SURE Project

September 05, 2022

Objective

Predict and project spatially the result from the models.

```
## Script by Anita Giraldo, 4 May 2022
## Last modified by Anita Giraldo, 4 May 2022

# Clear environment ----
# rm(list=ls())

# directories ----
m.dir <- here()
d.dir <- here('data')

## Load info on years RCCA ----
years <- read.csv(paste(d.dir, "RCCA_North_Coast_sites.csv", sep ='/')) # Rows: 25

# get the sites from with preMHW data ----
# 3 or more pre MHW surveys
ncsites <- years %>%
  mutate_at(vars(site_name), list(as.factor)) %>%
  # get only sites with PRE MHW data
  dplyr::filter(pre.mhw.years > 2) %>%
  droplevels() # Rows: 10
```

1. Load RCCA data

```
df <- read.csv(paste(d.dir, "RCCA_kelp_inverts_NC_depth-zones_wave_clim_temp_nit_subs_orbvel_npp.csv", s
  mutate_at(vars(site_name, month, year, transect, zone), list(as.factor)) %>%
  mutate(zone_new = case_when(
    transect == '1' ~ 'OUTER',
    transect == '2' ~ 'OUTER',
    transect == '3' ~ 'OUTER',
    transect == '4' ~ 'INNER',
    transect == '5' ~ 'INNER',
    transect == '6' ~ 'INNER')) %>%
  dplyr::select(-zone) %>%
  rename(zone = zone_new) %>%
  mutate_at(vars(zone), list(as.factor)) %>%
  relocate(zone, .after = transect) # Rows: 1,154

## get the sites for North Coast model ----
```

```r
df.nc <- df %>%
  dplyr::select(-c(latitude, longitude)) %>%
  right_join(ncsites, by = c('site_name')) %>%
  droplevels() %>% # glimpse()
  # dplyr::select(-c(total.years, pre.mhw.years, during.mhw.years, post.mhw.years)) %>%
  relocate(c(latitude, longitude), .after = zone) # Rows: 708

length(levels(df.nc$site_name)) # 10
```

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

```r
levels(df.nc$site_name)
```

```
##  [1] "Fort Ross"            "Gerstle Cove"         "Mendocino Headlands"
##  [4] "Ocean Cove"           "Point Arena MPA (M2)" "Point Arena Ref"
##  [7] "Portuguese Beach"     "Stillwater Sonoma"    "Stornetta"
## [10] "Van Damme"
```

```r
any(is.na(df.nc$Max_Monthly_Anomaly_Temp)) # FALSE
```

```
## [1] FALSE
```

