Improved Partial Effects Plots for GAMs Looking at the Plankton Community

Curtis C. Bohlen, Casco Bay Estuary Partnership

7/21/2022

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 marginal plots as follows (the page numbers refer to where the "draft" marginal plots appear in the PDF of the original analysis).

1. Zoop density vs. Turbidity (pg. 20)
2. Zoop diversity vs. Chl & Zoop diversity vs. Temp (combined 2-part figure), pg. 25
3. Eurytemora density vs. Turbidity, pg. 34
4. Barnacle vs. Chl & Barnacle vs. Temp (combined 2-part figure), pg. 31
5. Acartia vs. Temp & Acartia vs. Salinity (combined 2-part figure), pg. 29

Load Libraries

```
library(tidyverse)
#> -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
#> v ggplot2 3.3.6     v purrr   0.3.4
#> v tibble  3.1.7     v dplyr   1.0.9
#> v tidyr   1.2.0     v stringr 1.4.0
#> v readr   2.1.2     v forcats 0.5.1
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
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 1.8-40. For overview type 'help("mgcv-package")'.
library(ggeffects)
library(gridExtra) # or could use related functions in `cowplot`
#>
#> Attaching package: 'gridExtra'
#> The following object is masked from 'package:dplyr':
#>
#>     combine
```

Set Graphics Theme

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

```
theme_set(theme_classic())
```

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()`.

```
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
#> # ... with 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.

```
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
#> # ... with 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, and the other is to omit spring upstream samples, which include most of the problematic samples.

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

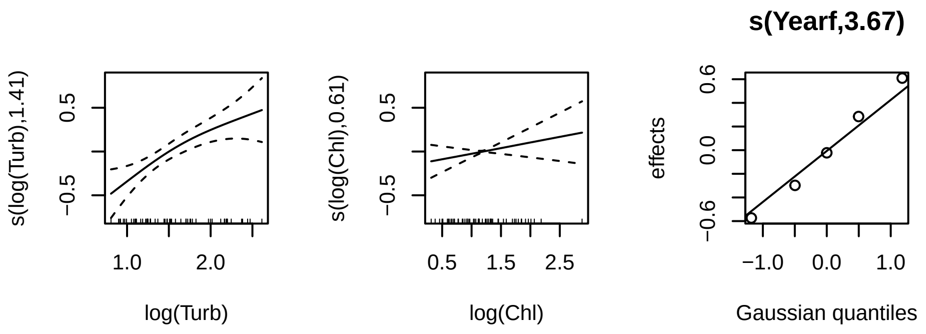
Total Zooplankton Density

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
```

```
oldpar <- par(mfrow = c(2,3))
plot(density_gam_reduced)
par(oldpar)
```



Combined Graphic

As far as I have been able to tell, you need to build up the combined plots for our mixed model GAMS piecewise. The easiest way is to pull marginal means for each marginal predictor and then assemble them using `grid.arrange()` (for display) or `arrangeGrob()` (for saving with `ggsave()`).

