Introduction to machine learning with applications - Report

Anderson E. Schwertner, Diego G. Silva, Liandra dos S. Jesus, Ricardo C. de Oliveira

2022-08-19

Clustering

Objective

Cluster the RCA cells based on the information on the average dissolved oxygen in the water, represented by "DOAVEG_avg", considering the records made during the year 2012.

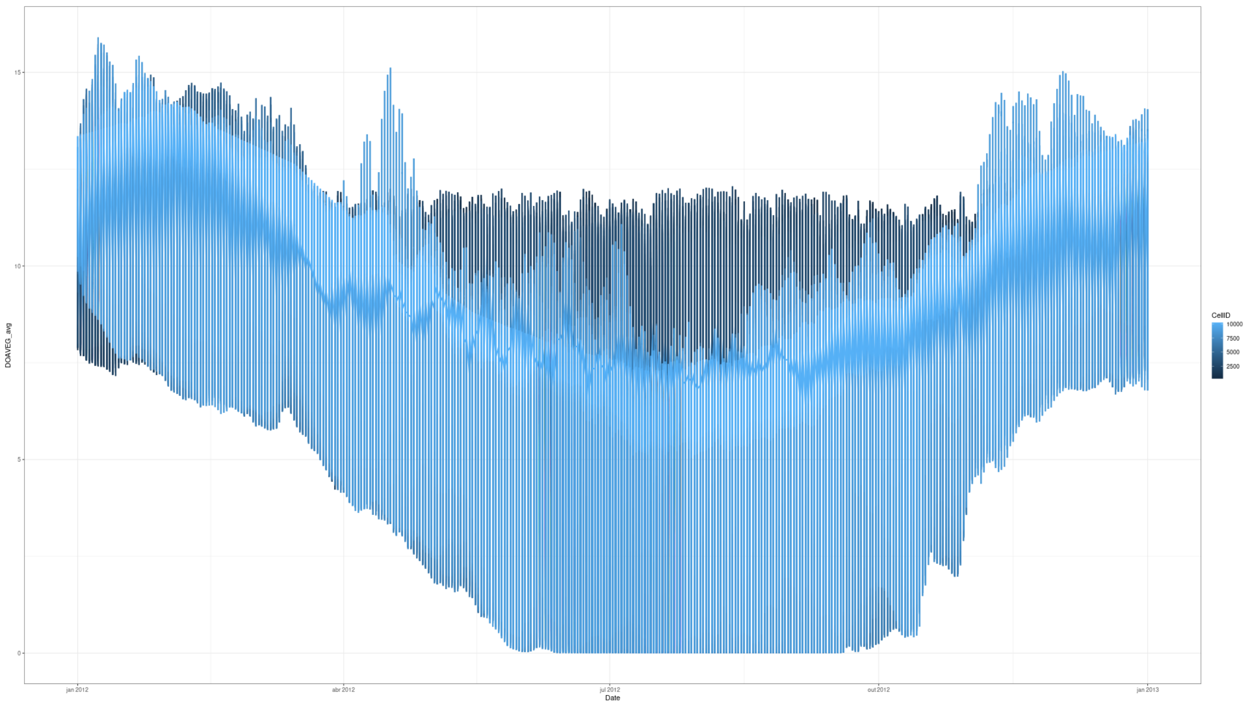

Figure 1: "DOAVEG_avg" time series

Packages

We use the following R packages:

General

- dplyr
- tidyr
- tidyverse

Clustering

- dtw
- cluster
- factoextra

Data manipulations

It was loaded the available dataset, select the columns of interest "Date", "CellID", and "DOAVEG_avg" referring to the year 2012, and computed the monthly average of "DOAVEG_avg" for each one of the RCA cells. Since the original dataset is relatively large, it was saved the already transformed data in a file with the extension ".RDS".

The code below illustrates the procedure performed:

```r
# Clean the workspace
rm(list = ls())

# Load libraries
library(dplyr)
library(tidyr)
library(tidyverse)
library(dtw)
library(cluster)
library(factoextra)

# Load dataset and select data of interest
Drca = read.csv("./dataraw/rca_data_2012_2022-06-29.csv") %>%
  select(Date, CellID, DOAVEG_avg) %>%
  mutate(Date = as.Date(Date)) %>%
  mutate(Month = as.integer(format(Date, "%m")),
         Year = as.integer(format(Date, "%Y"))) %>%
  filter(Year == 2012) %>%
  group_by(CellID, Month) %>%
  summarise(DOAVEG_avg = mean(DOAVEG_avg))

# Save data in RDS file
saveRDS(Drca, "./dataderived/Drca.RDS")
```

