Further improvements to Partial Effects Plots

Curtis C. Bohlen, Casco Bay Estuary Partnership

6/21/2023

Contents

Introduction

This notebook reprises selected analyses using GAMs, and then develops nicer partial effects plots than the `mgcv` defaults. In particular, I'm interested in generating plots of marginal means over data scatter plots. I focus only on graphics that include two related plots of marginal means.

Rachel Lasley-Rasher requested that I assemble plots that lack repeated Y axes. It would probably be possible to construct them directly, assembling a graphic out of two `GROB`s containing graphs and a third containing the axis title and axis labels, but that would not be easy.

The Approach

One alternative is to use `ggplot2`'s faceting capabilities. That is the approach I take here. The core idea is that you can assemble a couple of synthetic data frames (one for marginal means, one for raw data) and then produce faceted graphics.

A couple of Challenges

A few subtleties to make it all work:

1. The labels for the faceting factors most match in the two data frames and
2. Those labels may have to include expressions that `ggplot2` can parse to generate Greek letters, superscripts, etc. in the facet labels
3. Positioning of the facet labels is a bit tricky too.

Unlike the "GAM Analysis Partials" notebook, here I don't rely on a function to generate the graphics in a consistent manner, but instead just work directly with `ggplot()`.

General Instructions to Authors About Graphics

The instructions to authors suggests figure widths should line up with columns, and proposes figure widths should be:

39 mm ~ 1.54 inches 84 mm ~ 3.30 inches 129 mm ~ 5.04 inches 174 mm ~ 6.85 inches

With height not to exceed 235 mm (9.25 inches).

RMarkdown / `knitr` likes figure dimensions in inches. 174 mm is about 6.85 inches

Load Libraries

```
library(tidyverse)
#> -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
#> v dplyr     1.1.1     v readr     2.1.4
#> v forcats   1.0.0     v stringr   1.5.0
#> v ggplot2   3.4.1     v tibble    3.2.1
#> v lubridate 1.9.2     v tidyr     1.3.0
#> v purrr     1.0.1
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
library(readxl)
library(mgcv)      # for GAM models
#> Loading required package: nlme
#>
#> Attaching package: 'nlme'
#>
#> The following object is masked from 'package:dplyr':
#>
#>     collapse
#>
#> This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
library(emmeans)
library(lemon)
#>
#> Attaching package: 'lemon'
#>
#> The following object is masked from 'package:purrr':
#>
#>     %||%
```

Set Graphics Theme

theme_classic() sets `ggplot()`graphics for no background, no grid lines, etc. in a clean format suitable for (some) publications.

The custom theme bundles a number of other aesthetic choices to make graphics more similar to a preferred design generated in some earlier iteration of the graphics.

```
theme_set(theme_classic())

our_facet_theme <- function() {
  theme_classic() %+replace%
    theme(axis.title = element_text(size = 10,color = "gray20"),
          axis.text = element_text(size = 8, color = 'gray50'),
          axis.title.x = element_blank(),
          axis.ticks = element_blank(),
          axis.line = element_line(color = 'gray75'),
          strip.background = element_blank(),
```

```
           strip.placement = "outside",
           strip.text = element_text(size = 10, color = "gray20"),
           panel.spacing.x = unit(-15,"pt"))
}
```

Input Data

Folder References

```
data_folder <- "Original_Data"

dir.create(file.path(getwd(), 'figures'), showWarnings = FALSE)
```

Load Data

```
filename.in <- "penob.station.data EA 3.12.20.xlsx"
file_path <- file.path(data_folder, filename.in)
station_data <- read_excel(file_path,
                           sheet="Final", col_types = c("skip", "date",
                                             "numeric", "text", "numeric",
                                             "text", "skip", "skip",
                                             "skip",
                                             rep("numeric", 10),
                                             "text",
                                             rep("numeric", 47),
                                             "text",
                                             rep("numeric", 12))) %>%
  rename_with(~ gsub(" ", "_", .x)) %>%
  rename_with(~ gsub("\\.", "_", .x)) %>%
  rename_with(~ gsub("\\?", "", .x)) %>%
  rename_with(~ gsub("%", "pct", .x)) %>%
  rename_with(~ gsub("_Abundance", "", .x)) %>%
  filter(! is.na(date))
#> New names:
#> * `` -> `...61`
```

```
names(station_data)[10:12]
#> [1] "discharge_week_cftpersec" "discharg_day"
#> [3] "discharge_week_max"
names(station_data)[10:12] <- c('disch_wk', 'disch_day', 'disch_max')
```

Station names are arbitrary, and Erin previously expressed interest in renaming them from Stations 2, 4, 5 and 8 to Stations 1,2,3,and 4.