2. Choose variables and transform needed

```r
names(df.nc)
```

```
##   [1] "site_name"
##   [2] "month"
##   [3] "year"
##   [4] "transect"
##   [5] "zone"
##   [6] "latitude"
##   [7] "longitude"
##   [8] "den_STRPURAD"
##   [9] "den_HALRUF"
##  [10] "den_MESFRAAD"
##  [11] "den_PYCHEL"
##  [12] "den_NERLUE"
##  [13] "den_MACPYRAD"
##  [14] "den_NERLUEsmall"
##  [15] "den_MACSTIPES"
##  [16] "npp.mean"
##  [17] "pdo_mean"
##  [18] "npgo_mean"
##  [19] "mei_mean"
##  [20] "Days_15C"
##  [21] "Days_16C"
##  [22] "Days_17C"
##  [23] "Days_18C"
```

```
## [24] "Days_19C"
## [25] "Days_20C"
## [26] "Days_21C"
## [27] "Days_22C"
## [28] "Days_23C"
## [29] "Degree_Days_15C"
## [30] "Degree_Days_16C"
## [31] "Degree_Days_17C"
## [32] "Degree_Days_18C"
## [33] "Degree_Days_19C"
## [34] "Degree_Days_20C"
## [35] "Degree_Days_21C"
## [36] "Degree_Days_22C"
## [37] "Degree_Days_23C"
## [38] "Max_Monthly_Anomaly_Summer_Temp"
## [39] "Max_Monthly_Anomaly_Temp"
## [40] "Max_Monthly_Anomaly_Upwelling_Temp"
## [41] "Max_Monthly_Temp_Index"
## [42] "Max_Monthly_Temp"
## [43] "Mean_Monthly_Summer_Temp"
## [44] "Mean_Monthly_Temp"
## [45] "Mean_Monthly_Upwelling_Temp"
## [46] "MHW_Days"
## [47] "MHW_Intensity"
## [48] "MHW_Summer_Days"
## [49] "MHW_Summer_Intensity"
## [50] "MHW_Upwelling_Days"
## [51] "MHW_Upwelling_Intensity"
## [52] "Min_Monthly_Anomaly_Summer_Temp"
## [53] "Min_Monthly_Anomaly_Temp"
## [54] "Min_Monthly_Anomaly_Upwelling_Temp"
## [55] "Min_Monthly_Temp_Index"
## [56] "Min_Monthly_Temp"
## [57] "Days_10N"
## [58] "Days_11N"
## [59] "Days_12N"
## [60] "Days_13N"
## [61] "Days_14N"
## [62] "Days_15N"
## [63] "Days_1N"
## [64] "Days_2N"
## [65] "Days_3N"
## [66] "Days_4N"
## [67] "Days_5N"
## [68] "Days_6N"
## [69] "Days_7N"
## [70] "Days_8N"
## [71] "Days_9N"
## [72] "Degree_Days_10N"
## [73] "Degree_Days_11N"
## [74] "Degree_Days_12N"
## [75] "Degree_Days_13N"
## [76] "Degree_Days_14N"
## [77] "Degree_Days_15N"
```

```
##  [78] "Degree_Days_1N"
##  [79] "Degree_Days_2N"
##  [80] "Degree_Days_3N"
##  [81] "Degree_Days_4N"
##  [82] "Degree_Days_5N"
##  [83] "Degree_Days_6N"
##  [84] "Degree_Days_7N"
##  [85] "Degree_Days_8N"
##  [86] "Degree_Days_9N"
##  [87] "Max_Monthly_Anomaly_Nitrate"
##  [88] "Max_Monthly_Anomaly_Summer_Nitrate"
##  [89] "Max_Monthly_Anomaly_Upwelling_Nitrate"
##  [90] "Max_Monthly_Nitrate_Index"
##  [91] "Max_Monthly_Nitrate"
##  [92] "Mean_Monthly_Nitrate"
##  [93] "Mean_Monthly_Summer_Nitrate"
##  [94] "Mean_Monthly_Upwelling_Nitrate"
##  [95] "Min_Monthly_Anomaly_Nitrate"
##  [96] "Min_Monthly_Anomaly_Summer_Nitrate"
##  [97] "Min_Monthly_Anomaly_Upwelling_Nitrate"
##  [98] "Min_Monthly_Nitrate"
##  [99] "Min_Monthly_Temp_Nitrate"
## [100] "mean_depth"
## [101] "mean_prob_of_rock"
## [102] "mean_vrm"
## [103] "mean_slope"
## [104] "sd_depth"
## [105] "sd_prob_of_rock"
## [106] "sd_vrm"
## [107] "sd_slope"
## [108] "min_depth"
## [109] "min_prob_of_rock"
## [110] "min_vrm"
## [111] "min_slope"
## [112] "max_depth"
## [113] "max_prob_of_rock"
## [114] "max_vrm"
## [115] "max_slope"
## [116] "range_depth"
## [117] "range_prob_of_rock"
## [118] "range_vrm"
## [119] "range_slope"
## [120] "median_depth"
## [121] "median_prob_of_rock"
## [122] "median_vrm"
## [123] "median_slope"
## [124] "prop_map_depth"
## [125] "prop_map_prob_of_rock"
## [126] "prop_map_vrm"
## [127] "prop_map_slope"
## [128] "wh_mean"
## [129] "wh_max"
## [130] "wh_95prc"
## [131] "wh_99prc"
```

```
## [132] "mean_waveyear"
## [133] "max_waveyear"
## [134] "wh_95prc_wy"
## [135] "wh_99prc_wy"
## [136] "UBR_Max"
## [137] "UBR_Mean"
## [138] "UBRYear_Max"
## [139] "UBRYear_Mean"
## [140] "Wave_Max"
## [141] "Wave_Mean"
## [142] "WaveYear_Max"
## [143] "WaveYear_Mean"
## [144] "Max_Monthly_NPP"
## [145] "Max_Monthly_NPP_Summer"
## [146] "Max_Monthly_NPP_Upwelling"
## [147] "Mean_Monthly_NPP"
## [148] "Mean_Monthly_NPP_Summer"
## [149] "Mean_Monthly_NPP_Upwelling"
## [150] "Min_Monthly_NPP"
## [151] "Min_Monthly_NPP_Summer"
## [152] "Min_Monthly_NPP_Upwelling"
## [153] "total.years"
## [154] "pre.mhw.years"
## [155] "during.mhw.years"
## [156] "post.mhw.years"
```

```
dat1 <- df.nc %>%
  dplyr::select(
    # Factors
    latitude, longitude,
    site_name, year, transect, zone,
    # Bio vars
    den_NERLUE , den_MESFRAAD , den_STRPURAD , den_PYCHEL, den_HALRUF,
    # Nitrate vars
    Days_10N,
    Min_Monthly_Nitrate,
    Max_Monthly_Nitrate,
    Mean_Monthly_Nitrate,
    Mean_Monthly_Upwelling_Nitrate,
    Max_Monthly_Anomaly_Nitrate,
    Mean_Monthly_Summer_Nitrate,
    # Temperature vars
    Days_16C ,
    Mean_Monthly_Temp ,
    Mean_Monthly_Summer_Temp,
    MHW_Upwelling_Days  ,
    Min_Monthly_Anomaly_Temp,
    Max_Monthly_Anomaly_Upwelling_Temp,
    Min_Monthly_Temp,
    Mean_Monthly_Upwelling_Temp,
    #wh.95 ,   wh.max,
    npgo_mean , mei_mean,
    # substrate
    mean_depth, mean_prob_of_rock, mean_vrm, mean_slope,
```

```r
    # waves
    wh_max, wh_mean, mean_waveyear, wh_95prc,
    # Orb vel
    UBR_Mean, UBR_Max,
    # NPP
    Mean_Monthly_NPP, Max_Monthly_NPP_Upwelling, Mean_Monthly_NPP_Upwelling, Min_Monthly_NPP,
  ) %>%
  # Bio transformations
  mutate(log_den_NERLUE = log(den_NERLUE + 1),
         log_den_MESFRAAD = log(den_MESFRAAD + 1),
         log_den_STRPURAD = log(den_STRPURAD + 1),
         log_den_PYCHEL = log(den_PYCHEL + 1),
         log_den_HALRUF = log(den_HALRUF + 1),
         log_mean_vrm = log(mean_vrm + 1)) %>%
  dplyr::select(-c(den_NERLUE,
                   den_MESFRAAD,
                   den_STRPURAD,
                   den_PYCHEL,
                   den_HALRUF,
                   mean_vrm)) %>%
  # Temperature transformations
  mutate(log_Days_16C = log(Days_16C + 1)) %>%
  dplyr::select(-c(Days_16C)) %>%
  # Orb vel transformations
  mutate(log_UBR_Mean = log(UBR_Mean + 1),
         log_UBR_Max = log(UBR_Max + 1)) %>%
  dplyr::select(-c(UBR_Mean,
                   UBR_Max)) %>%
  # NPP transformations
  mutate(log_Mean_Monthly_NPP_Upwelling = log(Mean_Monthly_NPP_Upwelling + 1),
         log_Min_Monthly_NPP = log(Min_Monthly_NPP + 1)) %>%
  dplyr::select(-c(Mean_Monthly_NPP_Upwelling,
                   Min_Monthly_NPP)) # Rows: 708

# log(x + 1) avoids log(0)

#### Drop NAs ----
dat2 <- dat1 %>%
  drop_na() # Rows: 686


# glimpse(dat2)
levels(dat2$year)
```

```
##  [1] "2006" "2007" "2008" "2009" "2010" "2011" "2012" "2013" "2014" "2015"
## [11] "2016" "2017" "2018" "2019" "2020" "2021"
```

