Environmental Drivers of Golden Tilefish Landings*

Vyacheslav Lyubchich and Geneviève Nesslage
Chesapeake Biological Laboratory, University of Maryland Center for Environmental Science,
Solomons, MD, 20688, USA

2020-04-11

Contents

*These data and code support the results presented in (recommended citation):

Nesslage G, Lyubchich V, Nitschke P, Williams E, Grimes C, Wiedenmann J (2020) Environmental drivers of golden tilefish (*Lopholatilus chamaeleonticeps*) landings and catch per unit effort. In progress.

Citation for the current data and code:

Lyubchich V, Nesslage G (2020) github.com/vlyubchich/tilefish: Environmental Drivers of Golden Tilefish Landings. Zenodo. DOI 10.5281/zenodo.3743867. https://doi.org/10.5281/zenodo.3743867.

1 Packages and functions

```
rm(list = ls())
#Packages and custom functions used
#All packages are needed, but some are not loaded fully to avoid conflicts
library(Boruta)
# library(dplyr)
# library(forecast)
library(funtimes)
library(GGally)
# library(Hmisc)
library(mgcv)
library(ranger)
library(randomForest)
# library(plotmo)
library(xtable)
```

```
# Organize summaries for tables
summaryVL = function(X){
  out = sapply(c(1:ncol(X)), function(i) summary(X[,i])[1:6])
  colnames(out) = names(X)
  if(any(is.na(X))){
    Missing = apply(is.na(X), 2, sum)
    out = rbind(out, Missing)
  }
  out
}
```

```
# Forward-selection of variables for a GAM. See the exact specification in mgcv:: below.
gamforward = function(y, x, data, verbose = TRUE) {
  #y is a character name of the response variable;
  #x is a character vector of x-variables, with most important first (order matters);
  #data is a data frame containing y and x;
  #verbose is a logical values indicating to print out the progress or not.
  K = 5
  alpha = 0.05
  CCthresh = 0.7
  problems = c("non-significance", "concurvity")
  selected = character()
  for(i in 1:length(x)) {
    candidate = x[i]
    problem_check = rep(NA, length(problems))
    f = as.formula(paste0(y, " ~ s(", paste0(c(selected, candidate), collapse = ", k = K) + s("), ", k = K)"))
    z = FALSE
    tryCatch(
      mod  <<-  mgcv::gamm(f
                           , select = TRUE
                           , bs = "cr"
                           , method = "ML"
                           , correlation = corARMA(p = 1, q = 0)
                           , control = lmeControl(msMaxIter = 100)
                           , data = data
      ), error = function(c) {
        z <<- TRUE
        # assign("z", TRUE)
      }
    )
    if (z) {
      if(verbose) {print(paste0("Rejected ", candidate, " due to model non-convergence!!!"))}
    } else {
      #check significance
      fixeff = summary(mod$lme)$tTable[-1, , drop = FALSE]
      pv = fixeff[,ncol(fixeff)]
      problem_check[1] = !all(pv < alpha) #TRUE will indicate a problem
      #check concurvity
      cc = concurvity(mod$gam)[3,-1]
      problem_check[2] = !all(cc < CCthresh) #TRUE will indicate a problem
      if (all(!problem_check)) { #if no problems
        selected = c(selected, candidate)
        if(verbose) {print(paste0("Added ", candidate))}
```

```
        # acf(residuals(modflme, type = "normalized"), las = 1, main = candidate)
      } else {
        if(verbose) {print(paste0("Rejected ", candidate, " due to ", paste0(problems[problem_check], collapse = " and ")))}
      }
      rm(z)
    }
  }
  print(paste0("Selected variables: ", paste0(selected, collapse = ", ")))
  return(selected)
}
```

Some colors to start with

```
COL = c(black = "black"
        ,red = rgb(100, 38, 33, maxColorValue = 100)
        ,green = rgb(38, 77, 19, maxColorValue = 100)
        ,blue = rgb(28, 24, 61, maxColorValue = 100)
        ,purple = rgb(76, 32, 72, maxColorValue = 100)
        ,cyan = rgb(21, 75, 87, maxColorValue = 100)
        ,dark_cyan = rgb(0, 47, 59, maxColorValue = 100)
        ,yellow = rgb(99, 90, 13, maxColorValue = 100)
)
par(mar = c(0, 0, 0, 0), mgp = c(0, 0, 0))
barplot(rep(1, length(COL)), col = COL, border = NA, axes = FALSE, space = c(0, 0))
```

2 Data pre-processing

2.1 Load

```
D = read.csv("./dataraw/landings.csv", stringsAsFactors = FALSE)
D = D[order(D$Year),]
rownames(D) = D$Year
#convert to the (unordered) factor:
D$Time_block = factor(D$Time_block, levels = unique(D$Time_block))
```

Table 1 gives a summary of the numeric variables (the last row is the number of missing values, i.e., NA's).

```
tmp = summaryVL(D[, -3])
```

Table 1: Summary statistics for numeric variables in the dataset

	Year	Landings	AMO_annual	AMO_DJFMA	NAO_DJF_st	NAO_DJF_PC	NAO_DJFMA_PC	NAO_DJFMA_st
Min.	1915.00	5.00	-0.43	-0.41	-5.20	-2.97	-2.14	-2.82
1st Qu.	1940.50	450.00	-0.15	-0.15	-1.30	-0.68	-0.55	-0.53
Median	1966.00	767.50	0.01	-0.02	0.40	-0.02	-0.02	0.16
Mean	1966.00	983.18	-0.00	-0.03	0.10	-0.03	-0.04	0.05
3rd Qu.	1991.50	1208.25	0.14	0.13	1.45	0.72	0.49	0.66
Max.	2017.00	4501.00	0.36	0.33	4.60	2.44	1.87	2.36
Missing	0.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2 provides a summary of the categorical variables.

```
tmp = summary(D[, 3, drop = FALSE])
rownames(tmp) = NULL
```

Table 2: Summary statistics for categorical variables in the dataset

	Time_block
1	Campaign: 6
2	Early :20
3	WWII : 5
4	PostWWII:25
5	Longline:30
6	FMP :17

2.2 Create lagged series, remove FMP period

Fisher et al (2014) suggest that landings are correlated with the Dec–Feb station-based NAO index lagged by up to seven years, so we add those lagged values to the analysis (and lagged values of the other climate indices).

```
n = names(D)
v = grep("AMO_|NAO_", n) #variables to lag
L = 7 #number of lags to add to variables v
for (i in v) { #i=4
  tmp = sapply(1:L, function(x) dplyr::lag(D[,i], x))
  colnames(tmp) = paste(n[i], "_l", 1:L, sep = "")
  D = cbind(D, tmp)
}
```

Landings data from 2001–2017 (the terminal time block FMP) were removed from the analysis to exclude landings that were quota-limited.

```
#Save a copy of the full dataset as D0, cut off the FMP period with fishing restrictions
D0 = D
D = D[D$Time_block != "FMP", ]
D$Time_block = droplevels(D$Time_block)
```

2.3 Tests and visualizations

The landings time series has few missing values that can be reasonably guessed with linear interpolation (see Figure 1):

```
indNA = is.na(D$Landings) #save the location of what we fill-in for later
D$Landings = forecast::na.interp(D$Landings) #interpolate NAs
(TBend = tapply(D$Year, D$Time_block, max)) #end year of each time block

## Campaign    Early     WWII PostWWII Longline
##     1920     1940     1945     1970     2000

(TBcenter = round(tapply(D$Year, D$Time_block, mean))) #year-center of each time block

## Campaign    Early     WWII PostWWII Longline
##     1918     1930     1943     1958     1986
```

Try to remove the right skewness using a power transformation (log is too much, use square root):

```
D$sqrtLandings = sqrt(D$Landings)
```

Apply a non-parametric test for trends (Lyubchich et al, 2013) that is suitable for autocorrelated and conditionally heteroskedastic data. Specifically, we test the null hypothesis that the unknown trend function $\mu(t)$ in the observed time series is constant (i.e., the hypothesis of no trend) vs. the alternative of some other trend (including the cases of a linear, monotonic, or non-linear trend):

$$H_0 : \mu(t) \equiv 0 \quad \text{vs.} \quad H_1 : \mu(t) \neq 0. \tag{1}$$

```
set.seed(111)
wavk.test(D$Landings ~ 1,
          factor.length = "adaptive.selection", ar.method = "yw")

##
##  Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
## data:  D$Landings
## WAVK test statistic = 5, adaptively selected window = 6, p-value = 0.002
## alternative hypothesis: trend is not of the form D$Landings ~ 1.

wavk.test(D$sqrtLandings ~ 1,
          factor.length = "adaptive.selection", ar.method = "yw")

##
##  Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
## data:  D$sqrtLandings
## WAVK test statistic = 3, adaptively selected window = 6, p-value = 0.006
## alternative hypothesis: trend is not of the form D$sqrtLandings ~ 1.

set.seed(222)
wavk.test(DO$AMO_DJFMA ~ 1,
          factor.length = "adaptive.selection", ar.method = "yw")

##
##  Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
## data:  DO$AMO_DJFMA
## WAVK test statistic = -0.4, adaptively selected window = 4, p-value = 0.8
## alternative hypothesis: trend is not of the form DO$AMO_DJFMA ~ 1.

wavk.test(DO$AMO_annual ~ 1,
          factor.length = "adaptive.selection", ar.method = "yw")

##
##  Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
## data:  DO$AMO_annual
## WAVK test statistic = -0.5, adaptively selected window = 4, p-value = 0.8
## alternative hypothesis: trend is not of the form DO$AMO_annual ~ 1.
```

```
par(mar = c(4, 4, 0.1, 1) + 0.1, mgp = c(3, 0.7, 0), mfrow = c(3, 1))
attach(D)
plot(x = Year, y = Landings, las = 1, type = "o", xlab = "", ylim = c(0, 5000))
points(x = Year[indNA], y = Landings[indNA], bg = COL["red"], pch = 21)
abline(v = TBend + 0.5, lty = 2, col = COL["green"])
text(x = c(1916, TBcenter[-1]), y = 4900, col = COL["green"], labels = names(TBcenter))
plot(x = Year, y = AMO_DJFMA, las = 1, type = "o", xlab = "")
plot(x = Year, y = NAO_DJFMA_PC, las = 1, type = "o")
detach(D)
```