Methods and results

Initially, it was taken a sample of 100 time series and reshape the data in order to apply the "hclust" and "agnes" hierarchical clustering algorithms.

```r
# Load data
Drca = readRDS("./dataderived/Drca.RDS")

# Take a sample with 100 time series
CELLS = unique(Drca$CellID)
set.seed(123)
cells = sample(CELLS, 100)
drca = Drca %>%
  filter(is.element(CellID, cells))

# Reshape data to employ clustering algorithms: 'hclust' and 'agnes'
drca_reshaped = spread(drca, key = Month, value = DOAVEG_avg)
```

There are several ways to link the hierarchical cluster, namely “average”, “single”, “complete”, and “ward”. The “agnes” function, from the “cluster” package, returns the agglomeration coefficient of a hierarchical cluster. If the agglomeration coefficient is closer to 1, that suggests a strong agglomeration structure.

```
# Create a vector with the linkage types
linkages <- c( "average", "single", "complete", "ward")
names(linkages) <- c( "average", "single", "complete", "ward")

# Function to compute agglomerative coefficient
ac <- function(x) {
  agnes(drca, method = x)$ac
}

# Agglomerative coefficient
map_dbl(linkages, ac)

# average   single    complete  ward
# 0.9996731 0.9977773 0.9998462 0.9999864
```

According to the results presented, it could observe that all types of linkages give us values close to 1. It was chosen the “complete” method because it tends to produce more compact clusters.

In order to find the optimal number of clusters, it was used the Elbow Method.

```
# Plot of the Elbow method
fviz_nbclust(drca, FUN = hcut, method = "wss") +
  geom_vline(xintercept = 3, linetype = 2)
```

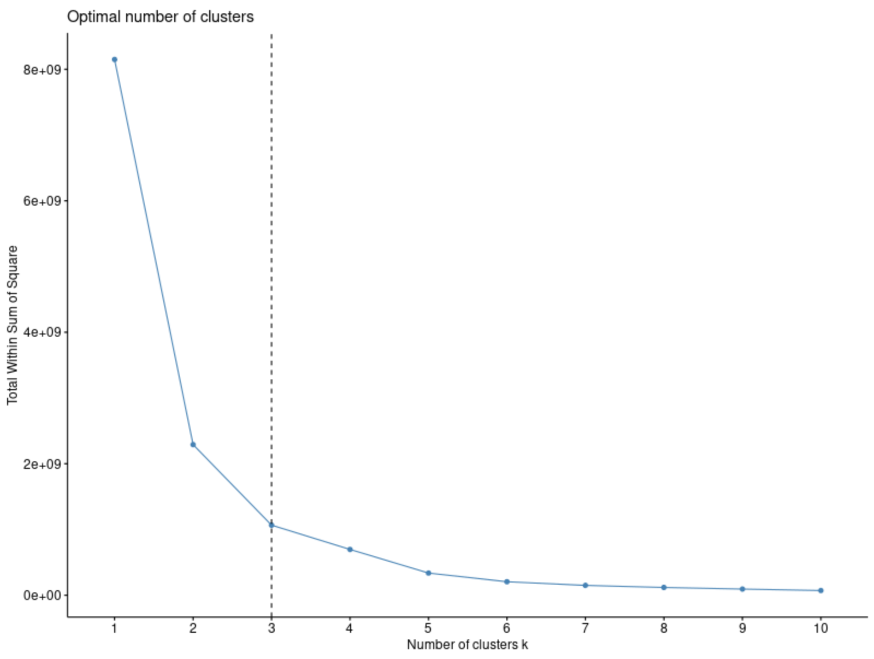

Figure 2: Elbow method.

The figure above suggests that the optimal number of clusters is 3, and the same behavior was observed with a sample of 1000 time series, the Elbow Method also suggests the optimal value of 3 clusters.

Therefore, based on the observations made previously, it was performed hierarchical clustering using “complete” linkage and k=3 clusters. Among the various methods available to compute the dissimilarity matrix, it was chosen dynamic time warping “DTW”, in order to align the time series generated by the RCA cells.

```r
# Reshape data to employ clustering algorithms
Drca_reshaped = spread(Drca, key = Month, value = DOAVEG_avg)

# Compute the dissimilarity matrix
cluster_dist <- dist(Drca_reshaped, method = "DTW")

# Save the data
saveRDS(cluster_dist, "./Drca_dist_dtw.RDS")
```

Finally, it was applied the hierarchical clustering algorithm “hclust”.

```r
# Load data
cluster_dist = readRDS("./Drca_dist_dtw.RDS")

# Hierarchical clustering using 'complete' linkage
hc <- hclust(cluster_dist, method = "complete")

# Cut the hierarchical tree in 3 groups
subgroups <- cutree(hc, k = 3)

# Plot the dendrogram
plot(hc, lwd = 2, hang = -1,
     main = "Cluster Dendrogram of DOAVEG_avg (Complete, DTW)")
par(lwd=4)
rect.hclust(hc, k = 3, border = 1:3)
```