The `factor()` function by default sorts levels before assigning numeric codes, so a convenient way to replace the existing station codes with sequential numbers is to create a factor and extract the numeric indicator values with `as.numeric()`.

```r
station_data <- station_data %>%
  mutate(station = factor(as.numeric(factor(station))))
head(station_data)
#> # A tibble: 6 x 76
#>   date                 year month month_num season riv_km station station_num
#>   <dttm>              <dbl> <chr>     <dbl> <chr>   <dbl> <fct>         <dbl>
#> 1 2013-05-28 00:00:00  2013 May           5 Spring  22.6  1                 1
#> 2 2013-05-28 00:00:00  2013 May           5 Spring  13.9  2                 2
#> 3 2013-05-28 00:00:00  2013 May           5 Spring   8.12 3                 3
#> 4 2013-05-28 00:00:00  2013 May           5 Spring   2.78 4                 4
#> 5 2013-07-25 00:00:00  2013 July          7 Summer  22.6  1                 1
#> 6 2013-07-25 00:00:00  2013 July          7 Summer  13.9  2                 2
#> # i 68 more variables: depth <dbl>, disch_wk <dbl>, disch_day <dbl>,
#> #   disch_max <dbl>, tide_height <dbl>, Full_Moon <dbl>, Abs_Moon <dbl>,
#> #   Spring_or_Neap <chr>, ave_temp_c <dbl>, ave_sal_psu <dbl>,
#> #   ave_turb_ntu <dbl>, ave_do_mgperl <dbl>, ave_DO_Saturation <dbl>,
#> #   ave_chl_microgperl <dbl>, sur_temp <dbl>, sur_sal <dbl>, sur_turb <dbl>,
#> #   sur_do <dbl>, sur_chl <dbl>, bot_temp <dbl>, bot_sal <dbl>, bot_turb <dbl>,
#> #   bot_do <dbl>, bot_chl <dbl>, max_temp <dbl>, max_sal <dbl>, ...
```

Subsetting to Desired Data Columns

I base selection of predictor variables here on the ones used in the manuscript.

```r
base_data <- station_data %>%
  rename(Date = date,
         Station = station,
         Year = year) %>%
  select(-c(month, month_num)) %>%
  mutate(Month = factor(as.numeric(format(Date, format = '%m')),
                                               levels = 1:12,
                                               labels = month.abb),
         DOY = as.numeric(format(Date,format = '%j')),
         season = factor(season, levels = c('Spring', 'Summer', 'Fall')),
         is_sp_up = season == 'Spring' & Station == 1,
         Yearf = factor(Year)) %>%
  rename(Season = season,
         Density = combined_density,
         Temp = ave_temp_c,
         Sal = ave_sal_psu,
         Turb = sur_turb,
         AvgTurb = ave_turb_ntu,
         DOsat = ave_DO_Saturation,
         Chl = ave_chl_microgperl,
         Fish = `___61`,
         RH = Herring
         ) %>%
  select(Date, Station, Year, Yearf, Month, Season, is_sp_up, DOY, riv_km,
         disch_wk, disch_day, disch_max,
         Temp, Sal, Turb, AvgTurb, DOsat, Chl,
         Fish, RH,
         Density, H, SEI,
         Acartia, Balanus, Eurytemora, Polychaete, Pseudocal, Temora) %>%
```

```
  arrange(Date, Station)
head(base_data)
#> # A tibble: 6 x 29
#>   Date                Station  Year Yearf Month Season is_sp_up   DOY riv_km
#>   <dttm>              <fct>   <dbl> <fct> <fct> <fct>  <lgl>    <dbl>  <dbl>
#> 1 2013-05-28 00:00:00 1        2013 2013  May   Spring TRUE       148  22.6
#> 2 2013-05-28 00:00:00 2        2013 2013  May   Spring FALSE      148  13.9
#> 3 2013-05-28 00:00:00 3        2013 2013  May   Spring FALSE      148   8.12
#> 4 2013-05-28 00:00:00 4        2013 2013  May   Spring FALSE      148   2.78
#> 5 2013-07-25 00:00:00 1        2013 2013  Jul   Summer FALSE      206  22.6
#> 6 2013-07-25 00:00:00 2        2013 2013  Jul   Summer FALSE      206  13.9
#> # i 20 more variables: disch_wk <dbl>, disch_day <dbl>, disch_max <dbl>,
#> #   Temp <dbl>, Sal <dbl>, Turb <dbl>, AvgTurb <dbl>, DOsat <dbl>, Chl <dbl>,
#> #   Fish <dbl>, RH <dbl>, Density <dbl>, H <dbl>, SEI <dbl>, Acartia <dbl>,
#> #   Balanus <dbl>, Eurytemora <dbl>, Polychaete <dbl>, Pseudocal <dbl>,
#> #   Temora <dbl>
```

```
rm(station_data)
```

Complete Cases

This drops only two samples, one for missing Zooplankton data, one for missing fish data. We need this reduced data set to run The `step()` function. It makes little sense to try stepwise model selection if each time you add or remove a variable, the sample you are studying changes. Since fish is never an important predictor, we will want need to refit models after stepwise elimination to use the most complete possible data set.

```
complete_data <- base_data %>%
  select(Season, Station, Yearf,
         is_sp_up, Temp, Sal, Turb, Chl, Fish, RH,
         Density, H,
         Acartia, Balanus, Eurytemora, Polychaete, Pseudocal, Temora) %>%
  filter(complete.cases(.))
```

Reduced Data

The low salinity spring samples are doing something rather different, and they complicate model fitting. Models are far better behaved if we exclude a few extreme samples. These are low salinity low zooplankton samples. We have two complementary ways to specify which samples to omit, without just omitting "outliers". The first is to restrict modeling to "marine" samples over a certain salinity range, and the other is to omit spring upstream samples, which include most of the problematic samples. We eventually decided to go with the first.

```
drop_low <- complete_data %>%
  filter(Sal > 10)     # Pulls three samples, including one fall upstream sample
                       # a fourth low salinity sample lacks zooplankton data
```

Functions for Data Preparation

I developed versions of these two functions in the "GAM Analysis Partials.Rmd" notebook. See that notebook and the "Testing indirection.Rmd" notebook for more of the logic involved.