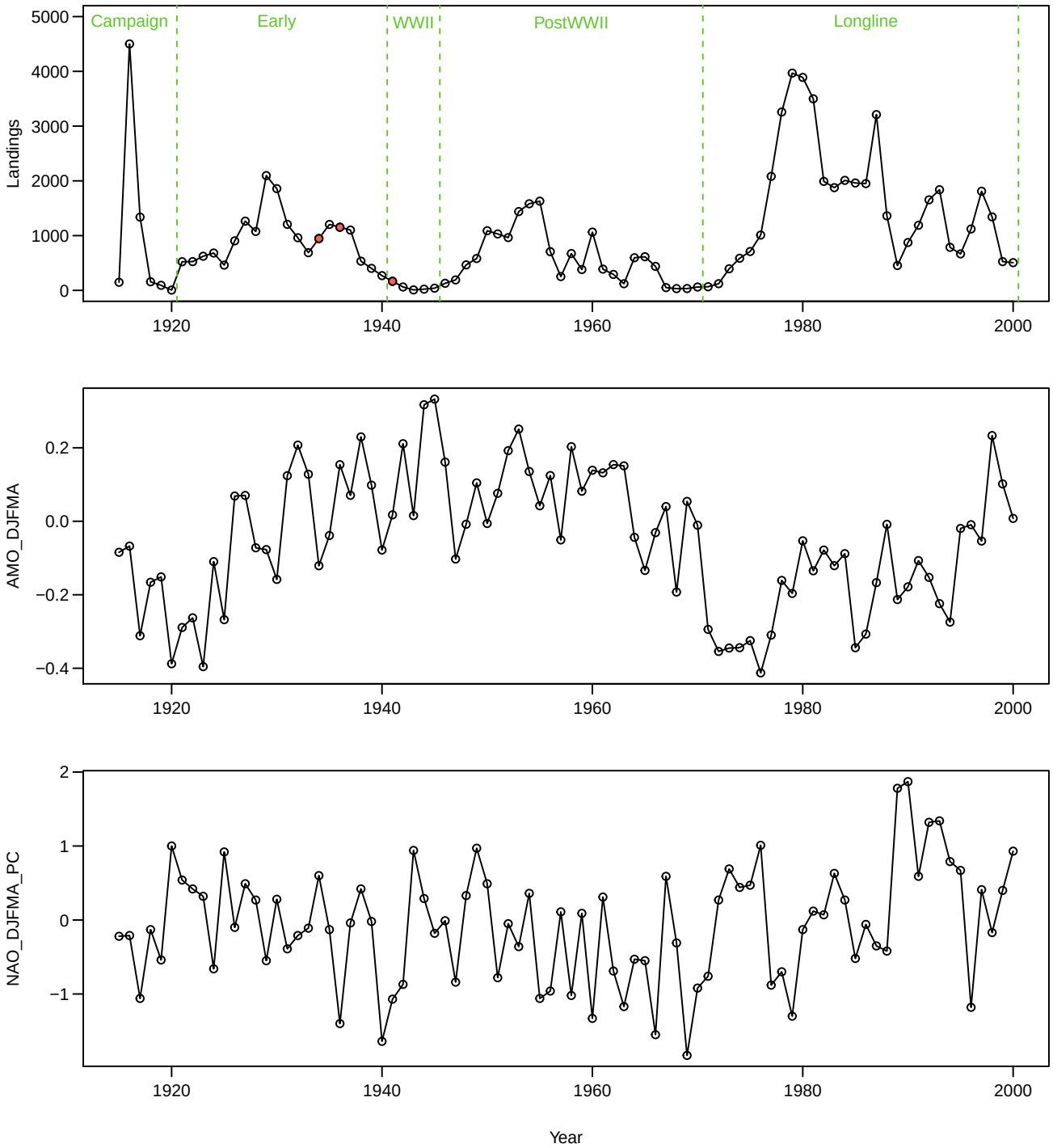


Figure 1: Plots of the main time series, 1915–2000. Filled circles represent filled-in (interpolated) values. Dashed vertical lines denote the time blocks.

```r
set.seed(333)
wavk.test(DO$NAO_DJF_PC ~ 1,
          factor.length = "adaptive.selection", ar.method = "yw")

##
##  Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
## data:  DO$NAO_DJF_PC
## WAVK test statistic = 3, adaptively selected window = 4, p-value = 0.04
## alternative hypothesis: trend is not of the form DO$NAO_DJF_PC ~ 1.

wavk.test(DO$NAO_DJF_st ~ 1,
          factor.length = "adaptive.selection", ar.method = "yw")

##
##  Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
## data:  DO$NAO_DJF_st
## WAVK test statistic = 3, adaptively selected window = 4, p-value = 0.02
## alternative hypothesis: trend is not of the form DO$NAO_DJF_st ~ 1.

wavk.test(DO$NAO_DJFMA_PC ~ 1,
          factor.length = "adaptive.selection", ar.method = "yw")

##
##  Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
## data:  DO$NAO_DJFMA_PC
## WAVK test statistic = 3, adaptively selected window = 4, p-value = 0.02
## alternative hypothesis: trend is not of the form DO$NAO_DJFMA_PC ~ 1.

wavk.test(DO$NAO_DJFMA_st ~ 1,
          factor.length = "adaptive.selection", ar.method = "yw")

##
##  Trend test by Wang, Akritas, and Van Keilegom (bootstrap p-values)
##
## data:  DO$NAO_DJFMA_st
## WAVK test statistic = 2, adaptively selected window = 4, p-value = 0.1
## alternative hypothesis: trend is not of the form DO$NAO_DJFMA_st ~ 1.
```

It is reasonable that the test rejects the H_0 for the time series of landings because there are important time blocks of the fisheries that affect the average landings; to account for that we will use the time block variable in the regression models.

See some of the scatterplots in Figure 2.

```
tmp = c(#grep("Time_block", names(D)),
  grep("l4", names(D)),
  grep("Land", names(D)))
ggpairs(D[,tmp])
```

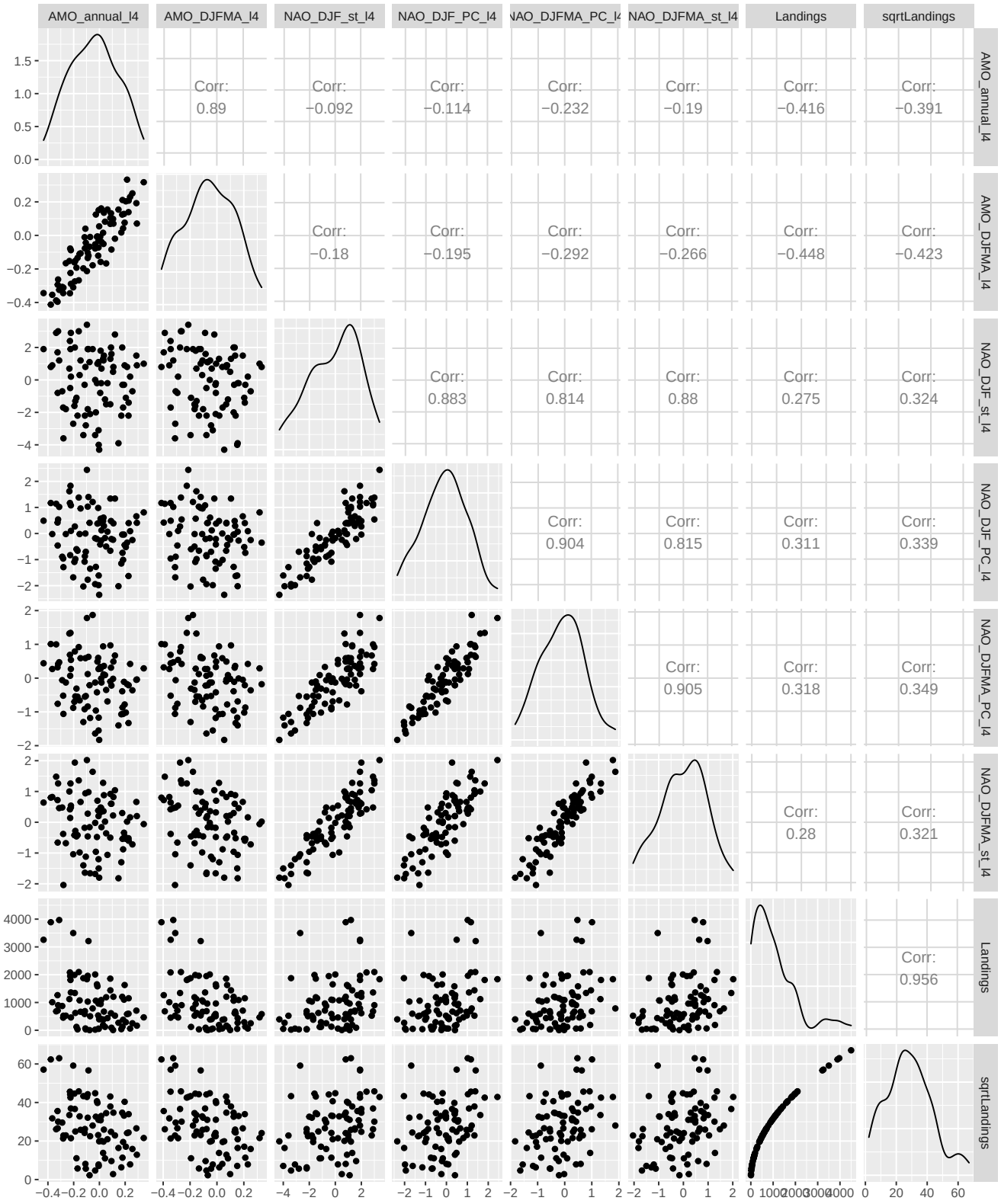


Figure 2: Matrix of scatterplots, univariate distributions (main diagonal), and pairwise correlations for the period of 1915–2000. Only some of the lagged variables are plotted for visibility.

3 Random forest

References about the methods: Breiman (2001); Hastie et al (2009); Kursa and Rudnicki (2010); Wright and Ziegler (2017).

Random forest inputs:

```
NTREE = 500
Nnode = 3
RESPONSE = "sqrtLandings"
RESPONSEprint = "sqrt(Landings)"
#Predictors:
n = names(D)
vnot = grep("Landings|Year", n, value = TRUE) #variables to exclude
predictors = sort(setdiff(n, vnot))
DATA = D
DATAnoNA = na.omit(DATA[,c(RESPONSE, predictors)])
```

Variable selection:

```
set.seed(444)
B = Boruta::Boruta(as.formula(paste(RESPONSE, ".", sep = " ~ ")),
                   doTrace = 1, maxRuns = 5000,
                   data = DATAnoNA)
print(B)
```

```
## Boruta performed 4492 iterations in 2.01 mins.
##  14 attributes confirmed important: AMO_annual_l5, AMO_annual_l6, AMO_annual_l7, AMO_DJFMA_l5,
## AMO_DJFMA_l6 and 9 more;
##  35 attributes confirmed unimportant: AMO_annual, AMO_annual_l1, AMO_annual_l2, AMO_annual_l3,
## AMO_annual_l4 and 30 more;
```

```
st = attStats(B) # print(st)
# v = rownames(st)[st$decision != "Rejected"]
v = rownames(st)[st$decision == "Confirmed"]
```

The selected variables in alphabetic order (green in Figure 3):

```
sort(v)
```

```
##  [1] "AMO_annual_l5"   "AMO_annual_l6"   "AMO_annual_l7"   "AMO_DJFMA_l5"    "AMO_DJFMA_l6"
##  [6] "AMO_DJFMA_l7"    "NAO_DJF_PC_l1"   "NAO_DJF_PC_l3"   "NAO_DJF_PC_l4"   "NAO_DJF_st_l3"
## [11] "NAO_DJF_st_l4"   "NAO_DJFMA_st_l4" "NAO_DJFMA_st_l6" "Time_block"
```

2nd step of selection – use Altmann algorithm. The predictors are the original ones (not conditional on the 1st step).

```
set.seed(2)
rf_i = ranger(dependent.variable.name = RESPONSE, data = DATAnoNA[,c(RESPONSE, predictors)],
              importance = 'impurity_corrected',
              min.node.size = Nnode, respect.unordered.factors = 'partition',
              num.trees = NTREE)
imp = importance_pvalues(rf_i, method = "altmann",
                         num.permutations = 1000,
                         formula = as.formula(paste(RESPONSE, ".", sep = " ~ ")),
                         data = DATAnoNA[,c(RESPONSE, predictors)])
imp = imp[imp[,2] < 0.05,]
v2 = sort(rownames(imp))
v = sort(intersect(v, v2))
```