3. Divide data into train and test

```r
# Split data into a training set (75%), and a testing set (25%)
inTraining <- createDataPartition(dat2$log_den_NERLUE, p = 0.75, list = FALSE)
```

```r
train.gam <- dat2[ inTraining,]
test.gam  <- dat2[-inTraining,]
```

4. Run GAM

```r
gam1 <- gam(formula = log_den_NERLUE ~
                s(log_den_STRPURAD, k = 5, bs = "cr") + # purple sea urchins
                s(Max_Monthly_Nitrate, k = 5, bs = "cr") +
                s(wh_max, k = 5, bs = "cr") + # wave height
                s(log_UBR_Max, k = 4, bs = "cr") + # orbital velocity, continuous variables
                s(site_name, zone, bs = "re") + # zone is nested within site
                s(year, bs = "re"), # discrete variables, categorical
            # random factor
              family = tw(), data = dat2, method = "REML")

# k: the dimension of the basis used to represent the smooth terms
# bs = 'cr': cubic regression splines
# bs = 're': random effects, penalized by a ridge penalty
```

5. Check GAM

```r
gam1$aic # model selection?
```

```
## [1] 1684.045
```

```r
gam1$deviance # goodness of fit
```

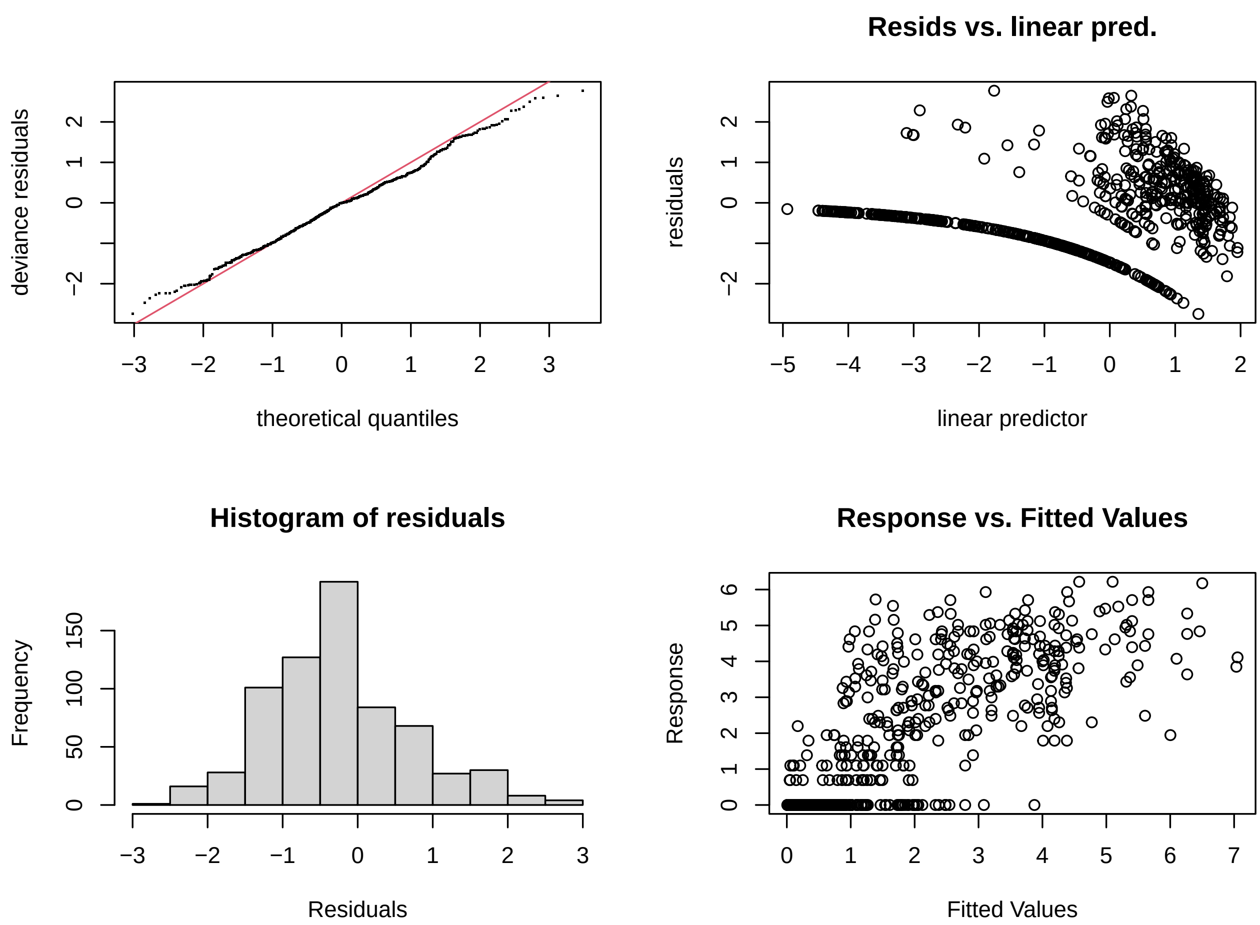
```
## [1] 650.3355
```

```r
summary(gam1)
```

```
##
## Family: Tweedie(p=1.085)
## Link function: log
##
## Formula:
## log_den_NERLUE ~ s(log_den_STRPURAD, k = 5, bs = "cr") + s(Max_Monthly_Nitrate,
##     k = 5, bs = "cr") + s(wh_max, k = 5, bs = "cr") + s(log_UBR_Max,
##     k = 4, bs = "cr") + s(site_name, zone, bs = "re") + s(year,
##     bs = "re")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.5254     0.3789  -1.387    0.166
##
## Approximate significance of smooth terms:
##                              edf Ref.df      F  p-value
```