Find Evenly Spaced Points

This finds evenly spaced points along the range of a specified variable.

```
find_stops <- function(.dat, .predictor, .nstops = 25) {
  .predictor <- ensym(.predictor)
  r <- range(.dat[[.predictor]])
  stops = seq(r[1], r[2], length.out = .nstops)
  return(stops)
}
```

Conduct The Analysis

This calculates marginal means along one predictor variable in a model. Much of the code complexity handles special cases where either the x or y variables are transformed, which changes the way parts of the output are named.

```
marginal_analysis <- function(.dat, .predictor, .model,
                              .nstops = 25, .logx = TRUE, .transy = TRUE) {
  .predictor <- ensym(.predictor)

  the_name <- as.character(.predictor)
  the_log_name <- paste0("log(", the_name, ")")

  # The following finds stops linear in the original predictor scale.
  # That is appropriate for the planned graphics, where both axes are
  # untransformed.
  stops <- find_stops(.dat, !!.predictor, .nstops)
  # browser()
  if (.logx) {
    stopslist <- list(log(stops))
    names(stopslist) <- the_log_name

    emms <- emmeans(.model, the_log_name,
                    at = stopslist,
                    type = 'response')
    emms <- as_tibble(emms)
    #browser()
    emms <- emms %>%
      mutate( !!the_name := exp(emms[[the_log_name]]))
  }
  else {
    #browser()
    stopslist <- list(stops)
    names(stopslist) <- the_name

    emms <- emmeans(.model, the_name,
```

```
                    at = stopslist,
                    type = 'response')
    emms <- as_tibble(emms)

  }
  #The default name of the output of emmeans() differs if the response
  #variable is transformed or untransformed. This makes names consistent.
  if (! .transy) {
    emms <-  emms %>%
      rename(response = emmean)
  }
  return(emms)
}
```

Total Zooplankton Density

I fit the simplified model without Station. The full model has the same concurvity problems as before, and here the model fails to converge. While I could alter the convergence criteria to search for a solution, we know the model that includes Station will have concurvity problems, so there is little point.

Reduced Complexity Model

```
density_gam_reduced<- gam(log(Density) ~
                          #s(Temp, bs="ts", k = 5) +
                          #s(Sal, bs="ts", k = 5) +
                          s(log(Turb), bs="ts", k = 5) +
                          s(log(Chl), bs="ts", k = 5) +
                          #s(log1p(Fish),bs="ts", k = 5) +
                          s(Yearf, bs = 're'),
                        data = drop_low, family = 'gaussian')
summary(density_gam_reduced)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> log(Density) ~ s(log(Turb), bs = "ts", k = 5) + s(log(Chl), bs = "ts",
#>     k = 5) + s(Yearf, bs = "re")
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   8.1283     0.2307   35.23   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>                 edf Ref.df     F  p-value
#> s(log(Turb)) 1.4120      4  6.18 0.000253 ***
#> s(log(Chl))  0.6072      4  0.83 0.122462
#> s(Yearf)     3.6720      4 10.52 1.63e-06 ***
```

```
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.561   Deviance explained = 60.8%
#> GCV = 0.26018  Scale est. = 0.22853   n = 55
```

Final Graphic

Generate Separate Marginal Means

```
turb <- marginal_analysis(drop_low, Turb, density_gam_reduced,
                                  .nstops = 25, .logx = TRUE)
chl <- marginal_analysis(drop_low, Chl, density_gam_reduced,
                                  .nstops = 25, .logx = TRUE)
```

Name Match

```
names(turb) <- c("log(Pred)", names(turb)[2:6], "Pred")
names(chl) <- c("log(Pred)", names(chl)[2:6], "Pred")
```

```
emms <- bind_rows(Turbidity = turb, Chlorophyll = chl, .id = "source")
```

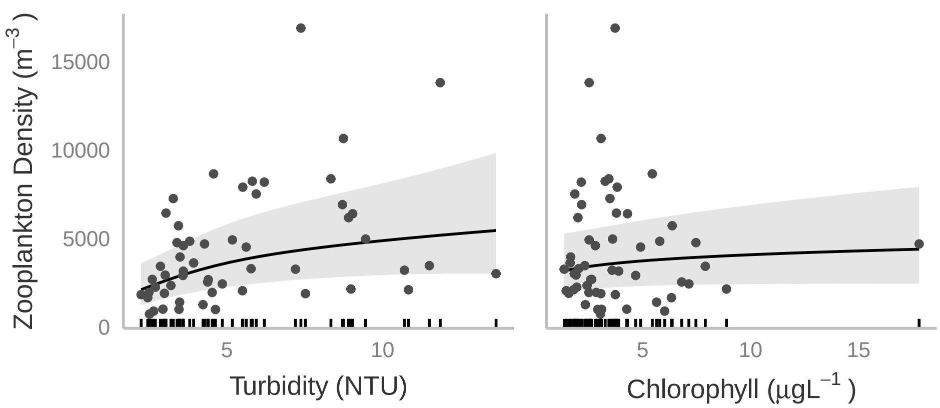
ggplot2 allows you to construct fancier labels by building up something like plotmath expressions. The syntax is a bit obscure, and as far as I can tell, you can only test whether you got it right by plotting the expression.