The variables selected **by both** algorithms in alphabetic order:

```
v
```

```
##  [1] "AMO_annual_l5" "AMO_annual_l6" "AMO_annual_l7" "AMO_DJFMA_l5"  "AMO_DJFMA_l6"  "AMO_DJFMA_l7"
##  [7] "NAO_DJF_PC_l4" "NAO_DJF_st_l3" "NAO_DJF_st_l4" "Time_block"
```

The selected variables in order of decreasing importance (i.e., most important first):

```
w = imp[order(imp[,1], decreasing = TRUE),] #selected by Altmann sorted
w = rownames(w)
(w = w[is.element(w, v)]) #from the Altman-sorted, keep only those that appear in v
```

```
par(mar = c(4, 9, 0.1, 1) + 0.1)
plot(B, horizontal = TRUE,
     colCode = COL[c("green", "yellow", "red", "blue")],
     las = 1, ylab = "", xlab = "Importance")
```

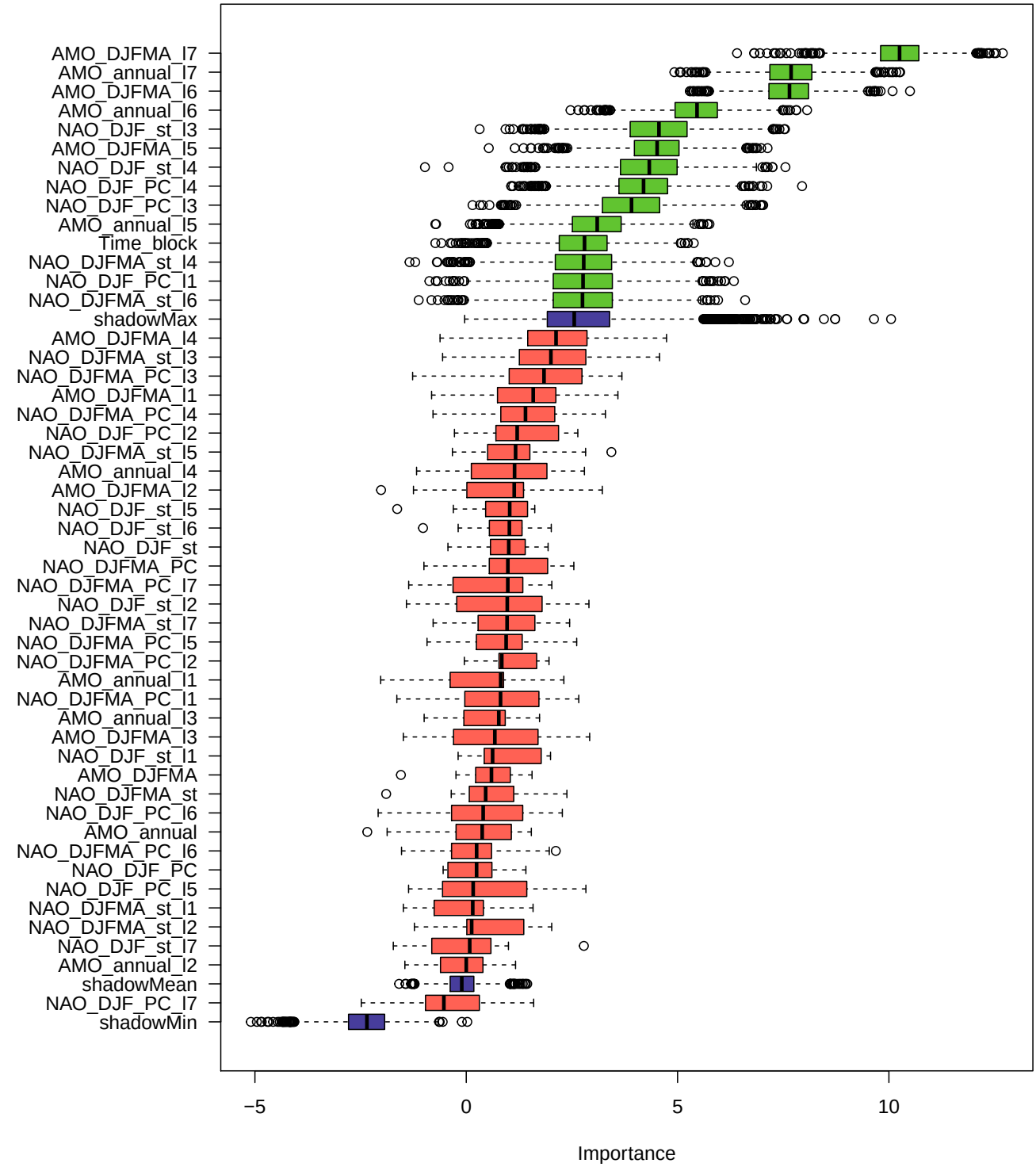


Figure 3: Boxplots of predictor importance over runs of the Boruta algorithm. Green boxes correspond to predictors confirmed as important; red boxes correspond to rejected predictors, and blue boxes combine importance information (min, mean, and max) from shadow predictors obtained by permuting the original ones. Yellow boxes (if any) correspond to predictors without decision (treated as rejected).

```
##  [1] "AMO_DJFMA_l7"  "AMO_annual_l7" "AMO_DJFMA_l6"  "AMO_annual_l6" "AMO_DJFMA_l5"  "AMO_annual_l5"
##  [7] "Time_block"    "NAO_DJF_PC_l4" "NAO_DJF_st_l3" "NAO_DJF_st_l4"
```

That is, from the original 49 variables we selected 10 variables. Get the final random forest with those 10 variables:

```
set.seed(555)
DATAnoNA = na.omit(DATA[, c(RESPONSE, v)])
rf_allv = ranger(dependent.variable.name = RESPONSE, data = DATAnoNA,
                 #importance = 'impurity_corrected',
                 min.node.size = Nnode,
                 respect.unordered.factors = 'partition',
                 num.trees = NTREE)
print(rf_allv)
```

```
## Ranger result
##
## Call:
##  ranger(dependent.variable.name = RESPONSE, data = DATAnoNA, min.node.size = Nnode,      respect.unordered.factors = "partition"
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      79
## Number of independent variables:  10
## Mtry:                             3
## Target node size:                 3
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       107
## R squared (OOB):                  0.457
```

Non-adjusted R^2:

```
# Not really useful for us
y = DATAnoNA[, RESPONSE]
e = y - predict(rf_allv, data = DATAnoNA)$predictions
1 - sum(e^2) / sum((y - mean(y))^2)
```

```
## [1] 0.918
```

Final random forest output from another package (see error plot in Figure 4 and partial dependence plots in Figure 5):

```
set.seed(666)
rf_allv2 = randomForest(y = DATAnoNA[,RESPONSE],
                        x = DATAnoNA[, v],
                        nodesize = rf_allv$min.node.size,
                        mtry = rf_allv$mtry,
                        ntree = rf_allv$num.trees)
print(rf_allv2)
```

```
##
## Call:
##  randomForest(x = DATAnoNA[, v], y = DATAnoNA[, RESPONSE], ntree = rf_allv$num.trees,      mtry = rf_allv$mtry, nodesize = rf_al
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 107
##                     % Var explained: 44.7
```

```
n = length(w) #number of single plots to plot
plotmo::plotmo(RF, pmethod = "partdep"
               ,degree1 = paste0("^", w[1:n], "$")
               ,main = c(w[1:n]), caption = "Partial dependence plots"
               ,degree2 = FALSE, type2 = "contour"
               ,las = 1)
```

```
## calculating partdep for AMO_DJFMA_l7
## calculating partdep for AMO_annual_l7
## calculating partdep for AMO_DJFMA_l6
## calculating partdep for AMO_annual_l6
```

```r
par(mar = c(4, 4, 0.1, 1) + 0.1)
plot(rf_allv2, las = 1, main = "")
```

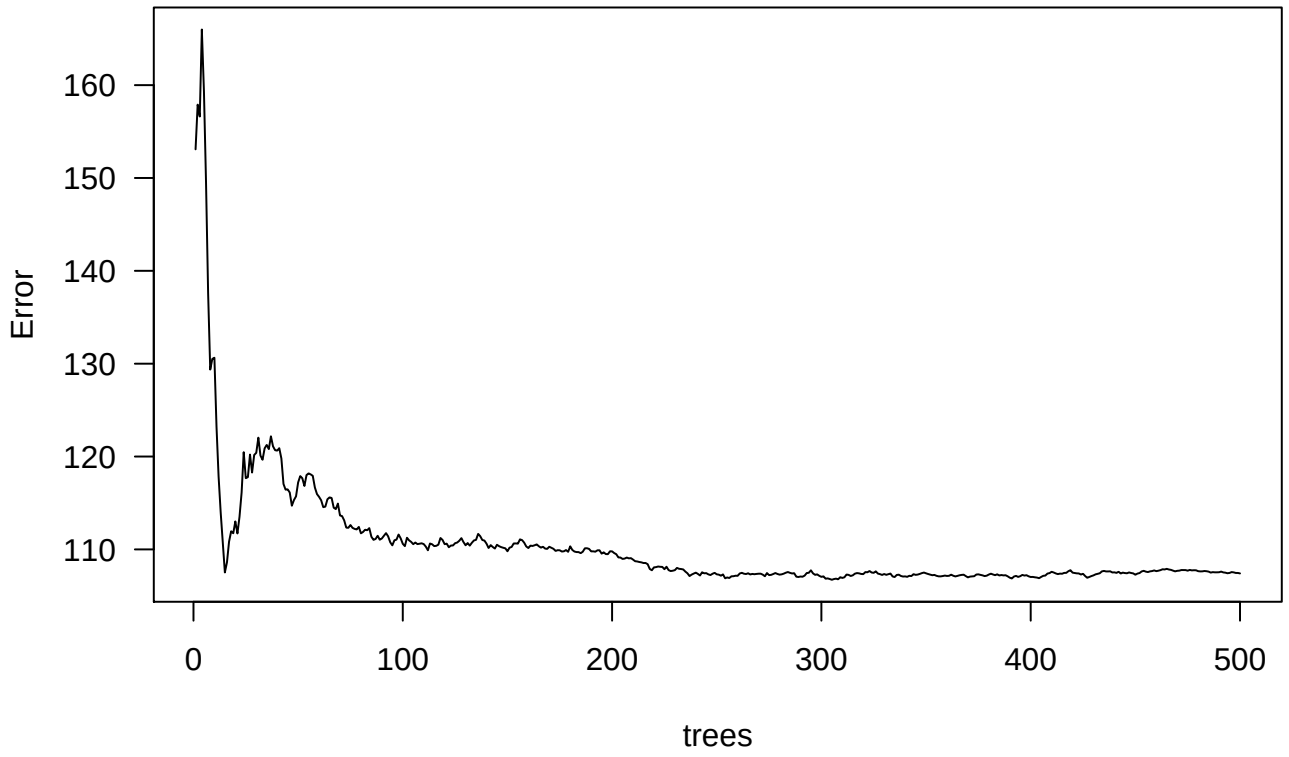


Figure 4: Random forest error based on the number of trees.

```
## calculating partdep for AMO_DJFMA_l5
## calculating partdep for AMO_annual_l5
## calculating partdep for Time_block
## calculating partdep for NAO_DJF_PC_l4
## calculating partdep for NAO_DJF_st_l3
## calculating partdep for NAO_DJF_st_l4
```

```
RF = rf_allv2
preds = sort(v)
par(mfrow = c(ceiling(length(preds)/4), 4))
par(bty = "L", mar = c(5, 4, 1, 1) + 0.1, mgp = c(2, 0.7, 0))
for(i in 1:length(preds)) {
  l = ifelse(is.factor(DATAnoNA[,preds[i]]), 2, 1) #rotate labels for categorical predictor
  x = ifelse(is.factor(DATAnoNA[,preds[i]]), "", preds[i]) #x-label
  partialPlot(RF, pred.data = DATAnoNA, x.var = preds[i],
              las = l, xlab = x, ylab = "", main = "", xpd = F
              ,ylim = c(23, 34) #fix y-scale across plots
  )
  mtext(RESPONSEprint, side = 2, line = 2, cex = 0.7)
  mtext(paste("(", letters[i], ")", sep = ""), side = 3, line = 0.1, cex = 0.8, adj = -0.37)
}
```

