Using machine learning to understand the drivers of nutrients in constrasting coastal ecosystems

Matt Duggan

2021-06-16

Introduction

We are interested in understanding and identifying the most important variables that affect bigeochemical signals in water chemistry within estuaries, arguably the most valuable bodies of water for directly and indirectly maintaining every ecosystem on Earth. In order to evaluate these variables that act as predictors for these signals, we are constructing random forests to predict variable importance for four significant bio signatures: ammonium (NH4), phosphate (PO4), nitrate (NO3), and Chlorophyll A (chla). Functions for the script can be found in the Functions/train_Rforest_functions.R scipt and constants, such as predictors are in the Constants/train_model_constants.R script.

Training

Data Preperation

We use approximately 20 years of data collected by the National Estuaries Research Reserve (NERR).The data is saved across years and for each station:

1. Old Women Creek (tributary to Lake Erie)
2. York River Estuary (tributary to Chesapeake Bay)

Therefore for each station, the data must be combined while taking into consideration the removal of outliers and missing data. As a forewarning, for the entirety of the script we try to stay consistent with the tidyverse syntax and conventions, so we highly suggest reading the tidyverse/tidymodels documentation before continuing deeper in the script.

```r
# prepare data from cbv
cbv_all <- read_station("./data_NERR/output/cbv_for_models.csv") %>%
    filter(is.na(no3) | no3 < 1, is.na(po4) | po4 < 0.15, is.na(chla) | chla < 200)

# prepare data from owc
owc_all <- read_station("./data_NERR/output/owc_for_models.csv") %>%
    filter(is.na(no3) | no3 < 8, is.na(po4) | po4 < 0.1)
```

Train Model Preperation

Since we are experimenting with different predictors, the best way to run training is to run them in parallel. This is the standard process for setting up the ports and running them in a cluster. Refer to the parallel package in base R for more details.

```r
#Create an apply function for parrallel computing
numCores <- detectCores()-1

#START cluster
cl <- makeCluster(numCores, outfile ='')

#export required constants
clusterExport(cl,
              c("cbv_all",
              "owc_all"))

#Export necessary libraries
clusterEvalQ(cl, {
  library(ggplot2)
  library(tidyverse)
  library(tidymodels)
  library(tidyverse)
  library(lubridate)
  library(ggthemes)
  library(hydroGOF)
  library(DALEXtra)
  source("Functions/train_Rforest_functions.R")
  source("Constants/initial_model_constants.R")
})
```

Reference Table

The reference table acts as a source for setup of project. For each run, we defined a row for the random forest model training that included:

1. Chemical Signal
   - ammonia (NH4)
   - nitrate (NO3)
   - phosphate (PO4)
   - chlorophyll a (CHLA)
2. Predictors
   - water quality (wq_predictors)
   - meteorology (met_predictors)
   - all (all_predictors)

As a side note: highly suggest to include the station and model as new columns and refer them here.

Train Random Forest

We ran two model architectures available in two different R packages:

1. randomForest
2. ranger

dep	predictor	name
nh4	met_predictors	nh4-met_predictors
nh4	wq_predictors	nh4-wq_predictors
nh4	all_predictors	nh4-all_predictors
po4	met_predictors	po4-met_predictors
po4	wq_predictors	po4-wq_predictors
po4	all_predictors	po4-all_predictors
no3	met_predictors	no3-met_predictors
no3	wq_predictors	no3-wq_predictors
no3	all_predictors	no3-all_predictors
chla	met_predictors	chla-met_predictors
chla	wq_predictors	chla-wq_predictors
chla	all_predictors	chla-all_predictors

Each model was created with the three aformentioned predictor groups:

Therefore, we will have 2(model types) x 3(predictor groups) x 4(signatures) separate random forest models to compare. mtry (number of predictors randomly sampled for each branching) is cross validated for optimization of the forest for each group. The number of trees (ntrees) will continuosly increase the accuracy of the model, however, at the cost of computation resources. Therefore, we chose to stick with 1000 trees. Reference the train_Rforest_functions.R scripts for more information on training. The training was mainly conducted in the conventions of the tidymodels R package.

```
#Train data on cbv location with ranger
result_cbv_ranger <- parApply(cl,reference_table,1,
                              function(x) choose_inputs(
                                                        cbv_all,
                                                        x[1],
                                                        eval(parse(text = x[2])),
                                                        x[3],
                                                        modelType = "ranger",
                                                        importance = "impurity"))
#Train data on owc location with ranger
result_owc_ranger <- parApply(cl,reference_table,1,
                              function(x) choose_inputs(
                                                        owc_all,
                                                        x[1],
                                                        eval(parse(text = x[2])),
                                                        x[3],
                                                        modelType = "ranger",
                                                        importance = "impurity"))
#Train data on cbv location with random forest
result_cbv_rf <- parApply(cl,reference_table,1,
                          function(x) choose_inputs(
                                                    cbv_all,
                                                    x[1],
                                                    eval(parse(text = x[2])),
                                                    x[3],
                                                    modelType = "randomForest",
                                                    importance = TRUE))
#Train data on owc location with random forest
result_owc_rf <- parApply(cl,reference_table,1,
                          function(x) choose_inputs(
                                                    owc_all,
                                                    x[1],
                                                    eval(parse(text = x[2])),
                                                    x[3],
                                                    modelType = "randomForest",
                                                    importance = TRUE))

#END parrallel processing
stopCluster(cl)
```

Evaluate

Table of Metrics

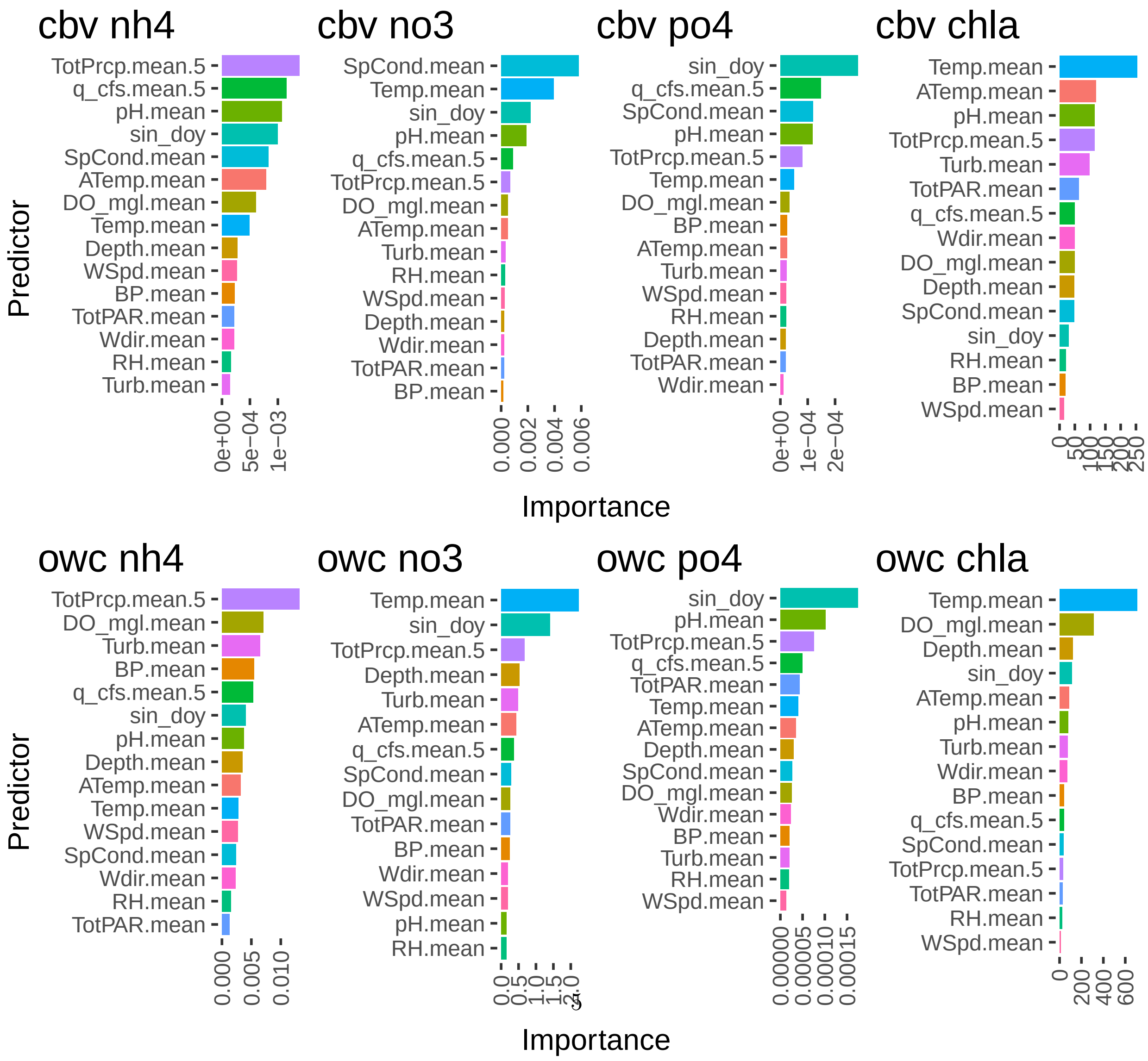
The table of metrics (Table 1) demonstrates the performance of the regression models between the two random forest packages. Across the two sites, randomForest outperforms ranger and, therefore, we chose to continue with randomForest in the rest of our predictions.