Assemble Data

```
fancy_turb <- expression("Turbidity" ~ "(NTU)")
fancy_chl  <- expression("Chlorophyll ("* mu * g * L ^-1 ~")")
fancy_dens <- expression("Zooplankton Density (" * m ^-3 ~ ")" )

dat <- drop_low %>%
  select(Turb, Chl, Density) %>%
  pivot_longer(c(Turb, Chl), names_to = "source", values_to = "Pred") %>%
  mutate(source = factor(source,
                         levels = c("Turb", "Chl"),
                         labels = c(fancy_turb, fancy_chl)))

emms <- bind_rows(Turbidity = turb, Chlorophyll = chl, .id = "source")  %>%
  mutate(source = factor(source,
                         levels = c("Turbidity", "Chlorophyll"),
                         labels = c(fancy_turb, fancy_chl)))
```

Draw the Graphic

```
ggplot(emms, aes(Pred, response)) +
  geom_ribbon(aes(ymin = lower.CL, ymax = upper.CL), fill = "grey90") +
  geom_line() +
  geom_point(data = dat, mapping = aes(x = Pred, y = Density),
             size = 1, color = "grey30") +
  geom_rug(data = dat, mapping = aes(x = Pred, y = NULL)) +
  facet_rep_wrap(~source, scales = "free_x", strip.position = "bottom",
             labeller=label_parsed) +
  our_facet_theme() +
  ylab(fancy_dens)
```

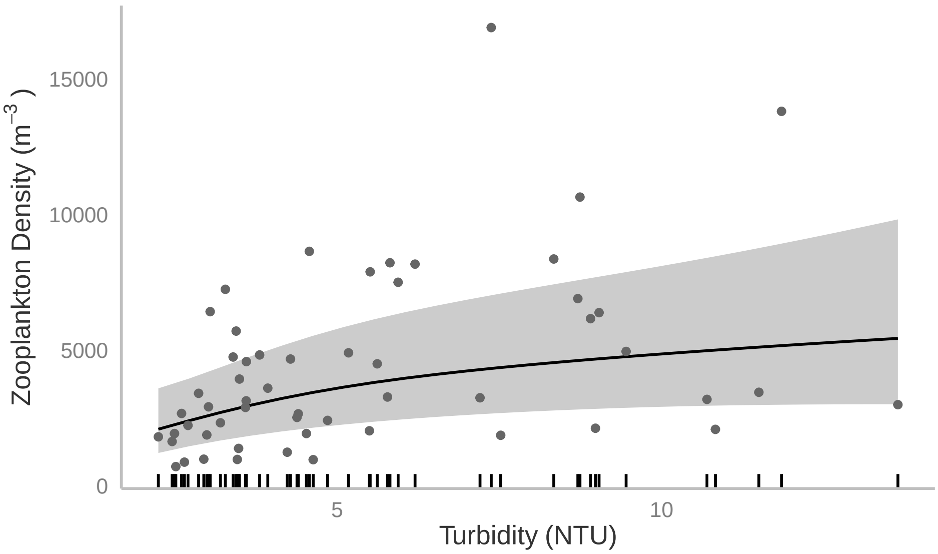


Save the Plot

```
ggsave(file='figures/density_4.png',
width = 5.04, height = 2.2)
ggsave('figures/density_4.pdf', device = cairo_pdf,
width = 5.04, height = 2.2)
```

Supplementary Graphic

```
ggplot(turb, aes(Pred, response)) +
  geom_ribbon(aes(ymin = lower.CL, ymax = upper.CL), fill = "grey80") +
  geom_line() +
  geom_point(data = drop_low, mapping = aes(x = Turb, y = Density),
             size = 1, color = "gray40") +
  geom_rug(data = drop_low, mapping = aes(x = Turb, y = NULL)) +
  our_facet_theme() +
  theme(axis.title.x = element_text()) +
  xlab(expression("Turbidity" ~ "(NTU)")) +
  ylab(expression("Zooplankton Density (" * m ^-3 ~ ")" ))
```



Save Plot

```
ggsave(file='figures/density by turbidity 4.png',
         width = 3.3, height = 2.2)
ggsave('figures/density by turbidity 4.pdf', device = cairo_pdf,
       width = 3.3, height = 2.2)
```

```
ggsave(file='figures/density by turbidity sq 4.png',
         width = 3.3, height = 3.3)
ggsave('figures/density by turbidity sq 4.pdf', device = cairo_pdf,
       width = 3.3, height = 3.3)
```