```
## s(log_den_STRPURAD)      2.358  2.680  8.543 6.17e-05 ***
## s(Max_Monthly_Nitrate)  3.294  3.651 11.159  < 2e-16 ***
## s(wh_max)               3.698  3.918 10.613  < 2e-16 ***
## s(log_UBR_Max)          2.667  2.884  7.611 0.000286 ***
## s(site_name,zone)      15.335 19.000  4.033 2.31e-05 ***
## s(year)                12.784 15.000  9.229  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.623   Deviance explained = 64.5%
## -REML = 880.45  Scale est. = 1.3074    n = 686
```

```
gam.check(gam1) # model diagnostic plots
```




```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-2.23194e-08,3.732632e-09]
## (score 880.4514 & scale 1.30745).
## Hessian positive definite, eigenvalue range [0.5091827,617.4705].
## Model rank =  52 / 52
##
## 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(log_den_STRPURAD)     4.00  2.36    0.80  <2e-16 ***
## s(Max_Monthly_Nitrate)  4.00  3.29    0.59  <2e-16 ***
## s(wh_max)               4.00  3.70    0.59  <2e-16 ***
## s(log_UBR_Max)          3.00  2.67    0.62  <2e-16 ***
## s(site_name,zone)      20.00 15.33      NA      NA
## s(year)                16.00 12.78      NA      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

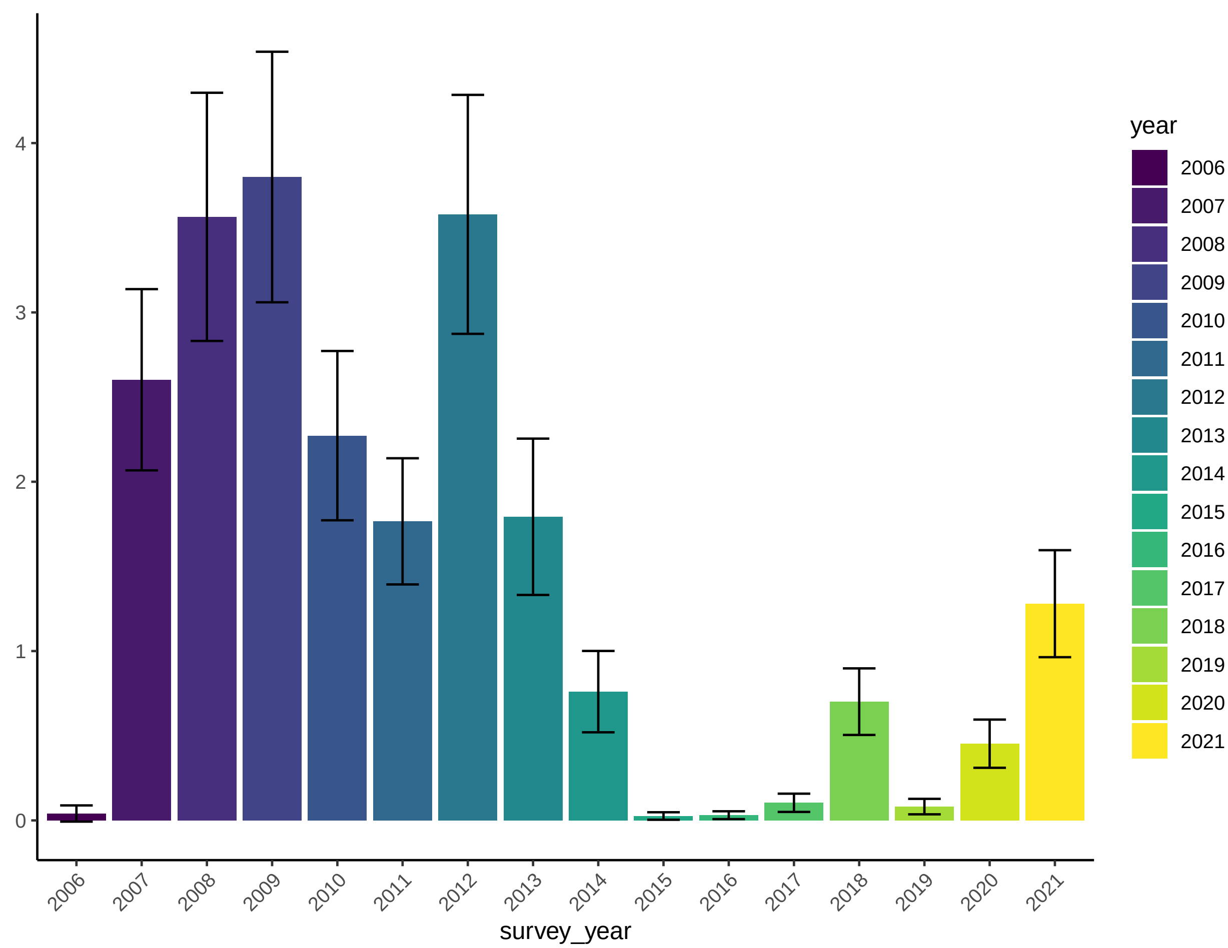
```
# The effective degrees of freedom (EDF) reflects the degree of non-linearity
# of a curve. As the edf increasingly exceeds 2, the degree of non-linearity
# progressively increases.