```
ggp1 <- ggemmeans(density_gam_reduced, terms = 'Turb')
#> Loading required namespace: emmeans
#> Model has log-transformed response. Back-transforming predictions to original response scale. Standa
ggp2 <- ggemmeans(density_gam_reduced, terms = 'Chl')
#> Model has log-transformed response. Back-transforming predictions to original response scale. Standa
```

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

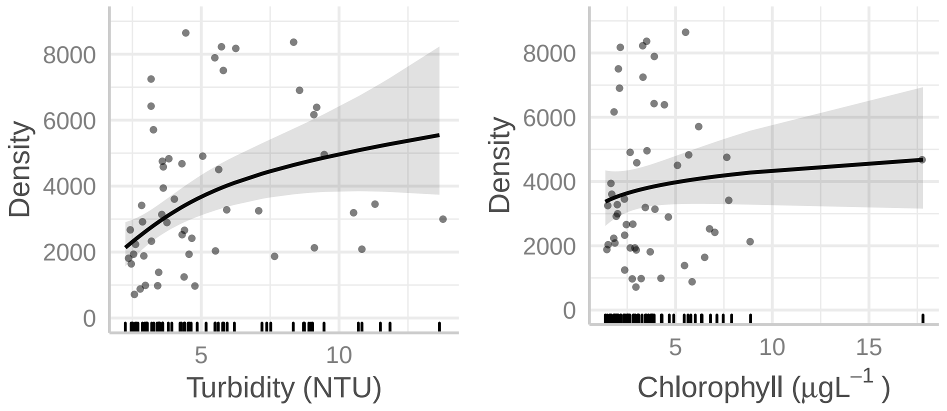
39 mm ~ 1.54 inches 84 mm ~ 3.30 inches 129 mm ~ 5.04 inches 174 mm ~ 6.85 inches

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

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

```
plt1 <- plot(ggp1, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  ggtitle('') +
  xlab(expression("Turbidity" ~ "(NTU)")) +
  scale_y_continuous( limits = c(0,9000), breaks = c(0, 2000, 4000, 6000, 8000))
#> Scale for 'y' is already present. Adding another scale for 'y', which will
#> replace the existing scale.
```

```
plt2 <- plot(ggp2, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  xlab(expression("Chlorophyll (" * mu * g * L ^-1 ~")")) +
  ggtitle('') +
  scale_y_continuous( limits = c(0,9000), breaks = c(0, 2000, 4000, 6000, 8000))
#> Scale for 'y' is already present. Adding another scale for 'y', which will
#> replace the existing scale.

plt3 <- grid.arrange(plt1, plt2, # alternatives here from cowplot and other pacj=hages,
             ncol = 2, nrow = 1)
#> Warning: Removed 3 rows containing missing values (geom_point).
#> Removed 3 rows containing missing values (geom_point).
```

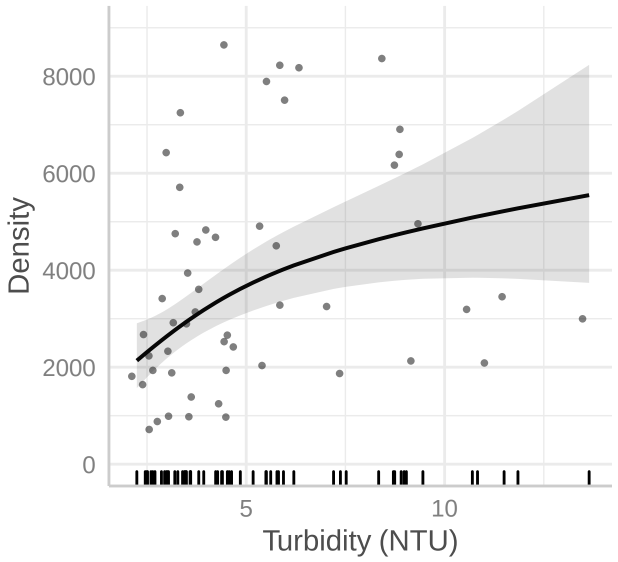


```
#plt3
```

```
grphc <- arrangeGrob(plt1, plt2, ncol = 2, nrow = 1) #generates graphic
#> Warning: Removed 3 rows containing missing values (geom_point).
#> Removed 3 rows containing missing values (geom_point).

ggsave(file='figures/density.png', grphc, type='cairo',
         width = 5, height = 2.5)
ggsave('figures/density.pdf', grphc, device = cairo_pdf,
       width = 5, height = 2.5)
```

Requested Graphic