Shannon Diversity

Model on Reduced Data

```
shannon_gam_no_low <- gam(H ~
                      s(Temp, bs="ts", k = 5) +
                      s(Sal, bs="ts", k = 5) +
                      s(log(Turb), bs="ts", k = 5) +
                      s(log(Chl), bs="ts", k = 5) +
                      s(log1p(Fish),bs="ts", k = 5) +
                      s(Yearf, bs = 're'),
                    data = drop_low, family = 'gaussian')
summary(shannon_gam_no_low)
#>
#> Family: gaussian
#> Link function: identity
#>
```

```
#> Formula:
#> H ~ s(Temp, bs = "ts", k = 5) + s(Sal, bs = "ts", k = 5) + s(log(Turb),
#>     bs = "ts", k = 5) + s(log(Chl), bs = "ts", k = 5) + s(log1p(Fish),
#>     bs = "ts", k = 5) + s(Yearf, bs = "re")
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   1.3310     0.1142   11.66  3.1e-15 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>                      edf Ref.df      F  p-value
#> s(Temp)        1.615e+00      4  4.222 0.002901 **
#> s(Sal)         2.259e-08      4  0.000 0.257386
#> s(log(Turb))   1.369e-08      4  0.000 0.608480
#> s(log(Chl))    3.767e+00      4 11.002 0.000252 ***
#> s(log1p(Fish)) 3.675e-01      4  0.167 0.197576
#> s(Yearf)       2.929e+00      4  2.802 0.008131 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.417   Deviance explained = 51.1%
#> GCV =    0.2  Scale est. = 0.1648    n = 55
```

Final Graphic

Generate Separate Marginal Means

```
temp <- marginal_analysis(drop_low, Temp, shannon_gam_no_low,
                                  .nstops = 25, .logx = FALSE)
chl <- marginal_analysis(drop_low, Chl, shannon_gam_no_low,
                                  .nstops = 25, .logx = TRUE, .transy = FALSE)
```

```
names(temp) <- c("Pred", names(temp)[2:6])
names(chl) <- c("log(Pred)", names(chl)[2:6], "Pred")
chl <- chl[,c(7, 2:6)]
```

Name match

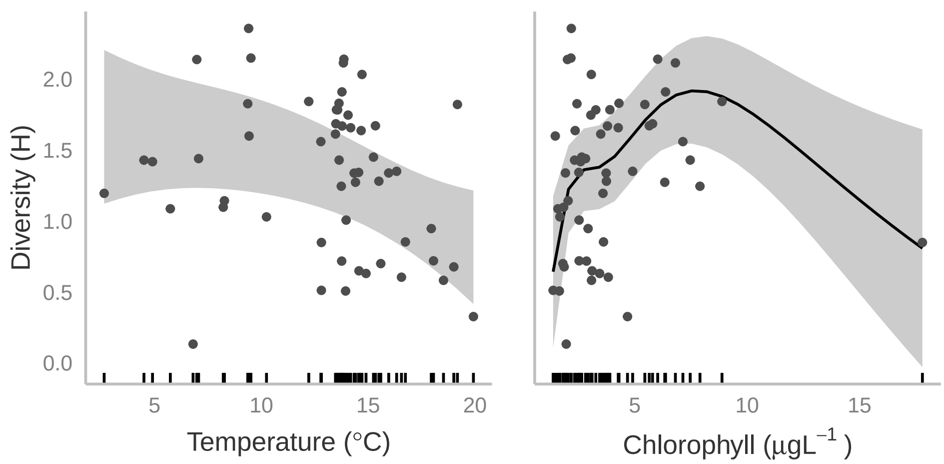
Assemble data

```
fancy_temp <- expression("Temperature (" * degree * "C)")
fancy_chl  <- expression("Chlorophyll ("* mu * g * L ^-1 ~")")
dat <- drop_low %>%
  select(Temp, Chl, H) %>%
```

```
  pivot_longer(c(Temp, Chl), names_to = "source", values_to = "Pred") %>%
  mutate(source = factor(source,
                         levels = c("Temp", "Chl"),
                         labels = c(fancy_temp, fancy_chl)))

emms <- bind_rows(Temperature = temp, Chlorophyll = chl, .id = "source")  %>%
  mutate(source = factor(source,
                         levels = c("Temperature", "Chlorophyll"),
                         labels = c(fancy_temp, fancy_chl)))
```

Generate Plot

```
ggplot(emms, aes(Pred, response)) +
  geom_ribbon(aes(ymin = lower.CL, ymax = upper.CL), fill = "grey80") +
  geom_line() +
  geom_point(data = dat, mapping = aes(x = Pred, y = H),
             size = 1, color = "grey30") +
  geom_rug(data = dat, mapping = aes(x = Pred, y = NULL)) +
  facet_rep_wrap(~source, scales = "free_x", strip.position = "bottom",
             labeller=label_parsed) +
  our_facet_theme() +
  theme(panel.spacing.x = unit(0,"pt")) +
  ylab("Diversity (H)")
#> Warning: Removed 25 rows containing missing values (`geom_line()`).
```



###Save the Plot

```
ggsave(file='figures/shannon_4.png',
         width = 5, height = 2.2)
#> Warning: Removed 25 rows containing missing values (`geom_line()`).
ggsave('figures/shannon_4.pdf', device = cairo_pdf,
       width = 5, height = 2.2)
#> Warning: Removed 25 rows containing missing values (`geom_line()`).
```

Single Species Models

Model Choice

Our model alternatives are similar to the choices we had for the Total Density model. The problem is, we can't use any of the continuous data distributions in GAMS with zero values (at least relying on the canonical link functions) because (log(0) = -Inf; 1/0 = Inf, 1 / 0*0 = Inf). The easiest solution is to add some finite small quantity to the density data, and predict that. Here we predict log(Density + 1) using Gaussian models.