# Residual plotting aims to show that there is something wrong with the model
# assumptions.
# The key assumptions are
# 1. The assumed mean variance relationship is correct, so that scaled residuals
# have constant variance.
# 2. The response data are independent, so that the residuals appear approximately
# so.

# visualize responses
par(mfrow = c(3, 3), mar = c(2, 4, 3, 1))
visreg(gam1)
dev.off()
```

```
## null device
##           1
```

6. Predict to compare to observed

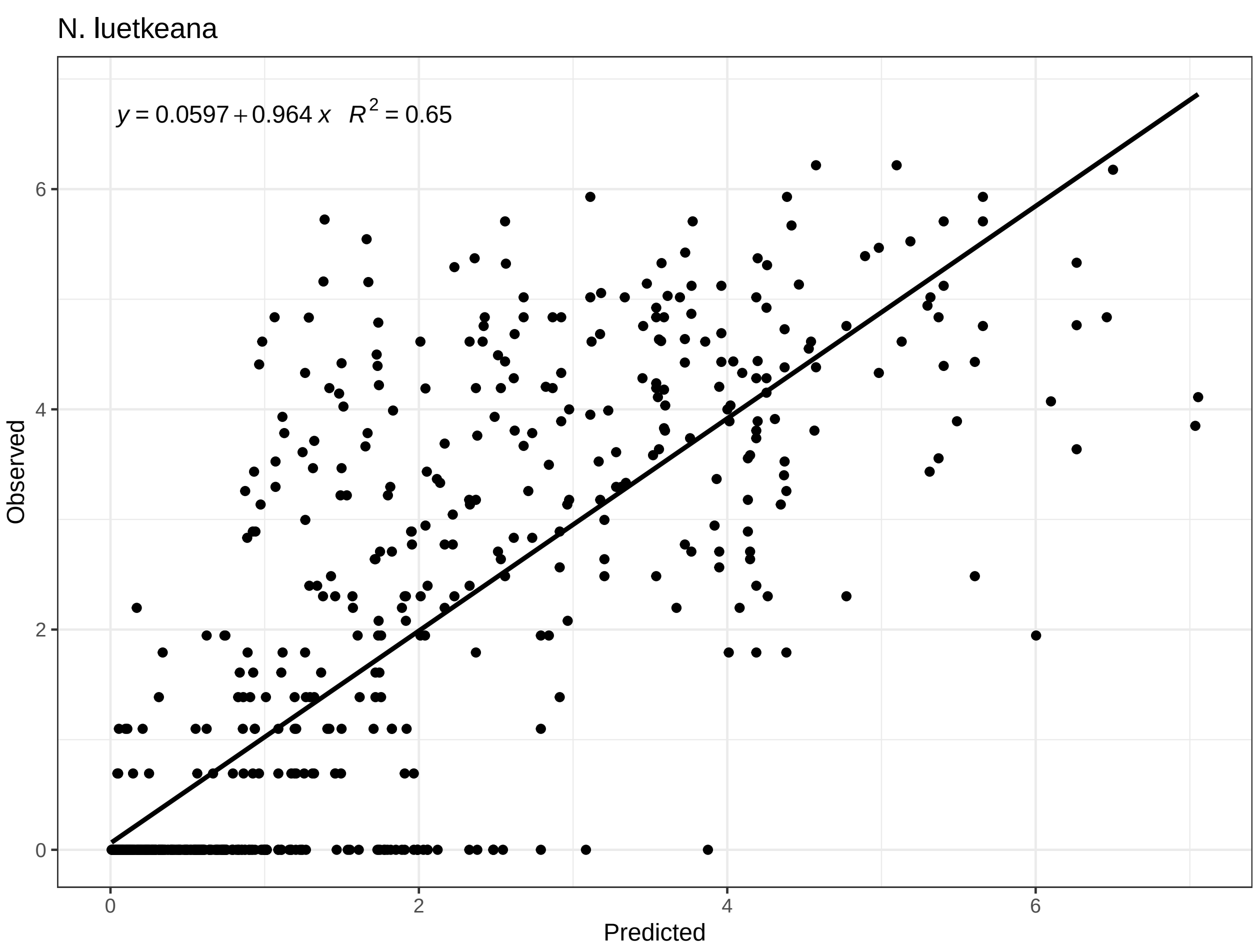
```
testdata <- dat2 %>%
  dplyr::select("log_den_STRPURAD",
                "log_mean_vrm",
                "log_UBR_Max",
                "Max_Monthly_Nitrate",
                "wh_max",
                "log_den_NERLUE",
                "site_name", "zone", "year")