Impotance Plots

In figure 1, you can see the variable importance based on the Gini impurity. The Gini impurity is an accumulation of the predictor to split to continue on it the decision tree.

Table 1: Table 1. Metrics of mean average error (MAE), root mean square error (RMSE), and Nash-Sutcliffe model efficiency coefficient (NSE). Across the two locations of Cheasepeak Bay (cbv) and Old Woman Creek (owc) and the different signatures of ammonia, nitrate, phosphate, and chlorophyll a, the randomForest package out-performs ranger.

		ranger						randomForest					
		CBV			OWC			CBV			OWC		
Signal	Predictor	MAE	RMSE	NSE	MAE	RMSE	NSE	MAE	RMSE	NSE	MAE	RMSE	NSE
nh4	met	0.02	0.03	0.50	0.05	0.07	0.67	0.02	0.03	0.57	0.05	0.08	0.66
nh4	wq	0.02	0.03	0.62	0.04	0.06	0.79	0.02	0.03	0.62	0.04	0.06	0.79
nh4	all	0.02	0.03	0.58	0.05	0.08	0.74	0.02	0.03	0.58	0.05	0.08	0.73
po4	met	0.01	0.01	0.70	0.01	0.01	0.34	0.01	0.01	0.64	0.01	0.01	0.34
po4	wq	0.01	0.01	0.78	0.00	0.01	0.52	0.01	0.01	0.75	0.00	0.01	0.49
po4	all	0.01	0.01	0.74	0.00	0.01	0.54	0.01	0.01	0.76	0.00	0.01	0.54
no3	met	0.04	0.06	0.52	0.49	0.88	0.69	0.03	0.06	0.59	0.49	0.88	0.68
no3	wq	0.03	0.04	0.79	0.42	0.76	0.76	0.03	0.04	0.78	0.43	0.78	0.74
no3	all	0.03	0.04	0.79	0.57	0.94	0.69	0.03	0.04	0.78	0.57	0.95	0.68
chla	met	6.69	10.19	0.48	8.65	16.39	0.60	6.66	10.37	0.49	8.35	16.49	0.59
chla	wq	6.20	9.85	0.56	6.08	12.83	0.69	6.67	12.14	0.53	6.06	12.81	0.69
chla	all	6.02	9.69	0.49	5.02	8.16	0.85	6.13	10.82	0.58	5.06	8.31	0.84



Partial Dependency Plots

These are the partial dependency plots for each of the predictors in water quality for both locations.

High Frequency Data Predictions

These are the predictions based on the high frequency data. Because of 1) the minor differences in the performance of the model from differences in predictors (Table 1) and 2) the high frequency data was missing a large amount of data for meteorological predictors, most likely due to a down weather station for several years, we chose to predict the chemical signatures with water quality predictors alone with the randomForest architecture.