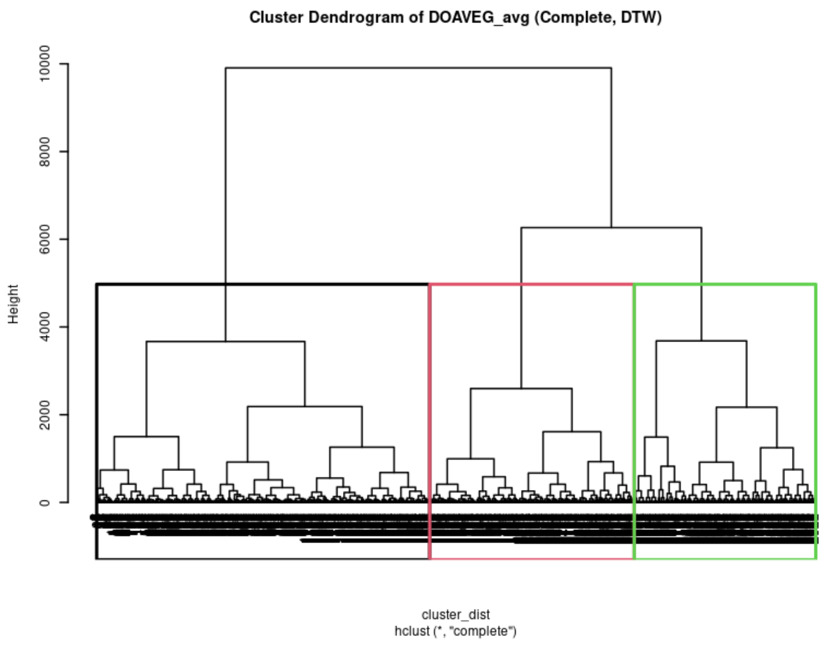

Figure 3: Cluster Dendrogram of DOAVEG_avg.

In addition, it was also ploted the solution obtained with the latitude and longitude data from each of the RCA cells.

```r
# Loading location data
dcells = read.csv("./dataraw/rca_cells_2022-06-29.csv")

# Merge Drca data with the cell location
Drca_locations = merge(Drca, dcells, by = "CellID")

# Plot the clusters
with(Drca_locations,
     plot(x = LON, y = LAT, col = subgroups, pch = 20, pty = "s")
)
```

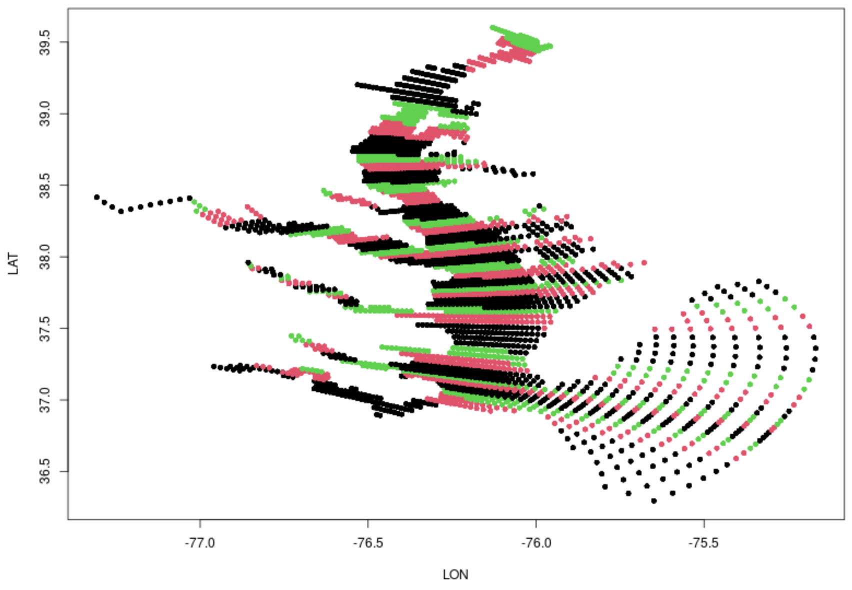

Figure 4: Clusters obtained by Hierarchical Clustering, with "DTW" dissimilarity matrix and "complete" linkage.