# head(testdata)

# fit the data
fits <- predict.gam(gam1, newdata = testdata, type = 'response', se.fit = T)


## predict average kelp per year --
predicts.year <- testdata %>%
```

```
  data.frame(fits) %>%
  group_by(year) %>% #only change here
  summarise(response = mean(fit, na.rm = T), se.fit = mean(se.fit, na.rm = T)) %>%
  ungroup()


ggmod.year <- ggplot(aes(x = year, y = response, fill = year), data = predicts.year) +
  ylab(" ")+
  xlab('survey_year')+
  scale_fill_viridis(discrete = T) +
  geom_bar(stat = "identity")+
  geom_errorbar(aes(ymin = response - se.fit, ymax = response + se.fit),width = 0.5) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, h = 1))

ggmod.year
```



7. Plot observed vs. predicted

```r
predicts.all <-  testdata %>%
  data.frame(fits) %>%
  #group_by(survey_year) %>% #only change here
  #summarise(response=mean(fit), se.fit = mean(se.fit)) %>%
  ungroup()


# Plot observed vs. predicted --
library(ggpmisc)

my.formula <- y ~ x

p <- ggplot(predicts.all, aes(x = fit, y = log_den_NERLUE)) +
  geom_smooth(method = "lm", se=FALSE, color="black", formula = my.formula) +
  stat_poly_eq(formula = my.formula,
               aes(label = paste(..eq.label.., ..rr.label.., sep = "~~~")),
               parse = TRUE) +
  geom_point() +
  #scale_color_viridis(discrete = T) +
  labs(x = 'Predicted', y = 'Observed', title = 'N. luetkeana') +
  theme_bw()
p
```



Predict best model across all years and site that I have data for

```r
# gam1
# Max_Monthly_Nitrate
# log_UBR_Max
# wh_max
# log_den_STRPURAD

### Get depth ----

depth.dir <- "/Volumes/GoogleDrive/My Drive/SURE_Project/Spatial_data/Predictors/depth"

# names(dat2)

depth <- rast(paste(depth.dir, "depth_mean_nc.all_wInterp_300m_30m.tif", sep ='/'))
# plot(depth)

n.extent <- ext(depth)

# depth # EPSG:26910

crs1 <- "epsg:4326"
d2 <- project(depth, crs1) # depth # EPSG:4326

n.extent <- ext(d2)
```

```r
## Get rock ----
sub.dir <- "/Volumes/GoogleDrive/My Drive/SURE_Project/Spatial_data/Predictors/rock"

# dir(sub.dir)

rock <- rast(paste(sub.dir, "prob_rock_nc_MASKED_LatLong_all_300m_wInterp.tif", sep ='/'))
# rock

# # crop to NC --
rock2 <- crop(rock, ext(d2))
# plot(rock2)

rock3 <- resample(rock2, d2)
# plot(rock3)

# Resample transfers values between non matching Raster objects (in terms of origin)
# and resolution).
```

```r
### Get Env predictors  ----

re.dir <- "/Volumes/GoogleDrive/My Drive/SURE_Project/Spatial_data/Predictors"


### Get nitrate predictors  ----
max_nit <- rast(paste(re.dir, "Nitrate", "Max_Monthly_Nitrate.tif", sep ='/'))
# max_nit # multiple layers
```

```r
# # crop to NC --
max_nit2 <- crop(max_nit, n.extent)
# plot(max_nit2[[1]])