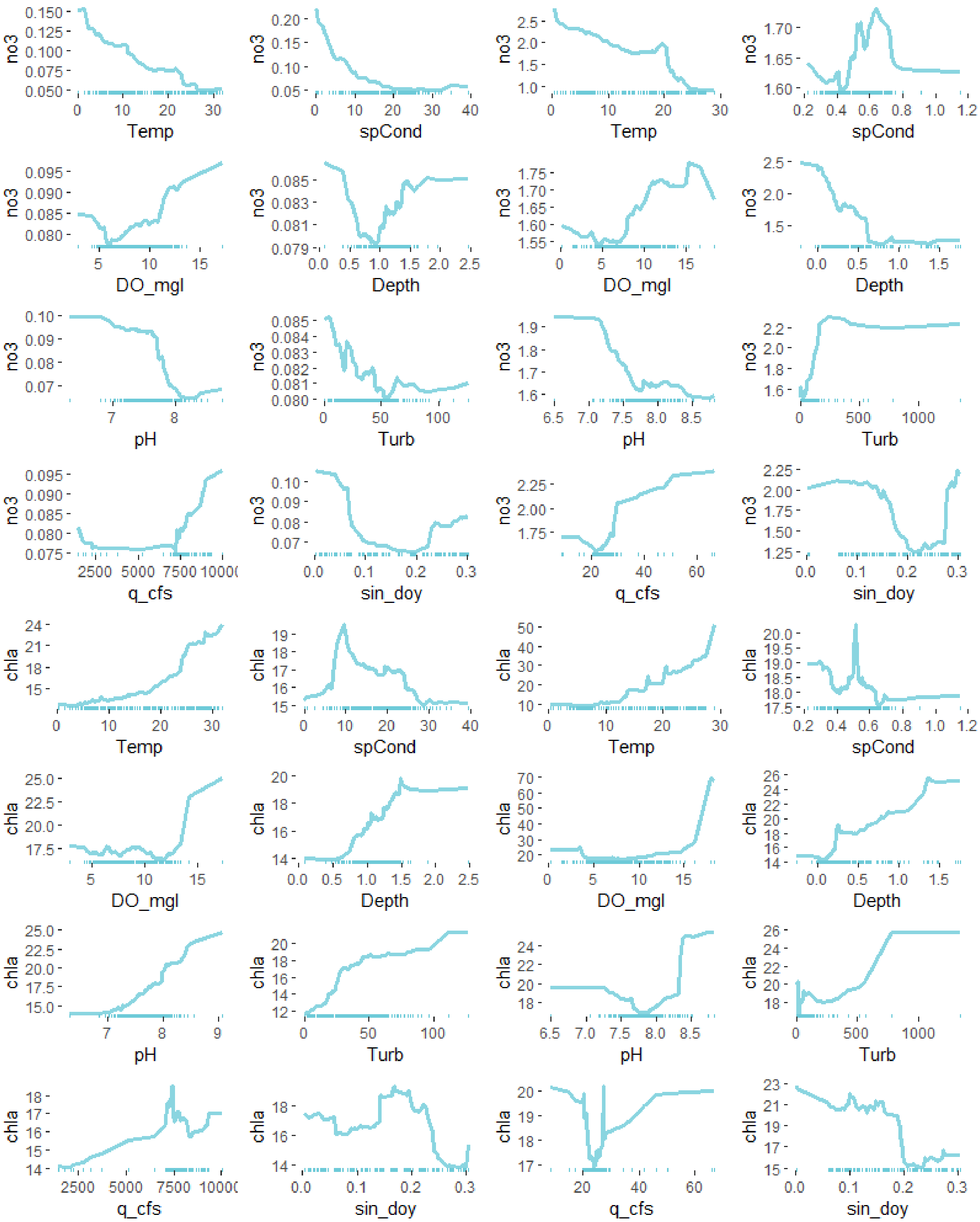


Figure 1: Partial Dependency Plots based on Water Quality Predictors (CBV & OWC)