Open questions

- Unfortunately, we were unable to run the "agnes" function for the entire dataset. This function has a high computational cost and made the execution time very high (more than 4 hours). With all the agglomeration coefficients calculated, we could make better choices with respect to the type of linkage.
- We also had problems using the elbow method for the entire dataset. Therefore, we had to choose the number of clusters based on smaller samples of 100 and 1000 elements.
- Figure 4 illustrates each one of the RCA cells and their respective cluster. It is possible to notice that the distribution of cells in each cluster seems to follow some rule or trend. When performing hierarchical clustering using the dissimilarity matrix calculated with Euclidean distances and Ward's linkage method, this pattern also seems to repeat itself.

Tree-based methods

Objective

The goal was to predict the average value of dissolved oxygen (“DOAVEG_avg”), based on the variables: average water temperature (“WTEMP_avg”), average salinity (“SAL_avg”), and average concentration of chlorophyll-a (“CHLAVEG_avg”).

Packages

To perform this task, we need the following R packages:

General

- dplyr
- ggplot2

Random Forest

- ranger
- pdp

Data manipulations

Initially, the dataset was loaded and selected the following columns: “Date”, “CellID”, “DOAVEG_avg”, “WTEMP_avg”, “SAL_avg”, and “CHLAVEG_avg”, all of them corresponding to the year 2012. Such columns contain the desired information about the variables that were studied. To reduce the amount of information, the data were grouped by month and corresponding RCA cell and calculated the average of each of the variables. We also carry out the same process considering the annual averages. Finally, these data were saved in files with the extension “.RDS”, to facilitate their manipulation.

```r
# Clean the workspace
rm(list = ls())

# Load library
library(dplyr)

# Load dataset, select data of interest, and compute monthly averages
Drca = read.csv("./dataraw/rca_data_2012_2022-06-29.csv") %>%
  select(Date, CellID, DOAVEG_avg, WTEMP_avg, SAL_avg, CHLAVEG_avg) %>%
  mutate(Date = as.Date(Date)) %>%
  mutate(Month = as.integer(format(Date, "%m")),
         Year = as.integer(format(Date, "%Y"))) %>%
  filter(Year == 2012) %>%
  group_by(CellID, Month) %>%
  summarise_at(c("DOAVEG_avg", "WTEMP_avg", "SAL_avg", "CHLAVEG_avg"), mean)

# Save data in RDS file
saveRDS(Drca, "./dataderived/Drca_forecasting_month.RDS")

# Load dataset, select data of interest, and compute annual averages
Drca = read.csv("./dataraw/rca_data_2012_2022-06-29.csv") %>%
  select(Date, CellID, DOAVEG_avg, WTEMP_avg, SAL_avg, CHLAVEG_avg) %>%
  mutate(Date = as.Date(Date)) %>%
  mutate(Month = as.integer(format(Date, "%m")),
         Year = as.integer(format(Date, "%Y"))) %>%
  filter(Year == 2012) %>%
  group_by(CellID) %>%
  summarise_at(c("DOAVEG_avg", "WTEMP_avg", "SAL_avg", "CHLAVEG_avg"), mean)

# Save data in RDS file
saveRDS(Drca, "./dataderived/Drca_forecasting_year.RDS")
```

Methods and results

Initially, it was loaded libraries that were used, and the data manipulated in the previous step.

```r
# Clean the workspace
rm(list = ls())

# Load Libraries
library(dplyr)
library(ranger)
library(ggplot2)
library(pdp)

# Set seed
set.seed(123)

# Load data
drca_monthly = readRDS("./dataderived/Drca_forecasting_month.RDS")
drca_annual = readRDS("./dataderived/Drca_forecasting_year.RDS")
```

The "ranger" function was executed without modifying the pre-defined arguments, to check which dataset were more promising.

```r
# Perform Random Forest algorithm for monthly average data
rf_drca_monthly = ranger(DOAVEG_avg ~ CHLAVEG_avg + WTEMP_avg + SAL_avg,
                 data = drca_monthly[,-1])
print(rf_drca_monthly)

# OOB prediction error (MSE):       0.4993812
# R squared (OOB):                  0.9418861
```

For the dataset with monthly averages, an MSE of 0.499 and an R-squared value of 0.942 were obtained.

```r
# Perform Random Forest algorithm for annual average data
rf_drca_annual = ranger(DOAVEG_avg ~ CHLAVEG_avg + WTEMP_avg + SAL_avg,
                 data = drca_annual[,-1])
print(rf_drca_annual)

# OOB prediction error (MSE):       0.1317563
# R squared (OOB):                  0.9541816
```

On the other hand, for the second dataset, which only considers the annual averages for each RCA cell, an MSE of 0.131 and an R-squared value of 0.954 were obtained. Therefore, the second dataset was selected because it has the lowest MSE.