Automating Analysis of Separate Species

I'm going to automate analysis of all selected species by using a "nested" Tibble. This is a convenient alternative to writing a "for" loop to run multiple identical analyses.

I create a "long" data source, based on the reduced data set that omits low salinity samples.

```
spp_data <- drop_low %>%
  select(Yearf, Season, Station, Temp,
         Sal, Turb, Chl, Fish,
         Acartia, Balanus, Eurytemora) %>%
  pivot_longer(-c(Yearf:Fish), names_to = 'Species', values_to = 'Density')
```

Next, I create a function to run the analysis. This function takes a data frame or tibble as an argument. The tibble mush have data columns with the correct names.

The initial model fits for some species had a lot of wiggles in them, to an extent that I thought did not make much scientific sense, so I decided to reduce the dimensionality of the GAM smoothers, by adding the parameter `k= 4`. Lowe numbers constrain the GAM to fit smoother lines.

```
my_gam <- function(.dat) {

  gam(log1p(Density) ~
        s(Temp, bs="ts", k = 5) +
        s(Sal, bs="ts", k = 5) +
        s(log(Turb), bs="ts", k = 5) +
        s(log(Chl), bs="ts", k = 5) +
        s(log1p(Fish),bs="ts", k = 5) +
        s(Yearf, bs = 're'),
      data = .dat, family = "gaussian")
}
```

Next, I create the nested tibble, and conduct the analysis on each species. . . .

```
spp_analysis <- spp_data %>%
  group_by(Species) %>%
  nest() %>%
  mutate(gam_mods = map(data, my_gam))
```

And finally, output the model results. I can do that in a "for" loop, but it's Awkward to look through a long list of output, so I step through each species in turn.

Acartia

```r
spp = 'Acartia'
mod <- spp_analysis$gam_mods[spp_analysis$Species == spp][[1]]
dat <- spp_analysis$data[spp_analysis$Species == spp][[1]]
summary(mod)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> log1p(Density) ~ s(Temp, bs = "ts", k = 5) + s(Sal, bs = "ts",
#>     k = 5) + s(log(Turb), bs = "ts", k = 5) + s(log(Chl), bs = "ts",
#>     k = 5) + s(log1p(Fish), bs = "ts", k = 5) + s(Yearf, bs = "re")
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    6.598      0.371   17.78   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>                   edf Ref.df      F  p-value
#> s(Temp)        3.6631      4 31.950  < 2e-16 ***
#> s(Sal)         3.2713      4  7.570 0.000232 ***
#> s(log(Turb))   0.6538      4  0.637 0.076037 .
#> s(log(Chl))    0.7323      4  1.316 0.055331 .
#> s(log1p(Fish)) 0.7316      4  0.610 0.080622 .
#> s(Yearf)       3.5153      4 11.237 6.14e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.763   Deviance explained = 81.8%
#> GCV = 0.93657  Scale est. = 0.70553   n = 55
```

Final Graphic

```r
temp <- marginal_analysis(dat, Temp, mod, .logx = FALSE)
```

```r
sal <- marginal_analysis(dat, Sal, mod, .logx = FALSE)
```

Generate Separate Marginal Means

```r
names(temp) <- c("Pred", names(temp)[2:6])
names(sal) <- c("Pred", names(sal)[2:6])
```

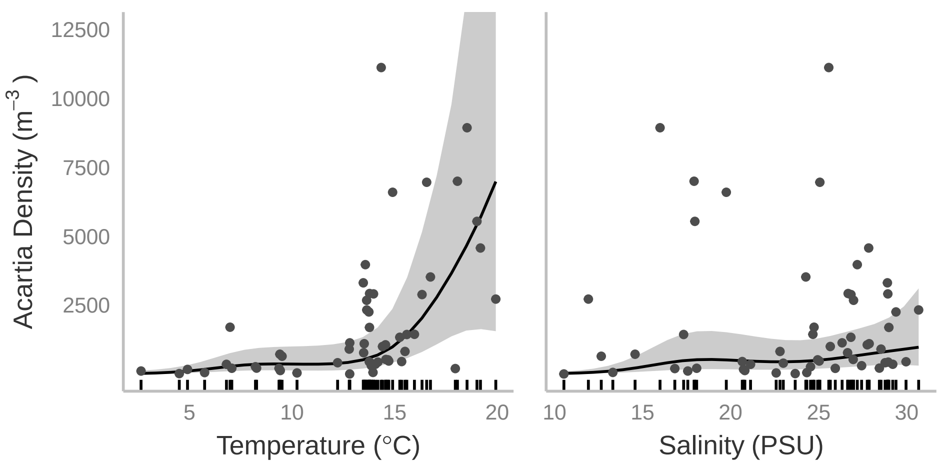
Name match

```
fancy_temp <- expression("Temperature (" * degree * "C)")
fancy_sal  <- expression("Salinity" ~ "(PSU)")

dat <- dat %>%
  select(Temp, Sal, Density) %>%
  pivot_longer(c(Temp, Sal), names_to = "source", values_to = "Pred") %>%
  mutate(source = factor(source,
                         levels = c("Temp", "Sal"),
                         labels = c(fancy_temp, fancy_sal)))

emms <- bind_rows(Temperature = temp, Salinity = sal, .id = "source")  %>%
  mutate(source = factor(source,
                         levels = c("Temperature", "Salinity"),
                         labels = c(fancy_temp, fancy_sal)))
```