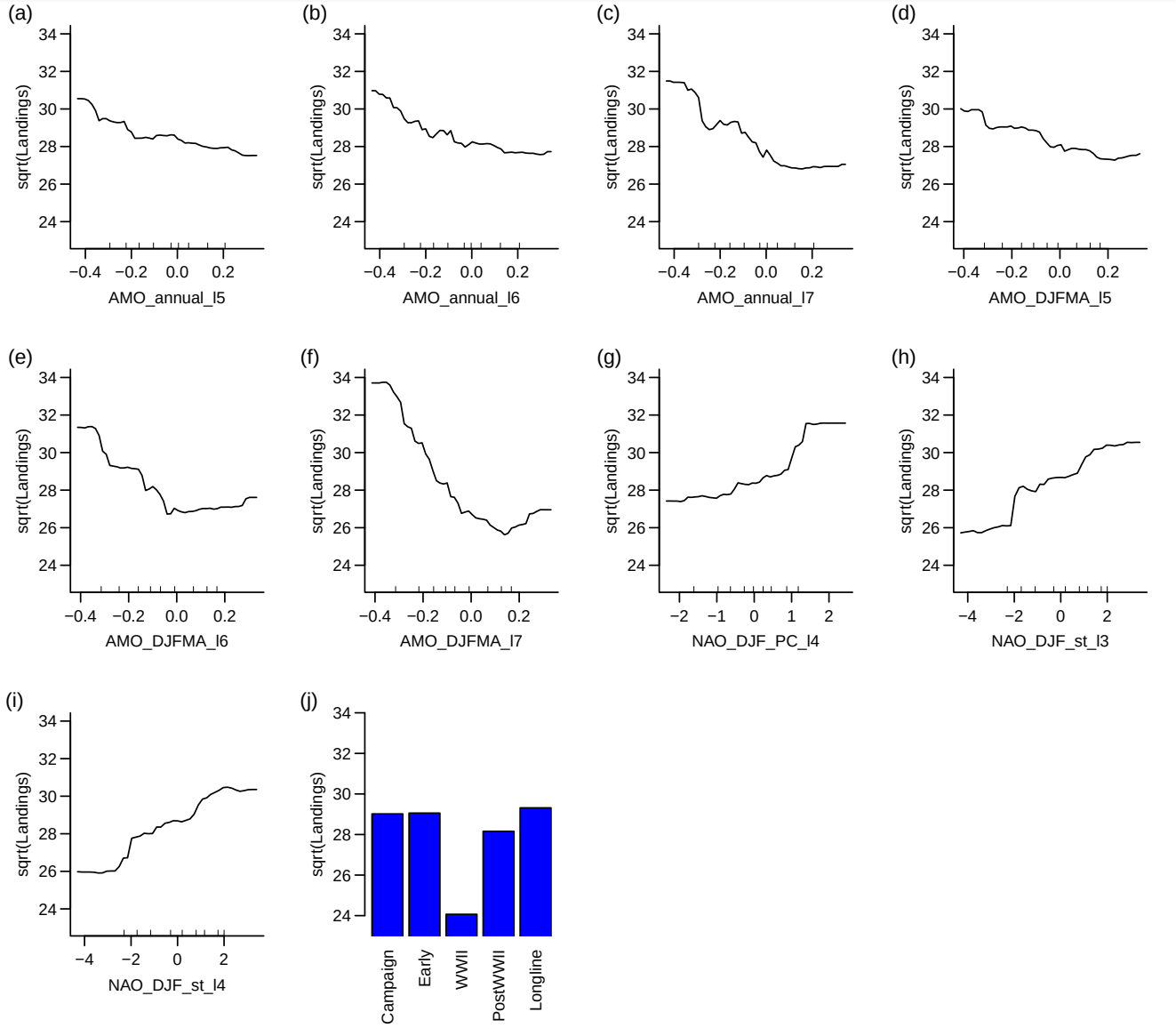
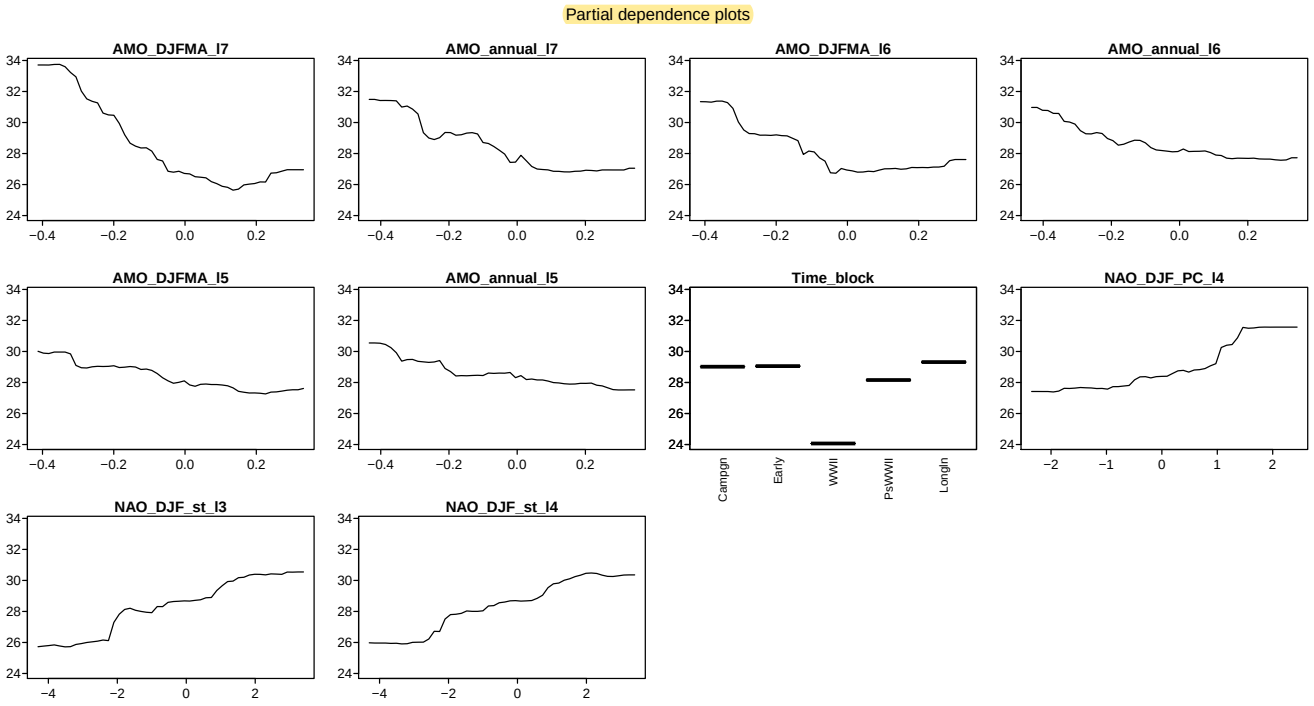
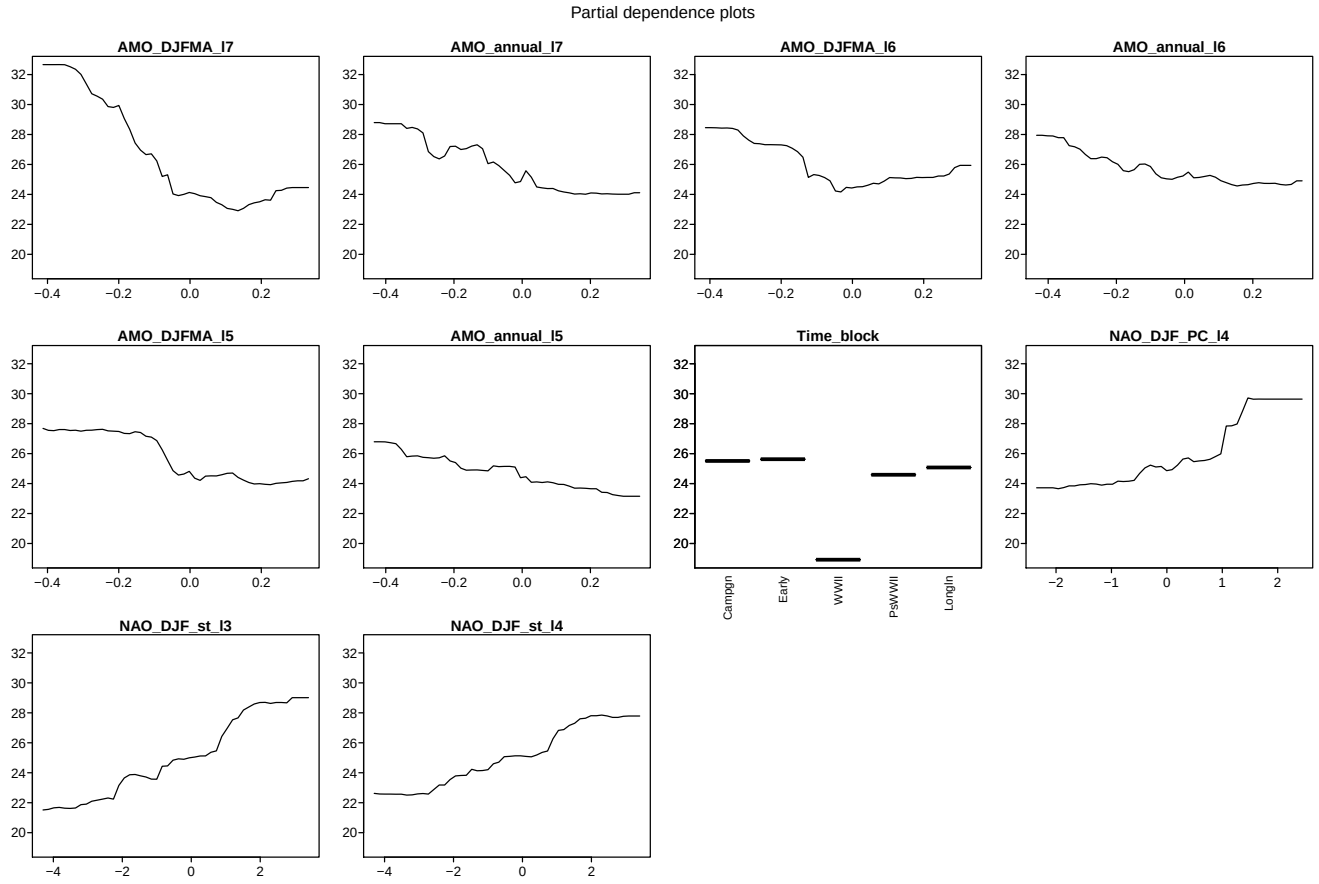