The "ranger" function has several importance modes for the variables, namely: "none", "impurity", "impurity_corrected", and "permutation". Thus, to choose the most appropriate way and optimize the number of trees in Random Forest, a comparison was made among them.

```r
# Compare different parameters for importance of variables
nt <- seq(1, 1001, 10)
oob_mse <- vector("numeric", length(nt))
var_importance = c('none', 'impurity', 'impurity_corrected', 'permutation')

par(mfrow=c(2,2))
for(j in 1:length(var_importance)){
  for(i in 1:length(nt)){
    rf <- ranger(DOAVEG_avg ~ CHLAVEG_avg + WTEMP_avg + SAL_avg,
                 data = drca_annual[,-1],
                 importance = var_importance[j],
                 num.trees = nt[i],
                 write.forest = FALSE)
    oob_mse[i] <- rf$prediction.error
  }
  plot(x = nt, y = oob_mse, col = "red", type = "l", main = var_importance[j],
       xlab="Number of trees", ylab="OOB MSE")
}
par(mfrow=c(1,1))
```

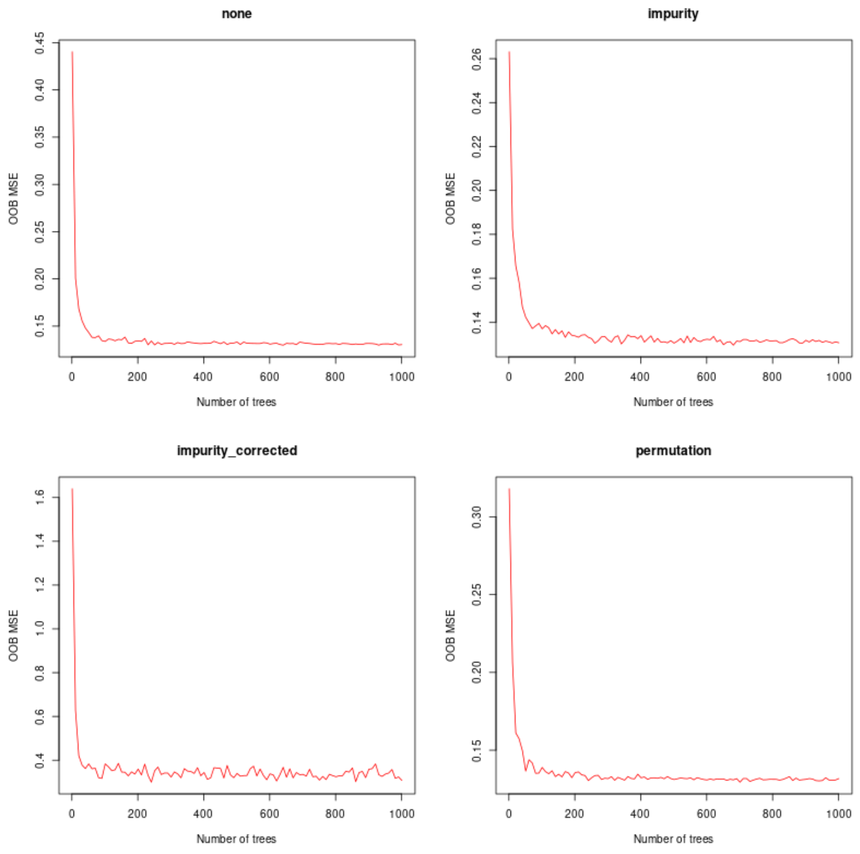

Figure 5: Comparison between OOB MSE and number of trees.

As can be seen in the graph, the “impurity_corrected” mode presented a very different behavior from the others, and the errors were stabilized with about 300 trees. Therefore, it was decided to run the Random Forest algorithm with 300 trees in “permutation” mode.

```r
# Perform Random Forest algorithm for annual average data, with 300 trees
rf_drca = ranger(DOAVEG_avg ~ CHLAVEG_avg + WTEMP_avg + SAL_avg,
                 data = drca_annual[,-1], importance = "permutation",
                 num.trees = 300)
# OOB prediction error (MSE):       0.1310688
# R squared (OOB):                  0.9544207
```

Observing the output of the “ranger” function, an MSE of 0.131 and an r-squared value of 0.954 were obtained. These values were considered satisfactory, and this model was chosen to proceed with the analysis that follows.