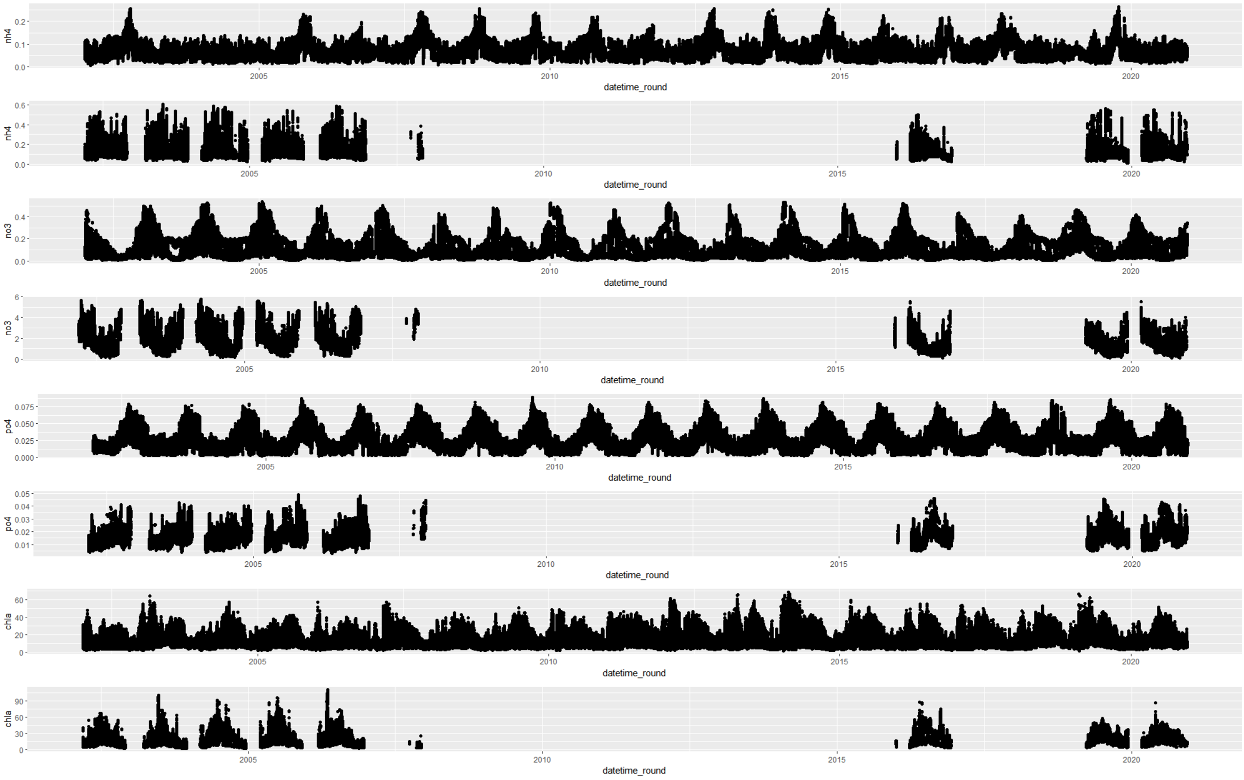


Figure 2: High Frequency prediction data based on water quality predictors