Figure 5: Partial dependence plots for the random forest for sqrt(Landings) of golden tilefish. The plots are in alphabetic order by the name of the explanatory variable. The inner tickmarks on the horizontal axis denote deciles of the respective explanatory variable.




```
plotmo::plotmo(RF, pmethod = "plotmo"
               ,degree1 = paste0("^", w[1:n], "$")
               ,main = c(w[1:n]), caption = "Partial dependence plots"
               ,degree2 = FALSE, type2 = "contour"
               ,las = 1)

##  plotmo grid:    AMO_annual_l5 AMO_annual_l6 AMO_annual_l7 AMO_DJFMA_l5 AMO_DJFMA_l6 AMO_DJFMA_l7
##                       -0.0277       -0.0328       -0.0277      -0.0532      -0.0676      -0.0676
##  NAO_DJF_PC_l4 NAO_DJF_st_l3 NAO_DJF_st_l4 Time_block
##          -0.02           0.2           0.2   Longline
```



3.1 Cross-validation

Scheme:

1. Train model using data up to the middle of the `PostWWII` period.
2. Forecast until the end of that period.
3. Repeat the above steps for the period `Longline`.

```
set.seed(777)
CVperiods = c("PostWWII", "Longline")
PRED = as.list(rep(NA, length(CVperiods)))
yrs = as.numeric(rownames(DATAnoNA))
for (cv in seq_along(CVperiods)){ # cv=1
  indtrain = yrs[1]:TBcenter[CVperiods[cv]]
  indtest = (TBcenter[CVperiods[cv]] + 1):TBend[CVperiods[cv]]
  rf_cv = ranger(dependent.variable.name = RESPONSE,
                 data = DATAnoNA[as.character(indtrain),],
                 #importance = 'impurity_corrected',
```

```
                min.node.size = Nnode,
                respect.unordered.factors = 'partition',
                num.trees = NTREE)
  obs =  DATAnoNA[as.character(indtest), RESPONSE] #observed in the testing set
  pred = predict(rf_cv, data = DATAnoNA[as.character(indtest),])$predictions #predictions
  PRED[[cv]] = list(obs = obs, pred = pred)
}
```

Count number of test samples per cross-validation run

```
(tmp = sapply(PRED, function(x) length(x[[1]])))
```

```
## [1] 12 14
```

and total

```
sum(tmp)
```

```
## [1] 26
```

Prediction mean absolute error (PMAE) and prediction root mean square error (PRMSE).

```
err = lapply(PRED, function(x) x$obs - x$pred)
err = unlist(err)
# PMAE
mean(abs(err))
```

```
## [1] 8.18
```

```
# PRMSE
sqrt(mean(err^2))
```

```
## [1] 10.5
```

4 Generalized additive modeling (GAM)

References about the methods: Wood (2006); Zuur et al (2009).

```
getOption("mgcv.vc.logrange") #25 #see ?mgcv::gamm about convergence
```

```
## [1] 25
```

```
options(mgcv.vc.logrange = 15)
```

4.1 Backward selection

Full model:

```
K = 5
gam_0 = gamfit = mgcv::gam(sqrtLandings ~
                             Time_block
                           + s(AMO_annual_l5, k = K)
                           + s(AMO_annual_l6, k = K)
                           + s(AMO_annual_l7, k = K)
                           + s(AMO_DJFMA_l5, k = K)
                           + s(AMO_DJFMA_l6, k = K)
                           + s(AMO_DJFMA_l7, k = K)
                           + s(NAO_DJF_PC_l4, k = K)
                           + s(NAO_DJF_st_l3, k = K)
                           + s(NAO_DJF_st_l4, k = K)
                           , select = TRUE
                           , bs = "cr"
                           , method = "REML"
                           , data = DATAnoNA)
summary(gamfit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## sqrtLandings ~ Time_block + s(AMO_annual_l5, k = K) + s(AMO_annual_l6,
##     k = K) + s(AMO_annual_l7, k = K) + s(AMO_DJFMA_l5, k = K) +
##     s(AMO_DJFMA_l6, k = K) + s(AMO_DJFMA_l7, k = K) + s(NAO_DJF_PC_l4,
##     k = K) + s(NAO_DJF_st_l3, k = K) + s(NAO_DJF_st_l4, k = K)
##
## Parametric coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)            27.47       2.05   13.41   <2e-16 ***
## Time_blockWWII        -14.71       4.69   -3.14   0.0025 **
## Time_blockPostWWII     -0.85       3.31   -0.26   0.7978
## Time_blockLongline      6.01       2.65    2.27   0.0263 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                        edf Ref.df    F p-value
## s(AMO_annual_l5) 8.93e-05      4 0.00  0.5173
## s(AMO_annual_l6) 9.03e-05      4 0.00  0.4990
## s(AMO_annual_l7) 2.83e-05      4 0.00  0.4773
## s(AMO_DJFMA_l5)  8.21e-05      4 0.00  0.5179
## s(AMO_DJFMA_l6)  1.81e+00      4 2.05  0.0098 **
## s(AMO_DJFMA_l7)  2.10e+00      4 4.95 6.9e-05 ***
## s(NAO_DJF_PC_l4) 7.14e-01      4 0.30  0.1070
## s(NAO_DJF_st_l3) 8.31e-01      4 1.23  0.0156 *
## s(NAO_DJF_st_l4) 1.31e+00      4 0.91  0.0239 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.63   Deviance explained = 67.7%
## -REML = 280.11  Scale est. = 72.747    n = 79
```

```
concurvity(gamfit)[3,-1]
```

```
## s(AMO_annual_l5) s(AMO_annual_l6) s(AMO_annual_l7)  s(AMO_DJFMA_l5)  s(AMO_DJFMA_l6)  s(AMO_DJFMA_l7)
##            0.849            0.923            0.908            0.896            0.897            0.863
## s(NAO_DJF_PC_l4) s(NAO_DJF_st_l3) s(NAO_DJF_st_l4)
##            0.870            0.471            0.872
```

Spearman correlations:

```
Hmisc::rcorr(as.matrix(DATAnoNA[,c(RESPONSE, v[-length(v)])]),
             type = "spearman")[[1]]

##               sqrtLandings AMO_annual_l5 AMO_annual_l6 AMO_annual_l7 AMO_DJFMA_l5 AMO_DJFMA_l6 AMO_DJFMA_l7
## sqrtLandings         1.000        -0.440        -0.540        -0.574       -0.485       -0.558       -0.601
## AMO_annual_l5       -0.440         1.000         0.733         0.560        0.891        0.613        0.498
## AMO_annual_l6       -0.540         0.733         1.000         0.747        0.823        0.895        0.639
## AMO_annual_l7       -0.574         0.560         0.747         1.000        0.574        0.822        0.885
## AMO_DJFMA_l5        -0.485         0.891         0.823         0.574        1.000        0.682        0.560
## AMO_DJFMA_l6        -0.558         0.613         0.895         0.822        0.682        1.000        0.690
## AMO_DJFMA_l7        -0.601         0.498         0.639         0.885        0.560        0.690        1.000
## NAO_DJF_PC_l4        0.355        -0.109        -0.167        -0.230       -0.112       -0.268       -0.263
## NAO_DJF_st_l3        0.356        -0.232        -0.221        -0.166       -0.280       -0.236       -0.203
## NAO_DJF_st_l4        0.393        -0.089        -0.226        -0.243       -0.123       -0.281       -0.246
##               NAO_DJF_PC_l4 NAO_DJF_st_l3 NAO_DJF_st_l4
## sqrtLandings          0.355         0.356         0.393
## AMO_annual_l5        -0.109        -0.232        -0.089
## AMO_annual_l6        -0.167        -0.221        -0.226
## AMO_annual_l7        -0.230        -0.166        -0.243
## AMO_DJFMA_l5         -0.112        -0.280        -0.123
## AMO_DJFMA_l6         -0.268        -0.236        -0.281
## AMO_DJFMA_l7         -0.263        -0.203        -0.246
## NAO_DJF_PC_l4         1.000         0.190         0.884
## NAO_DJF_st_l3         0.190         1.000         0.223
## NAO_DJF_st_l4         0.884         0.223         1.000
```

High concurvity. Most of the concurvity is due to duplicated information (e.g., annual and winter form of a climate index). From each pair of concurve terms, select such term to remove, which is less correlated with the response (use Spearman's correlation coefficient, which allows to account for possibly non-linear monotonic relationship). Based on this approach, we favor the winter form of AMO (keep in the model), and remove the annual AMO version; favor PC-based/station-based NAO and remove station-based/PC-based NAO.

Further variable selection is done by stepwise removal of terms with the largest non-significant p-value, until all remaining terms are statistically significant (also see `?mgcv::gam.selection`).

To additionally incorporate autocorrelation into the model (to get rid of autocorrelation in the model residuals), fit a generalized additive mixed model (GAMM) with autocorrelation structure specified as AR(1) – autoregression of the order 1 – and the autoregression coefficient ϕ estimated from the data. See the autocorrelation function (ACF) plot of the residuals from the final (reduced) model in Figure 7.

Reduced model:

```
#cc removed because of concurvity
##1 after that, switched to gamm and AR1 structure and continue removal
set.seed(111111)
K = 5
gam_1 = gamfit = mgcv::gamm(sqrtLandings ~
                          ##1 Time_block
                        #cc + s(AMO_annual_l5, k = K)
                        #cc + s(AMO_annual_l6, k = K)
                        #cc + s(AMO_annual_l7, k = K)
                          ##2 + s(AMO_DJFMA_l5, k = K)
                          ##3 + s(AMO_DJFMA_l6, k = K)
                            + s(AMO_DJFMA_l7, k = K)
                        #cc + s(NAO_DJF_PC_l4, k = K)
                            + s(NAO_DJF_st_l3, k = K)
```

```
                          + s(NAO_DJF_st_l4, k = K)
                          , select = TRUE
                          , bs = "cr"
                          , method = "REML"
                          , correlation = corARMA(p = 1, q = 0)
                          , data = DATAnoNA)
```

GAM part:

```
summary(gamfit$gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## sqrtLandings ~ +s(AMO_DJFMA_l7, k = K) + s(NAO_DJF_st_l3, k = K) +
##     s(NAO_DJF_st_l4, k = K)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    28.09       4.44    6.32  1.8e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df    F p-value
## s(AMO_DJFMA_l7)  1.33   1.33 4.32  0.0212 *
## s(NAO_DJF_st_l3) 1.00   1.00 8.85  0.0039 **
## s(NAO_DJF_st_l4) 1.94   1.94 5.37  0.0046 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.278
##   Scale est. = 147.7     n = 79
```