Next, the relative importance of each predictor variable for the Random Forest model and its respective significance were studied.

```r
# Plot of variable importance
ranimp <- importance_pvalues(rf_drca, method = "altmann",
                             num.permutations = 500,
                             formula = as.formula(paste(
                               "DOAVEG_avg ~ CHLAVEG_avg + WTEMP_avg + SAL_avg",
                               collapse = " ", sep = " ~ ")),
                             data = drca_annual[,-1])
ranimp <- ranimp[order(ranimp[,1]),]
#               importance    pvalue
# CHLAVEG_avg   2.045404 0.001996008
# WTEMP_avg     2.828762 0.001996008
# SAL_avg       3.234660 0.001996008

tmp <- ranimp[,1]
barplot(tmp,
        beside = TRUE, las = 1, xlim = c(0, 3.5),
        xlab = "Importance",
        col = 1, border = NA, cex.names = 0.6,
        horiz = TRUE)
```

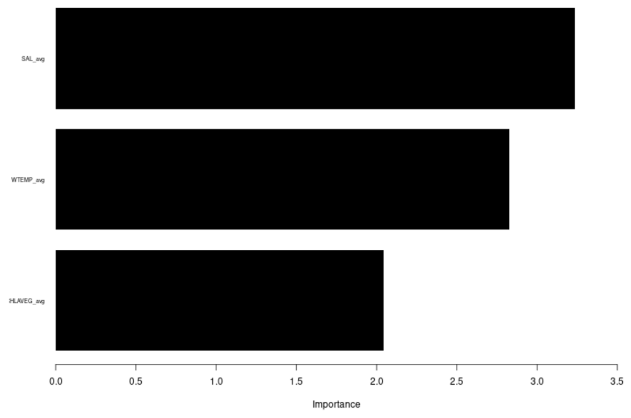

Figure 6: Importance of variables in prediction.

Those values put the different predictors on the same scale and allows you to compare their coefficients directly. Standardized coefficients represent the mean change in the response given a one standard deviation change in the predictor. Considering 95% of confidence level it appears all prediction variables are significant for the model obtained (p-value <5%).

Figure 7 shows the behavior of partial dependence on variables:

```
# Partial dependence plot
par.SAL_avg <- partial(rf_drca, pred.var = c("SAL_avg"), chull = TRUE)
par.WTEMP_avg <- partial(rf_drca, pred.var = c("WTEMP_avg"), chull = TRUE)
par.CHLAVEG_avg <- partial(rf_drca, pred.var = c("CHLAVEG_avg"), chull = TRUE)

par(mfrow=c(1,3))
plot(par.SAL_avg, contour = TRUE, type="l", ylab="DOAVEG_avg")
plot(par.WTEMP_avg, contour = TRUE, type="l", ylab="DOAVEG_avg")
plot(par.CHLAVEG_avg, contour = TRUE, type="l", ylab="DOAVEG_avg")
par(mfrow=c(1,1))
```

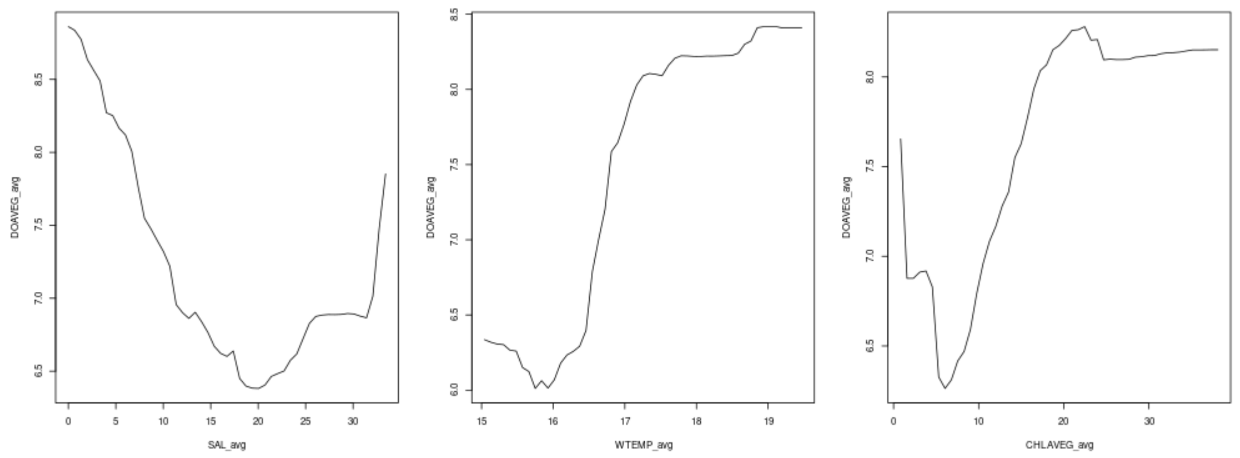

Figure 7: Partial dependence.