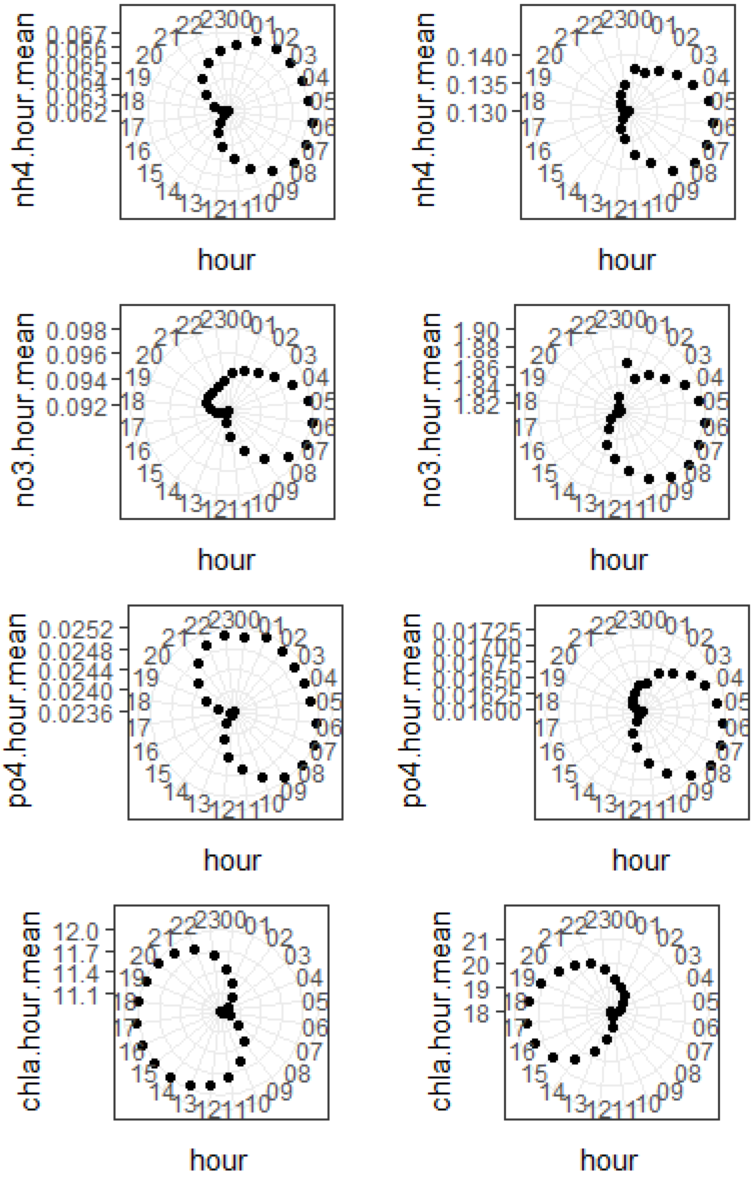


Figure 3: Average of each signature every hour through the day

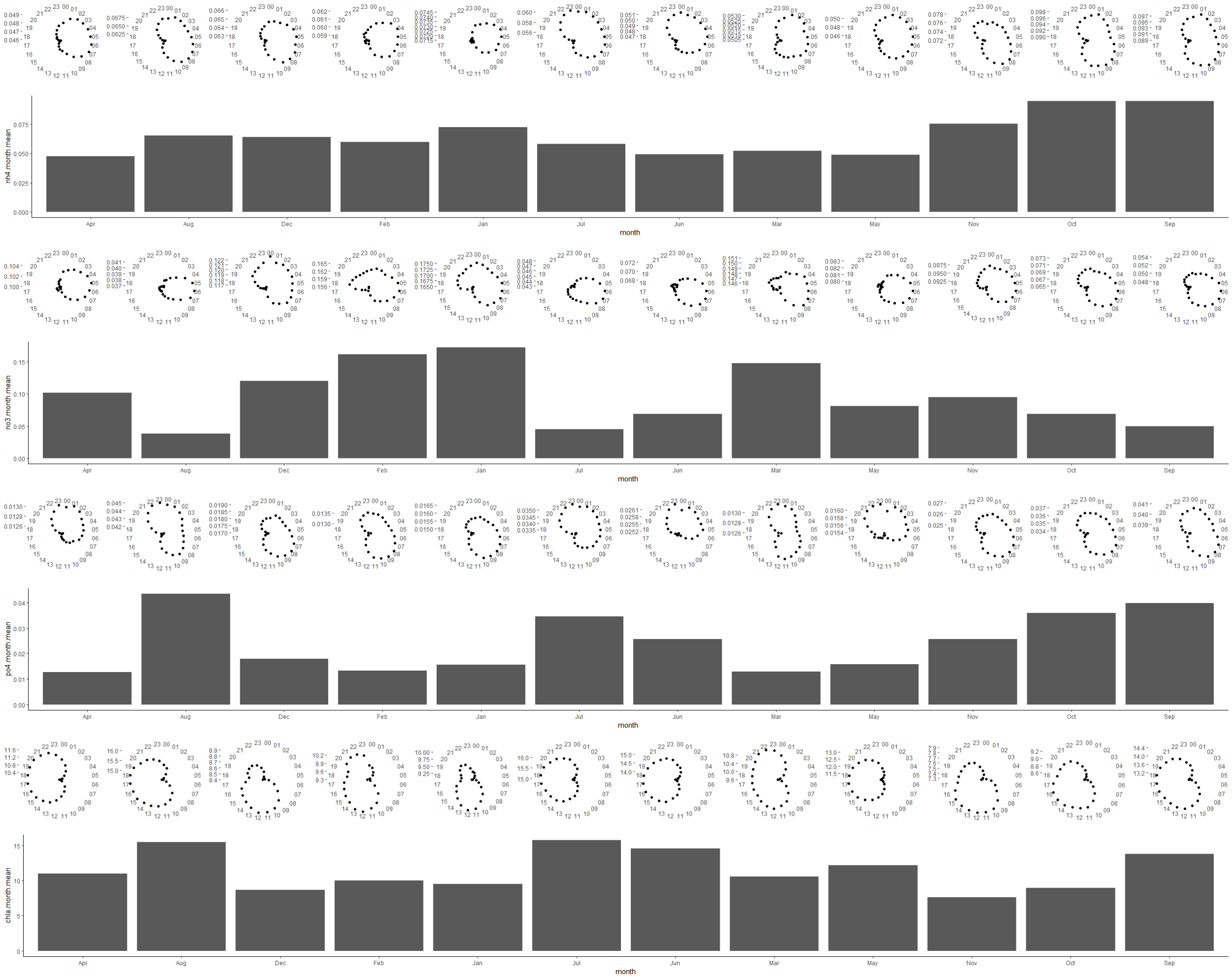