LME part:

```
summary(gamfit$lme)
```

```
## Linear mixed-effects model fit by REML
##  Data: strip.offset(mf)
##   AIC BIC logLik
##   528 549   -255
##
## Random effects:
##  Formula: ~Xr - 1 | g
##  Structure: pdIdnot
##          Xr1  Xr2  Xr3
## StdDev: 3.25 3.25 3.25
##
##  Formula: ~Xr.0 - 1 | g.0 %in% g
##  Structure: pdIdnot
##           Xr.01   Xr.02   Xr.03
## StdDev: 0.00673 0.00673 0.00673
##
##  Formula: ~Xr.1 - 1 | g.1 %in% g.0 %in% g
##  Structure: pdIdnot
##         Xr.11 Xr.12 Xr.13 Residual
## StdDev:  6.06  6.06  6.06     12.2
##
## Correlation Structure: AR(1)
##  Formula: ~1 | g/g.0/g.1
##  Parameter estimate(s):
##   Phi
## 0.846
## Fixed effects: y ~ X - 1
##                       Value Std.Error DF t-value p-value
## X(Intercept)          28.09      4.44 75    6.32  0.0000
## Xs(AMO_DJFMA_l7)Fx1   -2.60      1.16 75   -2.23  0.0287
## Xs(NAO_DJF_st_l3)Fx1   2.02      0.68 75    2.98  0.0039
## Xs(NAO_DJF_st_l4)Fx1   2.44      1.34 75    1.83  0.0719
```

```
##  Correlation:
##                       X(Int) X(AMO_ X(NAO_DJF__3
## Xs(AMO_DJFMA_l7)Fx1   0.010
## Xs(NAO_DJF_st_l3)Fx1  0.016 -0.047
## Xs(NAO_DJF_st_l4)Fx1  0.005 -0.028  0.277
##
## Standardized Within-Group Residuals:
##     Min      Q1     Med      Q3     Max
## -1.6346 -0.5572 -0.0173  0.6786  2.6688
##
## Number of Observations: 79
## Number of Groups:
##                   g          g.0 %in% g g.1 %in% g.0 %in% g
##                   1                   1                   1
```

Normality of residuals:

```
shapiro.test(residuals(gamfit$lme, type = "normalized"))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(gamfit$lme, type = "normalized")
## W = 1, p-value = 0.7
```

```
shapiro.test(residuals(gamfit$lme, type = "response"))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(gamfit$lme, type = "response")
## W = 1, p-value = 0.06
```

Run more checks:

```
gamfit = gamfit$gam
concurvity(gamfit)
```

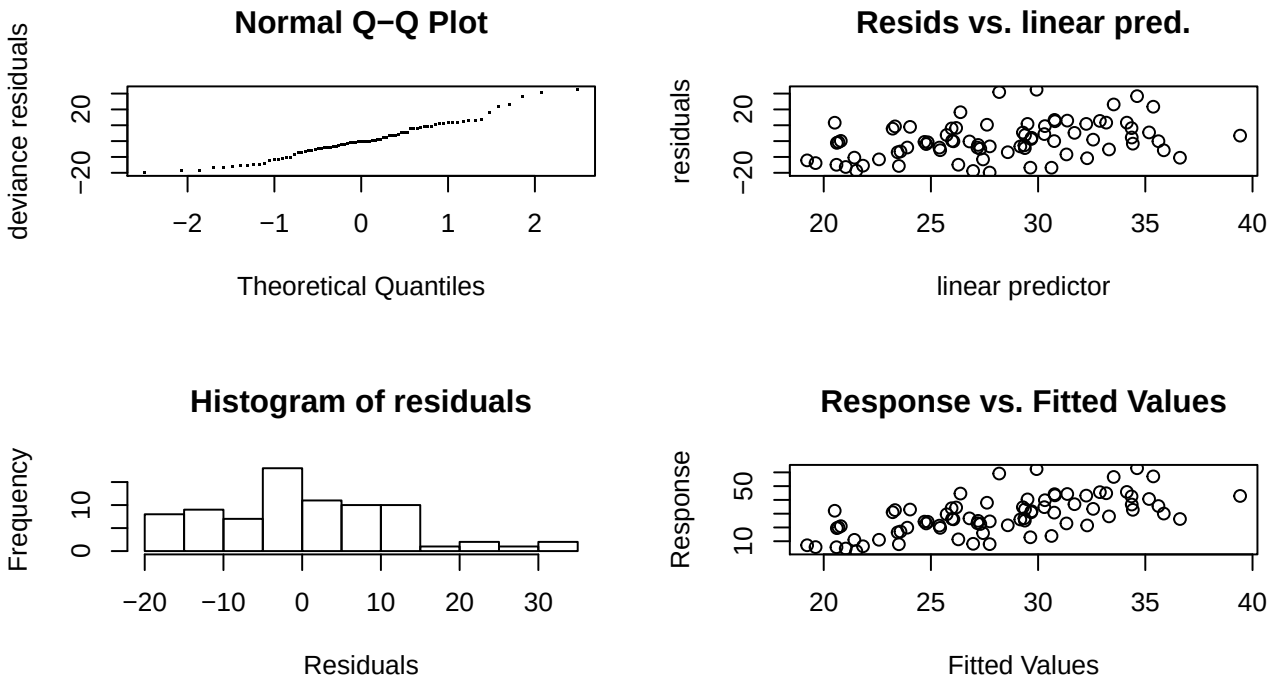
```
##              para s(AMO_DJFMA_l7) s(NAO_DJF_st_l3) s(NAO_DJF_st_l4)
## worst    6.45e-28           0.243            0.177            0.214
## observed 6.45e-28           0.116            0.132            0.160
## estimate 6.45e-28           0.131            0.128            0.142
```

```
concurvity(gamfit, full = FALSE)$estimate
```

```
##                      para s(AMO_DJFMA_l7) s(NAO_DJF_st_l3) s(NAO_DJF_st_l4)
## para             1.00e+00        2.09e-31         3.13e-30         4.45e-32
## s(AMO_DJFMA_l7)  1.34e-28        1.00e+00         8.01e-02         9.96e-02
## s(NAO_DJF_st_l3) 3.49e-28        4.95e-02         1.00e+00         5.68e-02
## s(NAO_DJF_st_l4) 2.64e-29        9.54e-02         5.37e-02         1.00e+00
```

```
gam.check(gamfit)
```

```
##
## 'gamm' based fit - care required with interpretation.
## Checks based on working residuals may be misleading.
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                    k'  edf k-index p-value
## s(AMO_DJFMA_l7)  4.00 1.33    0.75   0.015 *
## s(NAO_DJF_st_l3) 4.00 1.00    1.05   0.630
## s(NAO_DJF_st_l4) 4.00 1.94    1.12   0.855
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



4.2 Forward selection

Order of selection same as for CPUE (based on importance, not absolute Spearman correlation). Note: forward selection with order based on Spearman (use `x = tmp` below) gave the same resulting model as the backward selection in the previous section.

```
fcts = c("Time_block")
w2 = setdiff(w, fcts)
```

```
tmp = names(sort(abs(Hmisc::rcorr(as.matrix(DATAnoNA[,c(RESPONSE, w2)]),
                                  type = "spearman")[[1]][1,]), decreasing = TRUE))[-1]
gs_all = gamforward(y = RESPONSE,
                    x = w2, #w2 or tmp if want order by |r_spearman|
                    data = DATAnoNA)

## [1] "Added AMO_DJFMA_l7"
## [1] "Rejected AMO_annual_l7 due to non-significance and concurvity"
## [1] "Rejected AMO_DJFMA_l6 due to non-significance"
## [1] "Rejected AMO_annual_l6 due to non-significance"
## [1] "Rejected AMO_DJFMA_l5 due to non-significance"
## [1] "Rejected AMO_annual_l5 due to non-significance"
## [1] "Rejected NAO_DJF_PC_l4 due to non-significance"
## [1] "Rejected NAO_DJF_st_l3 due to non-significance"
## [1] "Added NAO_DJF_st_l4"
## [1] "Selected variables: AMO_DJFMA_l7, NAO_DJF_st_l4"

K = 5
f = as.formula(paste0(RESPONSE, " ~ s(", paste0(gs_all, collapse = ", k = K) + s("), ", k = K)"))
gam_1forward = gamfit = mgcv::gamm(f
                            , select = TRUE
                            , bs = "cr"
                            , method = "REML"
                            , correlation = corARMA(p = 1, q = 0)
                            , data = DATAnoNA)
# summary(gamfit$gam)
# summary(gamfit$lme)
# concurvity(gamfit$gam)[3,-1]
# acf(residuals(gamfit))
# acf(residuals(gamfit$lme, type = "normalized"), las = 1, main = "Standardized residual ACF")
```

GAM part:

```
summary(gamfit$gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## sqrtLandings ~ s(AMO_DJFMA_l7, k = K) + s(NAO_DJF_st_l4, k = K)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     27.8        4.8    5.79  1.5e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df    F p-value
## s(AMO_DJFMA_l7)  1.00   1.00 5.13   0.026 *
## s(NAO_DJF_st_l4) 1.74   1.74 2.14   0.078 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.195
##   Scale est. = 168.49    n = 79
```

LME part:

```
summary(gamfit$lme)
```

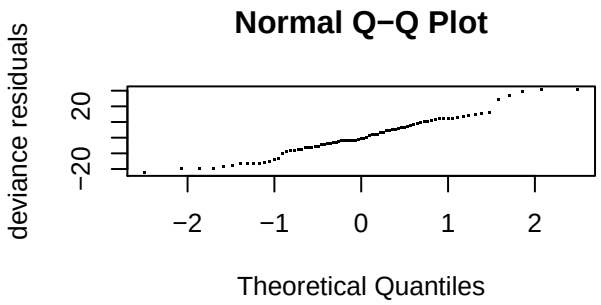
```
## Linear mixed-effects model fit by REML
##  Data: strip.offset(mf)
##   AIC BIC logLik
##   533 549   -260
##
## Random effects:
##  Formula: ~Xr - 1 | g
##  Structure: pdIdnot
##             Xr1     Xr2     Xr3
## StdDev: 0.00724 0.00724 0.00724
##
##  Formula: ~Xr.0 - 1 | g.0 %in% g
##  Structure: pdIdnot
##         Xr.01 Xr.02 Xr.03 Residual
## StdDev:     5     5     5       13
##
## Correlation Structure: AR(1)
##  Formula: ~1 | g/g.0
##  Parameter estimate(s):
##  Phi
## 0.85
## Fixed effects: y ~ X - 1
##                      Value Std.Error DF t-value p-value
## X(Intercept)         27.84      4.80 76    5.79  0.0000
## Xs(AMO_DJFMA_l7)Fx1  -2.34      1.03 76   -2.27  0.0263
## Xs(NAO_DJF_st_l4)Fx1  1.32      1.17 76    1.12  0.2642
##  Correlation:
##                      X(Int) X(AMO_
## Xs(AMO_DJFMA_l7)Fx1  0.015
## Xs(NAO_DJF_st_l4)Fx1 0.004  0.009
##
## Standardized Within-Group Residuals:
##     Min      Q1     Med      Q3     Max
## -1.7164 -0.5341 -0.0454  0.7352  2.3581
##
## Number of Observations: 79
## Number of Groups:
##          g g.0 %in% g
##          1          1
```

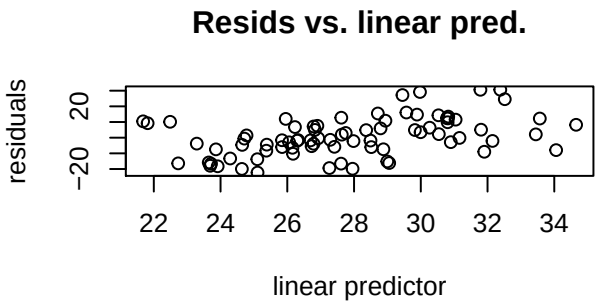
Normality of residuals:

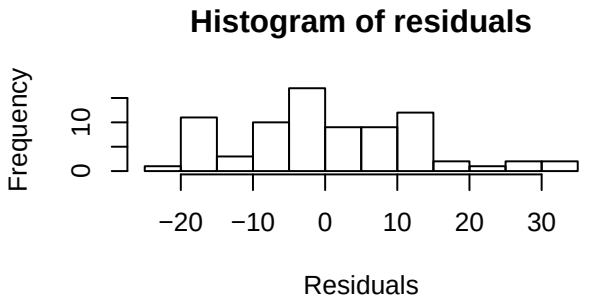
```
shapiro.test(residuals(gamfit$lme, type = "normalized"))