# resample predictors to bathy ----
max_nit3 <- resample(max_nit2, d2)

# mask predictors to bathy ----
max_nit4 <- mask(max_nit3, d2)
# plot(max_nit4[[1]])
```

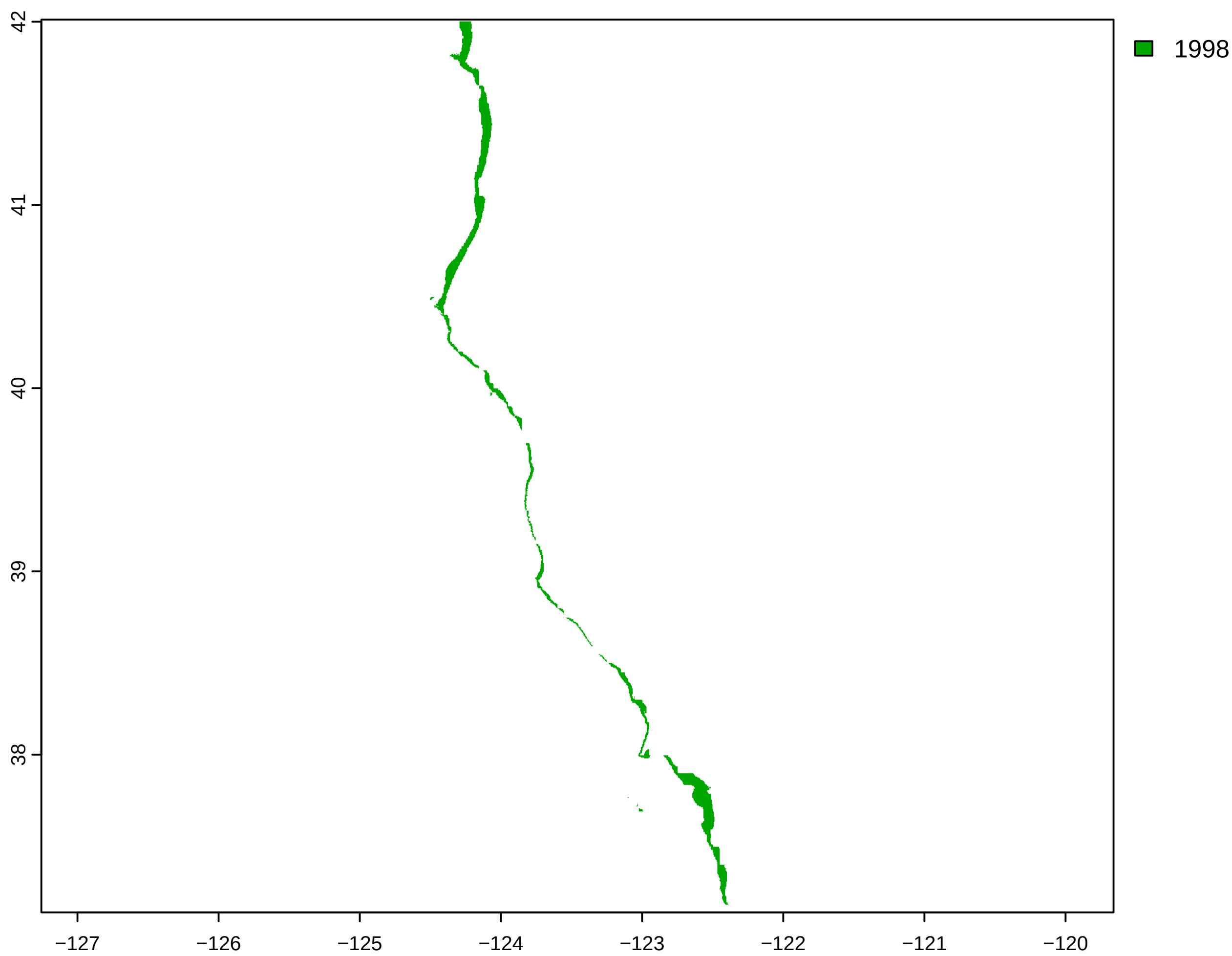
```r
### Get Wave predictors  ----

w.dir <- "/Volumes/GoogleDrive/My Drive/SURE_Project/Spatial_data/Predictors"

## Max Wave height --

# load raster data --

wave.dir <- paste(w.dir, "waves", "wh_max", sep ='/')

# load raster data --

n.files <- dir(wave.dir)
# list files in source --
n.files <- list.files(wave.dir, pattern = '.tif$', full.names = TRUE)
# n.files
# length(n.files)
# list names to load onto the Environment --
names.list <- list.files(wave.dir, pattern = '.tif$')
names.list <- str_replace(names.list, ".tif$", "")
# length(names.list)

# load csv files as a list --
tfiles <- lapply(n.files, rast) # this is a list
# tfiles[[1]]

# stack them ---
whmax.stack <- c()

# use do call to create a raster otherwise it creates a list
whmax.stack <- do.call("c", tfiles)
# plot(whmax.stack[[1]])


# # crop to NC --
whmax.stack2 <- crop(whmax.stack, n.extent)
# plot(whmax.stack2[[1]])

# resample predictors to bathy ----
whmax.stack3 <- resample(whmax.stack2, d2) # align origin, aggregate or disaggregate to have same resol

# mask predictors to bathy ----
```

```r
whmax.stack4 <- mask(whmax.stack3, d2) # make NA do not match to the d2 extent
# plot(whmax.stack4[[1]])
```

```r
## Mean UBR MAX ----

w2.dir <- "/Volumes/GoogleDrive/My Drive/SURE_Project/Spatial_data/Predictors/orbital_vel"

# load raster data --

ubr <- rast(paste(w2.dir, "UBR_Max_30m_NC.tif", sep ='/'))
# plot(ubr[[1]])

ubr <- project(ubr, rock)

# plot(ubr[[1]])

ubr1 <- classify(ubr, cbind(0, NA)) # assign the raster values 0 are reclassified
                                    # to take values NA
# plot(ubr1[[1]])


# # crop to NC --
ubr2 <- crop(ubr1, n.extent)
# plot(ubr2[[1]])

# resample predictors to bathy ----
ubr3 <- resample(ubr2, d2)

# mask predictors to bathy ----
ubr4 <- mask(ubr3, d2)
# plot(ubr4[[1]])

ubr5 <- log(ubr4)
```

```r
### Get urchins ----

# load purple urchin predictions ----
urch.dir <- "/Volumes/GoogleDrive/My Drive/SURE_Project/Spatial_data/Predictors/urchins/log_sp_predicti

# load raster data --

u.files <- dir(urch.dir)
u.files <- list.files(urch.dir, pattern = '.tif')
# u.files
# length(u.files)

##

# stack rasters --

preds1 <- c(max_nit4[[1]], whmax.stack4[[1]], ubr5[[1]])
# names(preds1)
```

```r
# get year raster ----

# make rasters for site, year, zone

# assign the raster values in the range 0-Inf are reclassifed to take values 1998
# max_nit4 has 24 layers
year1998 <- classify(max_nit4[[1]], cbind(0, Inf, 1998), right = FALSE)
plot(year1998)
```




```r
names(year1998) <- 'year'

year.list <- paste(1998:2021)
length(year.list)
```

```
## [1] 24
```

```r
preds2 <- c(preds1, year1998)
# names(preds2)

names(preds2) <- c("Max_Monthly_Nitrate" ,
```

```
                     "wh_max",
                     "log_UBR_Max",
                     "year")
```