The figure above suggests that:

- The concentration of dissolved oxygen in the water tends to decrease as the salinity increases to values close to 20 ppt, and then increases again.
- There is a decrease in oxygen concentration until about 16°C, followed by a sharp increase until about 17°C, stabilizing with a slight increase afterwards.
- The oxygen concentration decreases until about 7 ug/L of chlorophyll-a, and then shows a sharp increase until 22 ug/L, stabilizing afterwards.

Open questions

- Use more variables (or other variables) to try to explain the behavior of the mean oxygen concentration.
- Study the correlation matrix of variables.
- Trying to understand the physical-chemical relationship between the variables adopted and the other available data, seeking to list the factors of greatest influence.

Neural networks

Objective

Although, it was also aimed to predict the average value of dissolved oxygen ("DOAVEG_avg"), based on the same variables ("WTEMP_avg", "SAL_avg", CHLAVEG_avg"), it was applied a machine learning.

Packages

We use the following packages:

General

- dplyr
- plyr

Neural networks

- neuralnet

Data manipulations

It was used the same dataset in the forecast example removing the CellID column. The sample(x, size) function simply outputs a vector of the specified size of randomly selected samples from the vector x. By default, the sampling is without replacement: index is essentially a random vector of indices. Then 75% of the dataset were used for training and 25% for testing. With the min-max method and scale the data in the interval [0,1], the data was normalized and then split.

```
# Clean the workspace
rm(list = ls())

# Load library
library(dplyr)
library(plyr)

# Set seed
set.seed(450)

# Load data
drca = readRDS("./dataderived/Drca_forecasting_year.RDS")

# Exclude the first column of the dataset
data = drca[-1]

# Select the indexes for train and test data (75% of the data for train)
index = sample(1:nrow(data), round(0.75*nrow(data)))
train = data[index,]
test = data[-index,]

# Normalize the data
maxs = apply(data, 2, max)
mins = apply(data, 2, min)
scaled = as.data.frame(scale(data, center = mins, scale = maxs - mins))
train_ = scaled[index,]
test_ = scaled[-index,]
```

Methods and results

It was used the "neuralnet" package to construct neural networks. In the neuralnet() function, the formula y~ is not accepted it, so first it was wrote the formula and then passed it as an argument in the fitting function. Additionally, the hidden argument accepts a vector with the number of neurons for each hidden layer, as doesn't existed a rule to how many layers and neurons to use, by trying different combinations it ended in 3 hidden layers with 9, 6 and 3 neurons. Also, the argument linear.output is used to specify whether we want to do regression linear.

```
# Load library
library(neuralnet)

# Define and train the neural network
n = names(train_)
f = as.formula(paste("DOAVEG_avg ~", paste(n[!n %in% "DOAVEG_avg"],
                                            collapse = " + ")))
nn = neuralnet(f, data=train_, hidden=c(9,6,3), linear.output=T)

# Plot the neural network
plot(nn)
```

