

py_run_string("model.compile(optimizer='Adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])")

py_run_string("history = model.fit(X_train, y_train, epochs=50, validation_split=0.1)")
```

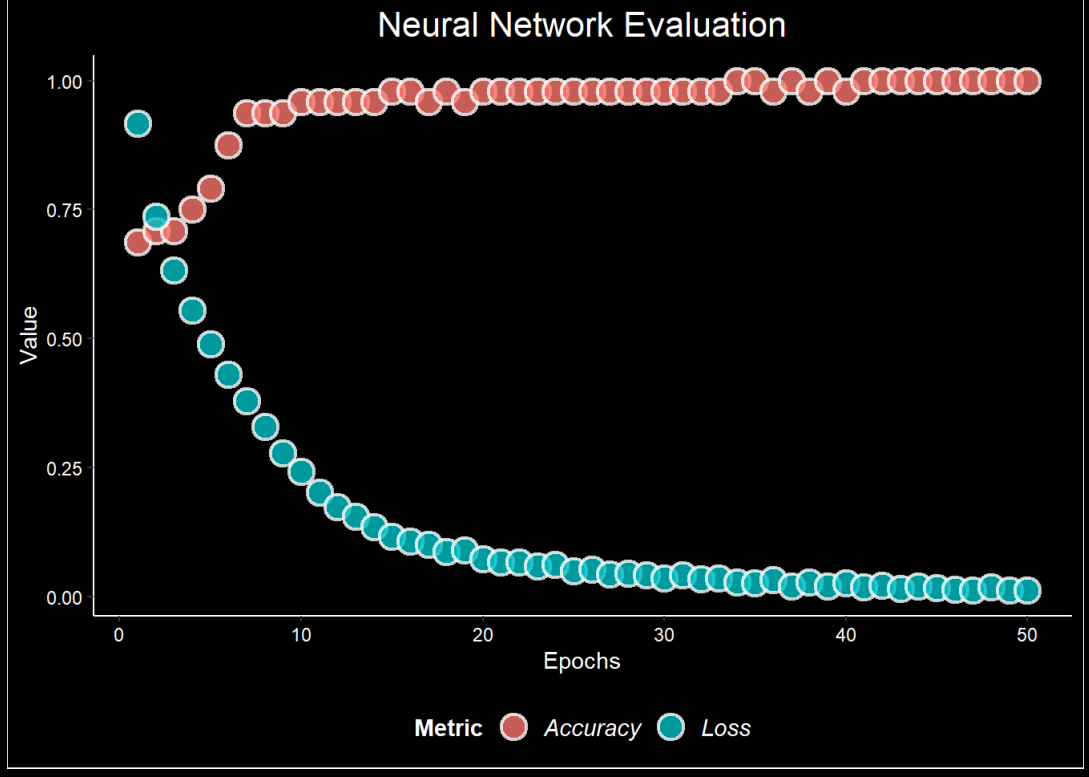
• **Neural network evaluation:**

```
Loss <- py$history$history$val_loss
Accuracy <- py$history$history$val_accuracy
Epochs <- 1:50

Neural_Network_Metrics <- data.frame(Epochs,Accuracy,Loss)

Neural_Network_Metrics_plot <- gather(Neural_Network_Metrics, key = "Metric", value = "Value", -Epochs)

# Create the ggplot graph
ggplot(data = Neural_Network_Metrics_plot, aes(x = Epochs, y = Value, fill = Metric)) +
  geom_point(shape = 21, color = "White", size = 5, stroke = 1.1, alpha = 0.8) +
  labs(title = "Neural Network Evaluation") +
  theme(
    plot.background = element_rect(fill = "black"),
    panel.background = element_rect(fill = "black"),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(color = "white"),
    axis.text = element_text(color = "white"),
    axis.title = element_text(color = "white"),
    plot.title = element_text(color = "white", hjust = 0.5, size = 16),
    legend.position = "bottom",
    legend.background = element_rect(fill = "black", color = NA),
    legend.key = element_rect(fill = "black", color = "black"),
    legend.text = element_text(color = "white", face = "italic", size = 11),
    legend.title = element_text(color = "white", face = "bold")
  )
```



INSIGHTS

Despite one of the three species being significantly different from the other two and the remaining two displaying a high level of overlap, the predictive model based on functional traits demonstrates a high accuracy. This suggests that despite the challenges posed by the dissimilarity and overlap among species, the model effectively leverages the information contained within the functional traits to make accurate predictions. Such predictive performance underscores the importance and potential of utilizing multivariate approaches, such as neural networks, in ecological studies, particularly in discerning complex patterns and relationships among species with varying ecological roles and adaptations.

More scripts can be found on my GitHub profile