```
# sites ----
rdf <- as.data.frame(year1998, xy = T)
# head(rdf)
rdf$site <- rdf$y
rdf <- rdf[,-3] # remove the column 'year'
# head(rdf)

site.raster <- rast(rdf, type = 'xyz', crs = "EPSG:4326", extent = ext(year1998))
# site.raster

# plot(site.raster)

ext(year1998)
```

```
## SpatExtent : -124.57660297128, -122.339221801492, 37.1390505044726, 42.0114450419344 (xmin, xmax, ym
```

```
ext(site.raster)
```

```
## SpatExtent : -124.50379708, -122.39242611, 37.17825368, 42.00024413 (xmin, xmax, ymin, ymax)
```

```
site.raster2 <- extend(site.raster, year1998)

preds3 <- c(preds2, site.raster2)
names(preds3) <- c("Max_Monthly_Nitrate"  ,
                     "wh_max",
                     "log_UBR_Max",
                     "year" ,
                     "site_name")
```

```
# zone ----

zone.raster <- d2
names(zone.raster) <- 'zone'
# plot(zone.raster)
# levels(dat2$zone)

rec.m <-  c(-Inf, -10, 2,
            -10, 0.1, 1)

rclmat <- matrix(rec.m, ncol = 3, byrow = TRUE)
#      [,1]  [,2] [,3]
# [1,] -Inf -10.0    2
# [2,]  -10   0.1    1

zone.raster2 <- classify(zone.raster, rclmat, right = FALSE)
# plot(zone.raster2)

preds4 <- c(preds3, zone.raster2)
```

```r
# names(preds4)

names(preds4) <- c("Max_Monthly_Nitrate" ,
                   "wh_max",
                   "log_UBR_Max",
                   "year",
                   "site_name",
                   "zone")
```

LOOP to predict each year using data frame

```r
nereo.mod <- gam1
summary(nereo.mod)

# make list of years --
year.list <- paste(2004:2021)
length(year.list)

# make template raster of year ----
year.raster <- classify(d2, cbind(-Inf, 0.1, 2004), right=FALSE)
plot(year.raster)
names(year.raster) <- 'year'

# make zone raster ----

# outputs dir ----
# * use an output directory of yours
o2.dir <- here('spatial_data')

preds.dir <- paste(o2.dir, "preds", sep ='/')
preds.dir

# output for rasters scaled by rock
# * use an output directory of yours
rock.preds.dir <- paste(o2.dir, "rock_preds", sep ='/')
rock.preds.dir


for (i in 1:length(year.list)) {

  # 1. get urchins
  urchin.rast <- rast(paste(urch.dir, u.files[i], sep ='/'))
  urchin.rast2 <- resample(urchin.rast, d2)

  # 2. stack with predictors for that year
  #env.raster <- c(d2, max_nit4[[i+6]], whmax.stack4[[i]], wymean.stack4[[i]])

  #env.raster <- c(d2, mean_up_T4[[i+6]], max_nit4[[i+6]], whmax.stack4[[i]], wymean.stack4[[i]])

  # V3
  env.raster <- c(max_nit4[[i+6]], whmax.stack4[[i]], ubr5[[i]])
```

```
preds1 <- c(urchin.rast2, env.raster)

# 3. get year and stack it
year.no <- as.numeric(year.list[i])
year.r <- classify(year.raster, cbind(-Inf, 0, year.no), right=FALSE)

preds2 <- c(preds1, year.r)

# 3. stack zone
preds3 <- c(preds2, zone.raster2)

# 4. stack site
preds4 <- c(preds3, site.raster2)

# name predictors
names(preds4) <- c("log_den_STRPURAD",
                   "Max_Monthly_Nitrate"  ,
                   "wh_max",
                   "log_UBR_Max",
                   "year",
                   "zone",
                   "site_name")

df4 <- as.data.frame(preds4, xy = T) %>%
  mutate_at(vars(year, zone, site_name), list(as.factor)) %>%
  mutate(zone = recode_factor(zone, '1' = 'INNER', '2' = 'OUTER')) %>%
  glimpse()

# 5. predict
year.pred.df <- predict.gam(nereo.mod, newdata = df4, type = 'response', se.fit = T)
head(year.pred.df)

# join with df for lats and lons
preds.all <-  df4 %>%
  data.frame(year.pred.df) %>%
  dplyr::select(x, y, fit) %>%
  glimpse()

# 6. Rasterize
crs.p <- "epsg:4326"
year.prediction <- rast(preds.all, type = 'xyz', crs = crs.p, digits = 6)
plot(year.prediction)

# 7. save raw raster
# name.raster <- paste(year.no, "Log_Nereo_GIVE_IT_A_NAME.tif", sep = '_')
name.raster <- paste(year.no, "Log_Nereo_NC.tif", sep = '_')
writeRaster(year.prediction, paste(preds.dir, name.raster, sep = '/'))

# 8. scale by rock
rock4 <- resample(rock3, year.prediction)
year.prediction2 <- rock4*year.prediction

# 9. save rastr scaled by rock
```

```
  # name.raster.rock <- paste(year.no, "Log_Nereo_rock__GIVE_IT_A_NAME.tif", sep = '_')
  name.raster.rock <- paste(year.no, "Log_Nereo_rock_NC.tif", sep = '_')
  writeRaster(year.prediction2, paste(rock.preds.dir, name.raster.rock, sep = '/'))
}
```