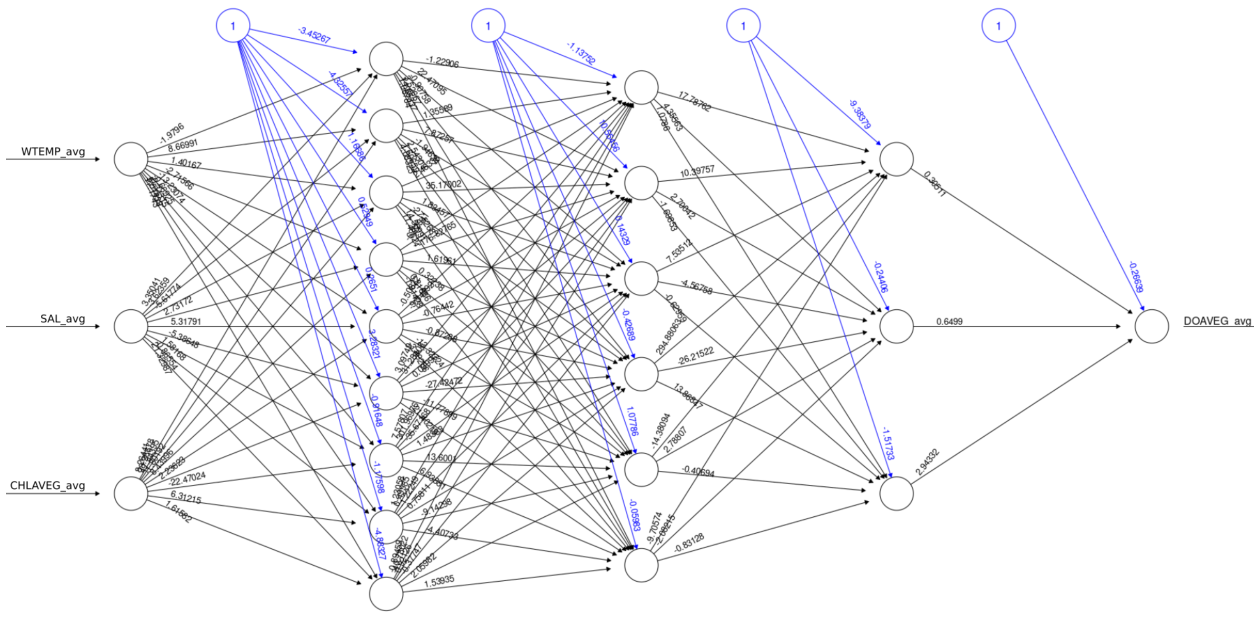

Figure 8: Neural network.

The black lines from the Figure 8 shows the connections between each layer and the weights on each connection while the blue lines show the bias term added in each step. The bias can be thought as the intercept of a linear model.

```
# Make predictions
pr.nn = compute(nn, test_[, 1:4])
pr.nn_ = pr.nn$net.result * (max(data$DOAVEG_avg) - min(data$DOAVEG_avg)) +
  min(data$DOAVEG_avg)
test.r = test_$DOAVEG_avg * (max(data$DOAVEG_avg) - min(data$DOAVEG_avg)) +
  min(data$DOAVEG_avg)
```

To measure how much the predictions are far away from the real data, it was calculated the Mean Squared

Error (MSE), Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

```r
# Compute the errors MSE, RMSE and MAE
e = test.r - pr.nn_
MSE = mean(e^2)
RMSE = sqrt(mean(e^2))
MAE = mean(abs(e))

print(paste(MSE, RMSE, MAE))
#"0.146814374945054 0.383163639904746 0.260301935369804"
```

Figura 9 shows the real vs predicted data. It can be noticed that a good fit on those conditions.

```r
# Plot the real versus predicted data
plot(test$DOAVEG_avg, pr.nn_, col='blue', main='Real vs predicted (DOAVEG_avg)',
     pch=18, cex=0.9, xlab='Real', ylab='Predicted') +
abline(0, 1, lwd=3, col='red')
```

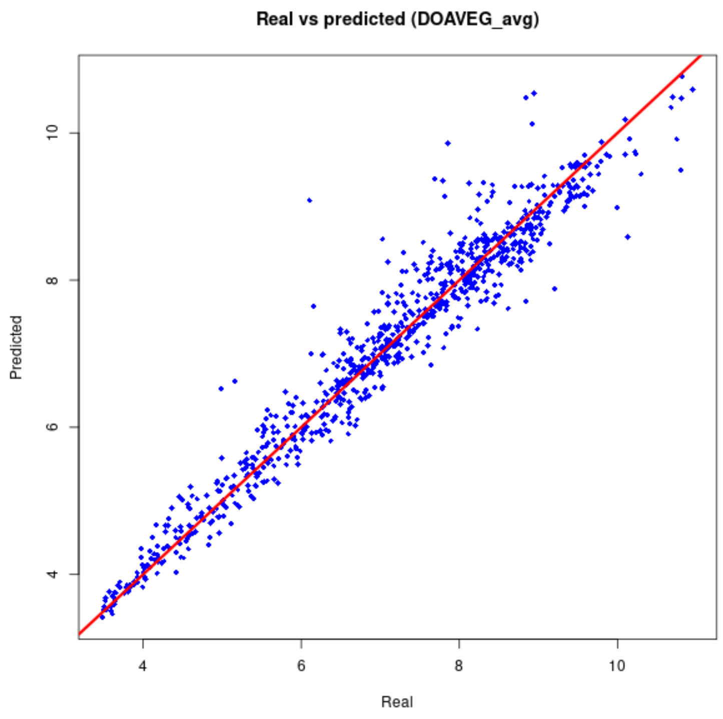

Figure 9: Real vs predicted data.

Open questions

- Perform cross-validation of the model we found.
- Test the performance of convolutional neural networks and LSTM neural networks.
- Employ the entire database to make predictions, using individual observations or monthly averages.
- Test other variables as features of the regression problem.

Conclusions

To conclude, all scripts presented in this work were executed in R language, version 4.2.1, on an AMD Ryzen 7 1700X machine with 16 Gb DDR3 RAM and Ubuntu 22.04.1 LTS operating system. So, the clustering was the first method implemented and used a hierarchical one, which was required around three hours get the results.

Additionally, it was programmed two regression techniques, Random Forest (RF) and Neural Network. They both had the same dataset, which made possible a comparation between them. As their MSE resulted in 0.131 and 0.147, respectively, it is noted that they both had similar results, but RF was best compared to the value error and computational cost.