Figure 4: Average of each signature each month at CBV

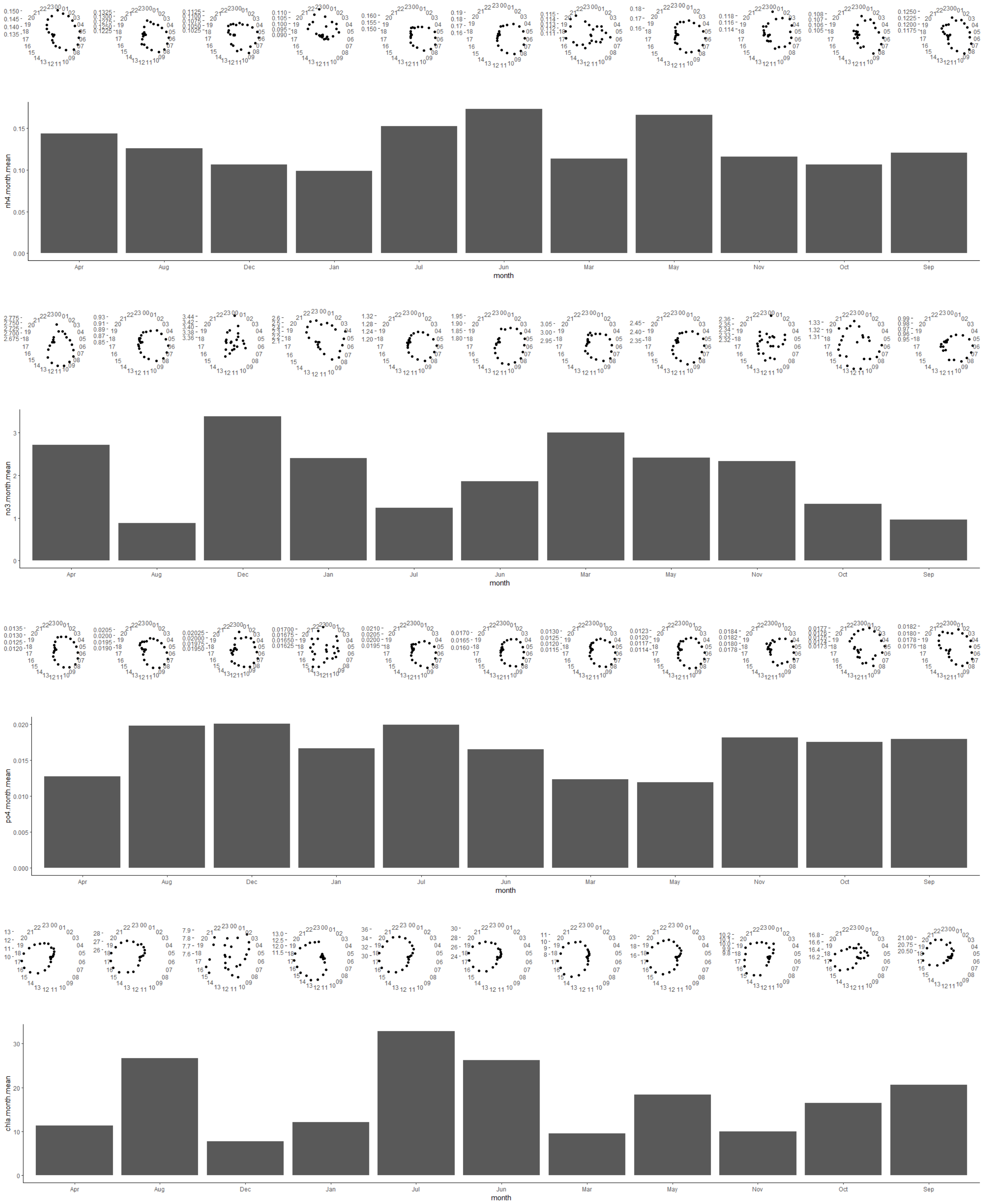


Figure 5: Average of each signature each month at OWC