Assemble data

```
ggplot(emms, aes(Pred, response)) +
  geom_ribbon(aes(ymin = lower.CL, ymax = upper.CL), fill = "grey80") +
  geom_line() +
  geom_point(data = dat, mapping = aes(x = Pred, y = Density),
             size = 1, color = "grey30") +
  geom_rug(data = dat, mapping = aes(x = Pred, y = NULL)) +
  facet_rep_wrap(~source, scales = "free_x", strip.position = "bottom",
             labeller=label_parsed) +
 our_facet_theme() +
  ylab(expression("Acartia Density (" * m ^-3 ~ ")" )) +
  scale_y_continuous(breaks = c(1:5*2500)) +
  coord_cartesian(ylim = c(0, 12500))
```

Generate Plot



Save Plot

```
ggsave(file='figures/Acartia_4.png',
         width = 5.04, height = 2.5)
ggsave('figures/Acartia_4.pdf', device = cairo_pdf,
       width = 5.04, height = 2.5)
```

Balanus

```
spp = 'Balanus'
mod <- spp_analysis$gam_mods[spp_analysis$Species == spp][[1]]
dat <- spp_analysis$data[spp_analysis$Species == spp][[1]]
summary(mod)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> log1p(Density) ~ s(Temp, bs = "ts", k = 5) + s(Sal, bs = "ts",
#>     k = 5) + s(log(Turb), bs = "ts", k = 5) + s(log(Chl), bs = "ts",
#>     k = 5) + s(log1p(Fish), bs = "ts", k = 5) + s(Yearf, bs = "re")
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   3.6930     0.6478   5.701 8.74e-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(Temp)        9.192e-01      4  2.998  0.00414 **
#> s(Sal)         1.782e-10      4  0.000  0.52552
#> s(log(Turb))   1.967e+00      4  1.779  0.06016 .
#> s(log(Chl))    1.004e+00      4 14.125 2.07e-05 ***
#> s(log1p(Fish)) 1.686e+00      4  0.691  0.22444
#> s(Yearf)       3.568e+00      4  7.912 1.75e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.581   Deviance explained = 65.2%
#> GCV = 2.7021  Scale est. = 2.2038    n = 55
```

Final Graphic

```
temp <- marginal_analysis(dat, Temp, mod, .logx = FALSE)
```

```r
chl <- marginal_analysis(dat, Chl, mod, .logx = TRUE)
```

Generate Separate Marginal Means

```r
names(temp) <- c("Pred", names(temp)[2:6])
names(chl) <- c("log(Pred)", names(chl)[2:6], "Pred")
chl <- chl[,c(7, 2:6)]
```

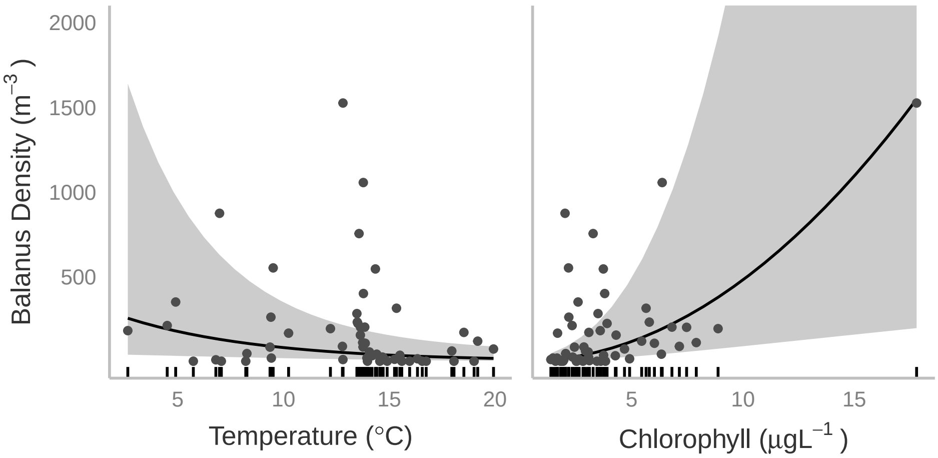
Name match

```r
fancy_temp <- expression("Temperature (" * degree * "C)")
fancy_chl  <- expression("Chlorophyll ("* mu * g * L ^-1 ~")")

dat <- dat %>%
  select(Temp, Chl, Density) %>%
  pivot_longer(c(Temp, Chl), names_to = "source", values_to = "Pred") %>%
  mutate(source = factor(source,
                         levels = c("Temp", "Chl"),
                         labels = c(fancy_temp, fancy_chl)))

emms <- bind_rows(Temperature = temp, Chlorophyll = chl, .id = "source")  %>%
  mutate(source = factor(source,
                         levels = c("Temperature", "Chlorophyll"),
                         labels = c(fancy_temp, fancy_chl)))
```