##
##  Shapiro-Wilk normality test
##
## data:  residuals(gamfit$lme, type = "normalized")
## W = 1, p-value = 0.5

shapiro.test(residuals(gamfit$lme, type = "response"))

##
##  Shapiro-Wilk normality test
##
## data:  residuals(gamfit$lme, type = "response")
## W = 1, p-value = 0.1
```

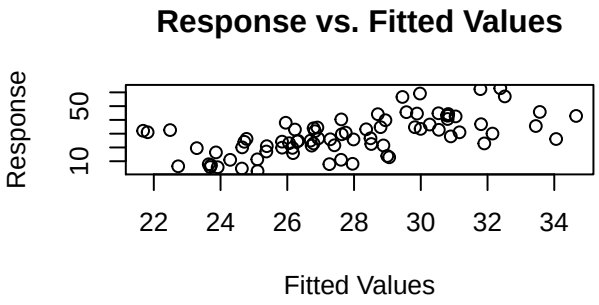
Run more checks:

```
gamfit = gamfit$gam
concurvity(gamfit)

##              para s(AMO_DJFMA_l7) s(NAO_DJF_st_l4)
## worst    1.65e-28          0.2045           0.2045
## observed 1.65e-28          0.0904           0.1225
## estimate 1.65e-28          0.0954           0.0996

concurvity(gamfit, full = FALSE)$estimate

##                      para s(AMO_DJFMA_l7) s(NAO_DJF_st_l4)
## para             1.00e+00        2.09e-31         4.45e-32
## s(AMO_DJFMA_l7)  1.34e-28        1.00e+00         9.96e-02
## s(NAO_DJF_st_l4) 2.09e-29        9.54e-02         1.00e+00

gam.check(gamfit)

##
## 'gamm' based fit - care required with interpretation.
## Checks based on working residuals may be misleading.
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                     k'  edf k-index p-value
## s(AMO_DJFMA_l7)  4.00 1.00    0.67   0.005 **
## s(NAO_DJF_st_l4) 4.00 1.74    1.07   0.730
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```


Normal Q−Q Plot
deviance residuals
−20
20
−2
−1
0
1
2
Theoretical Quantiles


Resids vs. linear pred.
residuals
−20
20
22
24
26
28
30
32
34
linear predictor


Histogram of residuals
Frequency
0
10
−20
−10
0
10
20
30
Residuals


Response vs. Fitted Values
Response
10
50
22
24
26
28
30
32
34
Fitted Values

```
ggpairs(DATAnoNA[,c(v, RESPONSE)])
```

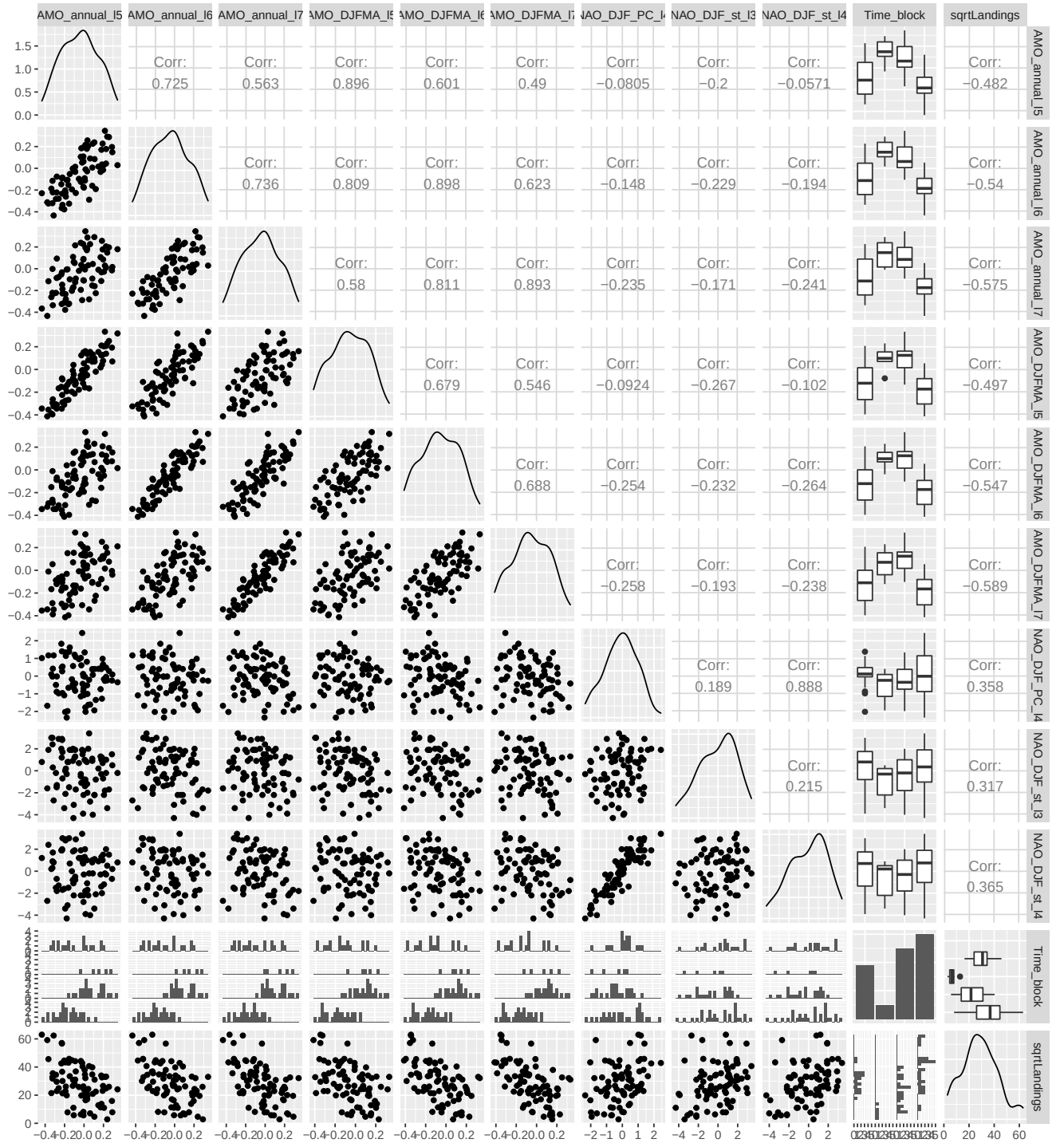


Figure 6: Scatterplot matrix of sqrt(Landings) and variables selected in the random forest.

```
par(mfrow = c(1, 2))
plot.ts(residuals(gamfit$lme, type = "normalized"), las = 1)
abline(h = 0, col = COL["green"], lty = 2)
acf(residuals(gamfit$lme, type = "normalized"), las = 1, main = "Standardized residual ACF")
```

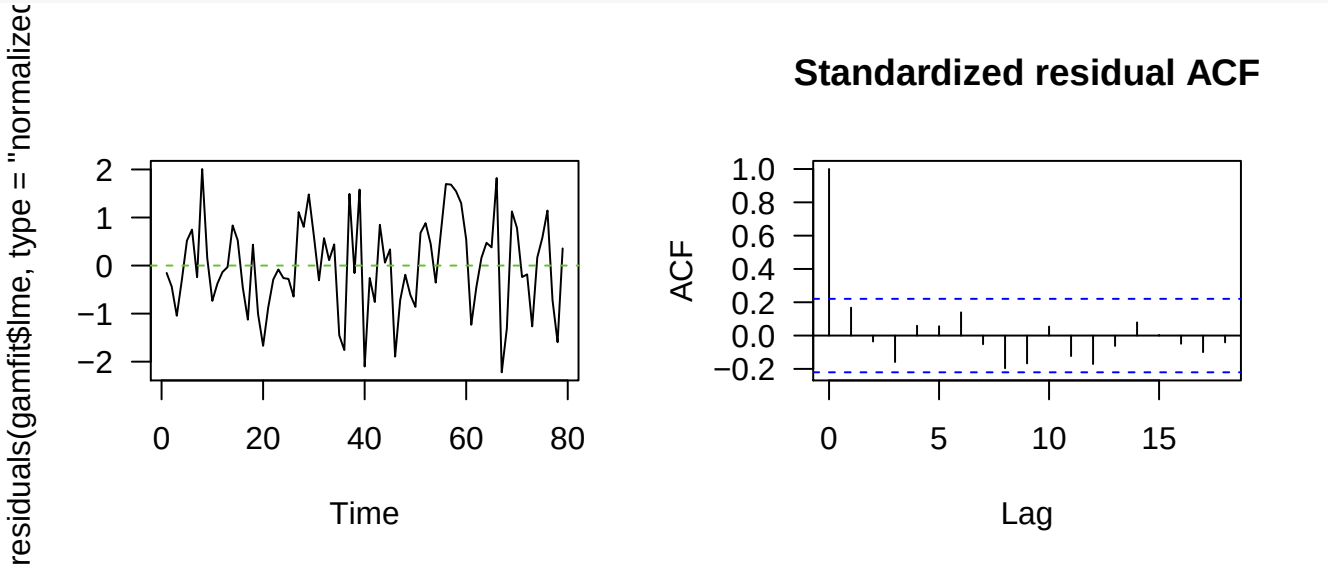


Figure 7: Time series plot and autocorrelation function of residuals of the reduced GAMM.

```
par(mfrow = c(1, 3))
plot(gamfit, las = 1)
```

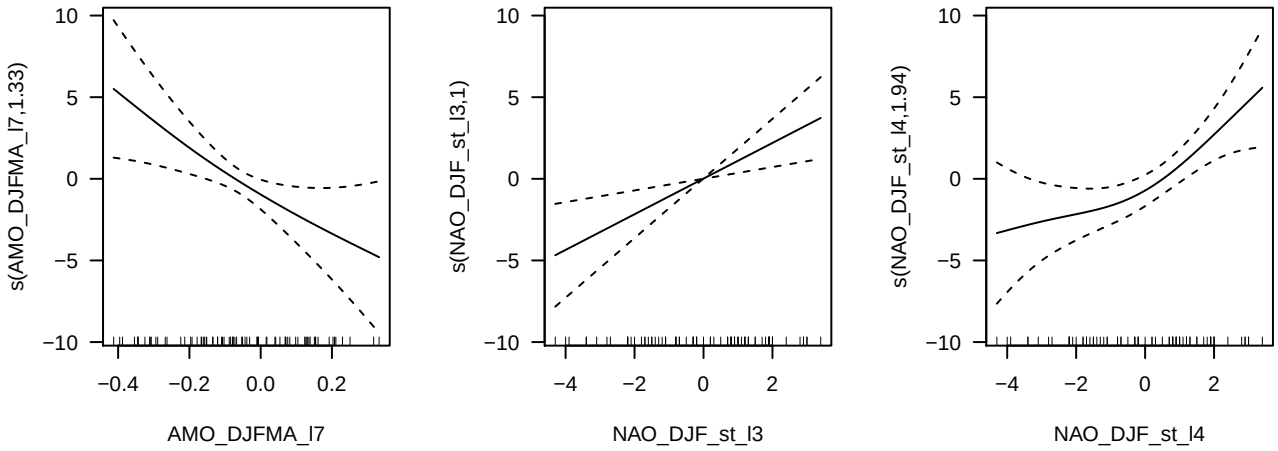


Figure 8: Smoothed terms of the final GAMM for the sqrt(Landings).

```
par(mfrow = c(1, 2))
plot.ts(residuals(gamfit$lme, type = "normalized"), las = 1)
abline(h = 0, col = COL["green"], lty = 2)
acf(residuals(gamfit$lme, type = "normalized"), las = 1, main = "Standardized residual ACF")
```