Assemble data

```r
ggplot(emms, aes(Pred, response)) +
  geom_ribbon(aes(ymin = lower.CL, ymax = upper.CL), fill = "grey80") +
  geom_line() +
  geom_point(data = dat, mapping = aes(x = Pred, y = Density),
             size = 1, color = "grey30") +
  geom_rug(data = dat, mapping = aes(x = Pred, y = NULL)) +
  facet_rep_wrap(~source, scales = "free_x", strip.position = "bottom",
             labeller=label_parsed) +
  our_facet_theme() +
  ylab(expression("Balanus Density (" * m ^-3 ~ ")" )) +
  scale_y_continuous(breaks = c(1:4*500)) +
  coord_cartesian(ylim = c(0, 2000))
```

Generate Plot




```r
ggsave(file='figures/Balanus_4.png',
         width = 5.04, height = 2.5)
ggsave('figures/Balanus_4.pdf', device = cairo_pdf,
       width = 5.05, height = 2.5)
```

Save Plot

Eurytemora

```r
spp =  "Eurytemora"
mod <- spp_analysis$gam_mods[spp_analysis$Species == spp][[1]]
dat <- spp_analysis$data[spp_analysis$Species == spp][[1]]
summary(mod)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> log1p(Density) ~ s(Temp, bs = "ts", k = 5) + s(Sal, bs = "ts",
#>     k = 5) + s(log(Turb), bs = "ts", k = 5) + s(log(Chl), bs = "ts",
#>     k = 5) + s(log1p(Fish), bs = "ts", k = 5) + s(Yearf, bs = "re")
#>
#> Parametric coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   6.5275     0.1297   50.34   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Approximate significance of smooth terms:
#>                     edf Ref.df     F  p-value
#> s(Temp)       8.514e-10      4 0.000 0.522777
#> s(Sal)        1.698e+00      4 0.439 0.360992
#> s(log(Turb))  9.561e-01      4 3.326 0.000375 ***
```

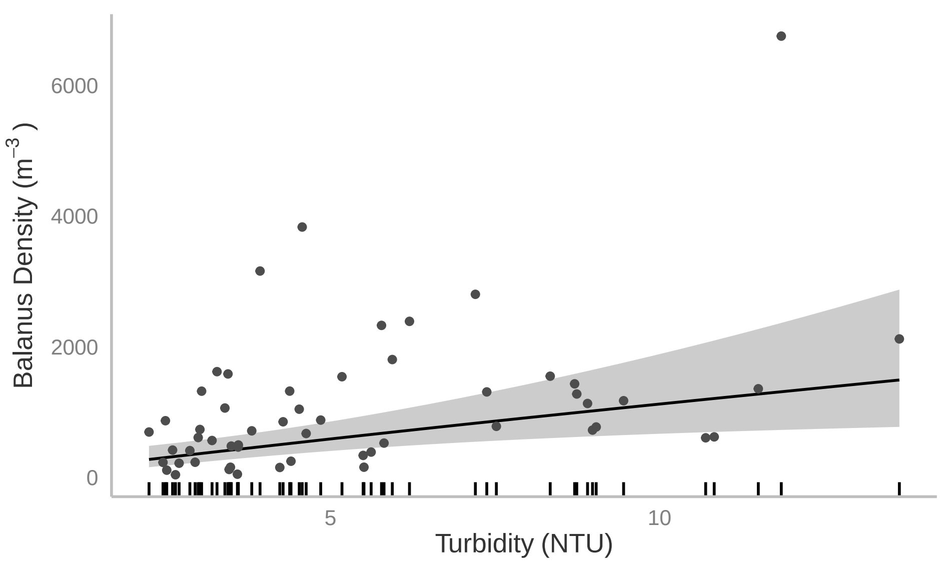
```
#> s(log(Chl))     1.509e+00      4 0.541 0.241190
#> s(log1p(Fish)) 4.650e-10      4 0.000 0.340527
#> s(Yearf)       3.805e-01      4 0.101 0.368007
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.241   Deviance explained = 30.5%
#> GCV = 0.91936  Scale est. = 0.8267    n = 55
```

Generate Data

```r
p1.emms <- marginal_analysis(dat, Turb, mod,
                                  .nstops = 25, .logx = TRUE)
```

Generate Graphic

```r
ggplot(p1.emms, aes(Turb, response)) +
  geom_ribbon(aes(ymin = lower.CL, ymax = upper.CL), fill = "grey80") +
  geom_line() +
  geom_point(data = dat, mapping = aes(x = Turb, y = Density),
             size = 1, color = "grey30") +
  geom_rug(data = dat, mapping = aes(x = Turb, y = NULL)) +

  our_facet_theme() +
  theme(axis.title.x = element_text()) +

  xlab(expression("Turbidity" ~ "(NTU)")) +
  ylab(expression("Balanus Density (" * m ^-3 ~ ")" ))
```



Save Plot

```
ggsave(file='figures/Eurytemora_4.png',
         width = 3.3, height = 2.2)
ggsave('figures/Eurytemora_4.pdf', device = cairo_pdf,
       width = 3.3, height = 2.2)
```

```
ggsave(file='figures/Eurytemora_sq_4.png',
         width = 3.3, height = 3.3)
ggsave('figures/Eurytemora_sq_4.pdf', device = cairo_pdf,
       width = 3.3, height = 3.3)
```