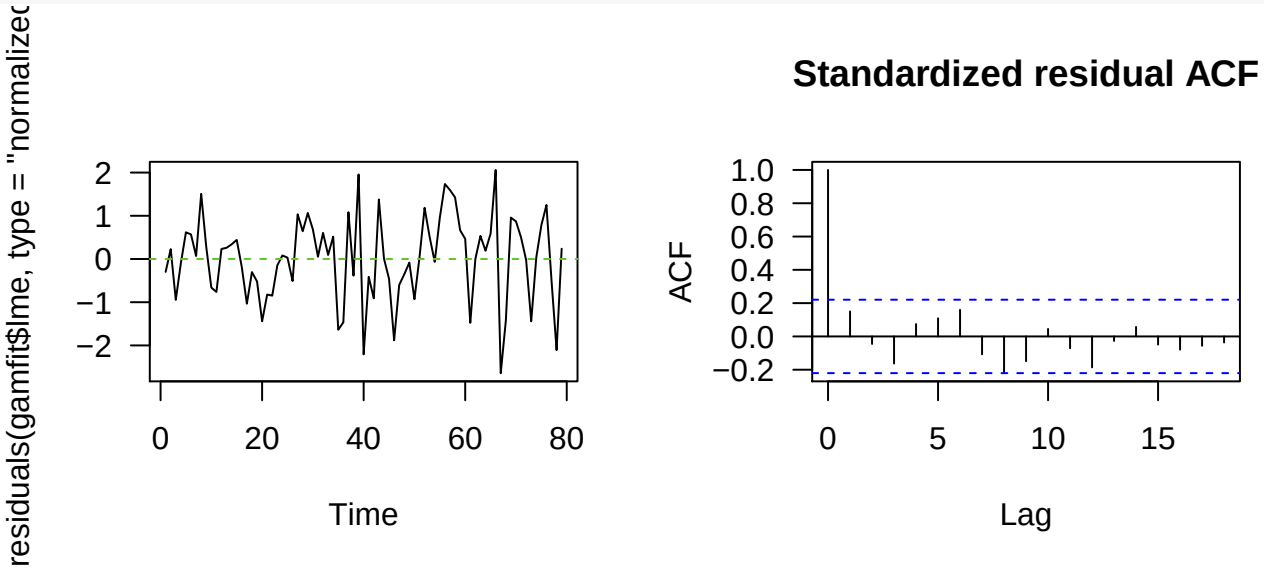


Figure 9: Time series plot and autocorrelation function of residuals of the reduced GAMM (forward selection).

```
par(mfrow = c(1, 3))
plot(gamfit, las = 1)
```

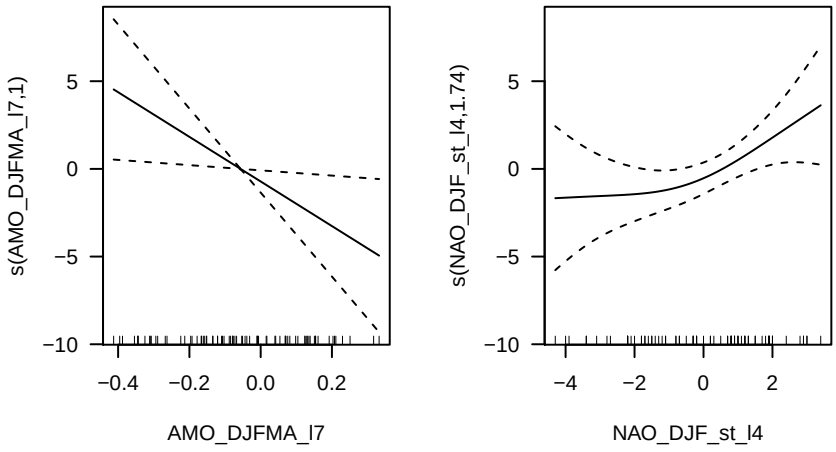


Figure 10: Smoothed terms of the final GAMM for the sqrt(Landings).

4.3 Cross-validation

The scheme is the same as the one applied to random forests (see p. 15).

```
set.seed(222222)
CVperiods = c("PostWWII", "Longline")
PRED = as.list(rep(NA, length(CVperiods)))
yrs = as.numeric(rownames(DATAnoNA))
for (cv in seq_along(CVperiods)){ # cv=1
  indtrain = yrs[1]:TBcenter[CVperiods[cv]]
  indtest = (TBcenter[CVperiods[cv]] + 1):TBend[CVperiods[cv]]
  gam_cv = mgcv::gamm(sqrtLandings ~
                        + s(AMO_DJFMA_l7, k = K)
                      + s(NAO_DJF_st_l3, k = K)
                      + s(NAO_DJF_st_l4, k = K)
                      , select = TRUE
                      , bs = "cr" #cr/cs/cc
                      , method = "REML"
                      , correlation = corARMA(p = 1, q = 0)
                      , control = list(maxIter = 1000)
                      , data = DATAnoNA[as.character(indtrain),])
  obs =  DATAnoNA[as.character(indtest), RESPONSE] #observed in the testing set
  # Predictions:
  ## make a prediction, with random effects zero
  pred_gam = predict(gam_cv$gam, newdata = DATAnoNA[as.character(indtest),])
  phi = gam_cv$lme$modelStruct$corStruct
  phi = coef(phi, unconstrained = FALSE) #AR(1) coefficient
  ## extract autocorrelated errors in the training period (we'll need the last observed error)
  fit_gam = fitted(gam_cv$lme)
  e_train = DATAnoNA[as.character(indtrain), RESPONSE] - fit_gam
  #acf(e_train) #ar(e_train, aic = FALSE, order.max = 1) #check that is similar to phi
  e = obs - pred_gam #autocorrelated error in the testing set
  pred_gamm = pred_gam + phi * c(e_train[length(e_train)], e[-length(e)])
  PRED[[cv]] = list(obs = obs, pred_gamm = pred_gamm, pred_gam = pred_gam)
}
```

Prediction mean absolute error (PMAE) and prediction root mean square error (PRMSE). Note, `err_gam` correspond to errors when we do not know the previous month landings, so it is more like h-step ahead prediction errors, rather than `err_gamm` that use the observed error from the previous month, that is more like 1-step ahead predictions.

```
err_gam = lapply(PRED, function(x) x$obs - x$pred_gam)
err_gam = unlist(err_gam)
err_gamm = lapply(PRED, function(x) x$obs - x$pred_gamm)
err_gamm = unlist(err_gamm)
# PMAE
mean(abs(err_gam))

## [1] 7.82

mean(abs(err_gamm))

## [1] 6.68

# PRMSE
sqrt(mean(err_gam^2))

## [1] 9.85

sqrt(mean(err_gamm^2))

## [1] 8.36
```

4.4 Cross-validation of forward-selected model

The scheme is the same as the one applied to random forests (see p. 15).

```
set.seed(222222)
CVperiods = c("PostWWII", "Longline")
PRED = as.list(rep(NA, length(CVperiods)))
```

```
yrs = as.numeric(rownames(DATAnoNA))
for (cv in seq_along(CVperiods)){ # cv=1
  indtrain = yrs[1]:TBcenter[CVperiods[cv]]
  indtest = (TBcenter[CVperiods[cv]] + 1):TBend[CVperiods[cv]]
  gam_cv = mgcv::gamm(sqrtLandings ~
                        + s(AMO_DJFMA_l7, k = K)
                      # + s(NAO_DJF_st_l3, k = K)
                      + s(NAO_DJF_st_l4, k = K)
                      , select = TRUE
                      , bs = "cr" #cr/cs/cc
                      , method = "REML"
                      , correlation = corARMA(p = 1, q = 0)
                      , control = list(maxIter = 1000)
                      , data = DATAnoNA[as.character(indtrain),])
  obs =  DATAnoNA[as.character(indtest), RESPONSE] #observed in the testing set
  # Predictions:
  ## make a prediction, with random effects zero
  pred_gam = predict(gam_cv$gam, newdata = DATAnoNA[as.character(indtest),])
  phi = gam_cv$lme$modelStruct$corStruct
  phi = coef(phi, unconstrained = FALSE) #AR(1) coefficient
  ## extract autocorrelated errors in the training period (we'll need the last observed error)
  fit_gam = fitted(gam_cv$lme)
  e_train = DATAnoNA[as.character(indtrain), RESPONSE] - fit_gam
  #acf(e_train) #ar(e_train, aic = FALSE, order.max = 1) #check that is similar to phi
  e = obs - pred_gam #autocorrelated error in the testing set
  pred_gamm = pred_gam + phi * c(e_train[length(e_train)], e[-length(e)])
  PRED[[cv]] = list(obs = obs, pred_gamm = pred_gamm, pred_gam = pred_gam)
}
```

Prediction mean absolute error (PMAE) and prediction root mean square error (PRMSE). Note, `err_gam` correspond to errors when we do not know the previous month landings, so it is more like h-step ahead prediction errors, rather than `err_gamm` that use the observed error from the previous month, that is more like 1-step ahead predictions.

```
err_gam = lapply(PRED, function(x) x$obs - x$pred_gam)
err_gam = unlist(err_gam)
err_gamm = lapply(PRED, function(x) x$obs - x$pred_gamm)
err_gamm = unlist(err_gamm)
# PMAE
mean(abs(err_gam))
```

```
## [1] 7.97
```

```
mean(abs(err_gamm))
```

```
## [1] 6.79
```

```
# PRMSE
sqrt(mean(err_gam^2))
```

```
## [1] 10
```

```
sqrt(mean(err_gamm^2))
```

```
## [1] 8.68
```

```
save.image(paste0("./dataderived/tmp_", Sys.Date(), ".RData"))
```

References

Breiman L (2001) Random forests. Machine Learning 45(1):5–32, DOI 10.1023/A:1010933404324

Fisher JAD, Frank KT, Petrie B, Leggett WC (2014) Life on the edge: environmental determinants of tilefish (*Lopholatilus chamaeleonticeps*) abundance since its virtual extinction in 1882. ICES Journal of Marine Science 71(9):2371–2378, DOI 10.1093/icesjms/fsu053

Hastie TJ, Tibshirani RJ, Friedman JH (2009) The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer, New York, DOI 10.1007/978-0-387-84858-7

Kursa MB, Rudnicki WR (2010) Feature selection with the Boruta package. Journal of Statistical Software 36(11):1–13

Lyubchich V, Gel YR, El-Shaarawi A (2013) On detecting non-monotonic trends in environmental time series: a fusion of local regression and bootstrap. Environmetrics 24(4):209–226, DOI 10.1002/env.2212

Wood SN (2006) Generalized Additive Models: An Introduction with R. Chapman and Hall/CRC, New York

Wright MN, Ziegler A (2017) ranger: A fast implementation of random forests for high dimensional data in C++ and R. Journal of Statistical Software 77(1):1–17, DOI 10.18637/jss.v077.i01

Zuur A, Ieno EN, Walker NJ, Saveliev AA, Smith GM (2009) Mixed Effects Models and Extensions in Ecology with R. Springer, New York, DOI 10.1007/978-0-387-87458-6