```
ggp <- ggemmeans(density_gam_reduced, terms = "Turb")
#> Model has log-transformed response. Back-transforming predictions to original response scale. Standa
plt <- plot(ggp, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  ggtitle('') +
  xlab(expression("Turbidity" ~ "(NTU)")) +
  scale_y_continuous( limits = c(0,9000), breaks = c(0, 2000, 4000, 6000, 8000))
#> Scale for 'y' is already present. Adding another scale for 'y', which will
#> replace the existing scale.
plt
#> Warning: Removed 3 rows containing missing values (geom_point).
```



```
ggsave('figures/density by turbidity.png', type='cairo',
         width = 6.85, height = 4.9)
#> Warning: Removed 3 rows containing missing values (geom_point).
ggsave('figures/density by turbidity.pdf', device = cairo_pdf,
       width = 6.85, height = 4.9)
#> Warning: Removed 3 rows containing missing values (geom_point).
```

Shannon Diversity

Model on Reduced Data

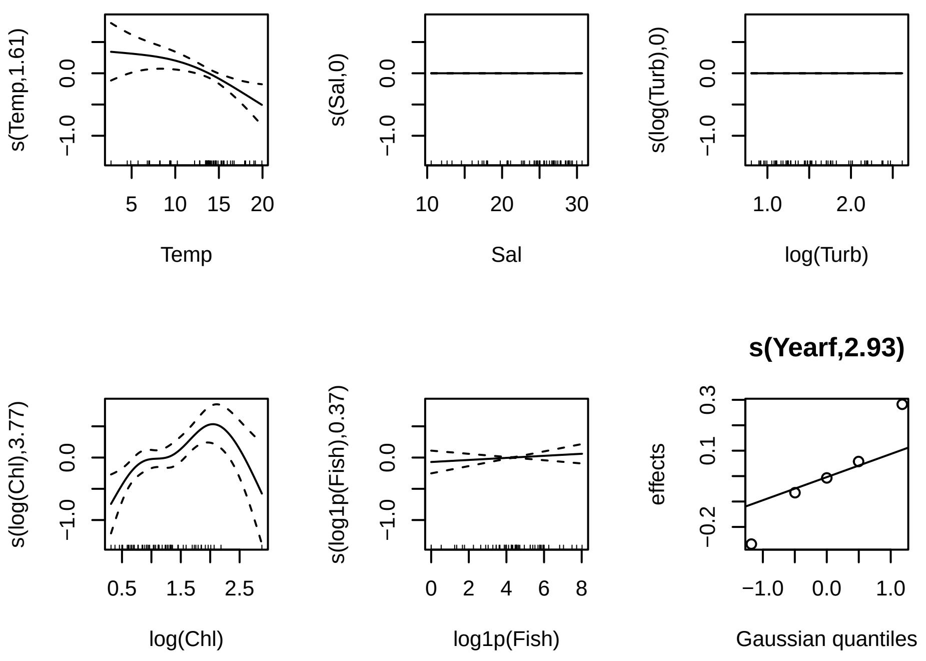
```
shannon_gam_no_low <- gam(H ~
                     s(Temp, bs="ts", k = 5) +
                     s(Sal, bs="ts", k = 5) +
                     s(log(Turb), bs="ts", k = 5) +
                     s(log(Chl), bs="ts", k = 5) +
                     s(log1p(Fish),bs="ts", k = 5) +
                     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
```

```
anova(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")
#>
#> Approximate significance of smooth terms:
#>                      edf    Ref.df      F  p-value
#> s(Temp)        1.615e+00 4.000e+00  4.222 0.002901
#> s(Sal)         2.259e-08 4.000e+00  0.000 0.257386
#> s(log(Turb))   1.369e-08 4.000e+00  0.000 0.608480
#> s(log(Chl))    3.767e+00 4.000e+00 11.002 0.000252
#> s(log1p(Fish)) 3.675e-01 4.000e+00  0.167 0.197576
#> s(Yearf)       2.929e+00 4.000e+00  2.802 0.008131
```

Plot the GAM

```
oldpar <- par(mfrow = c(2,3))
plot(shannon_gam_no_low)
```



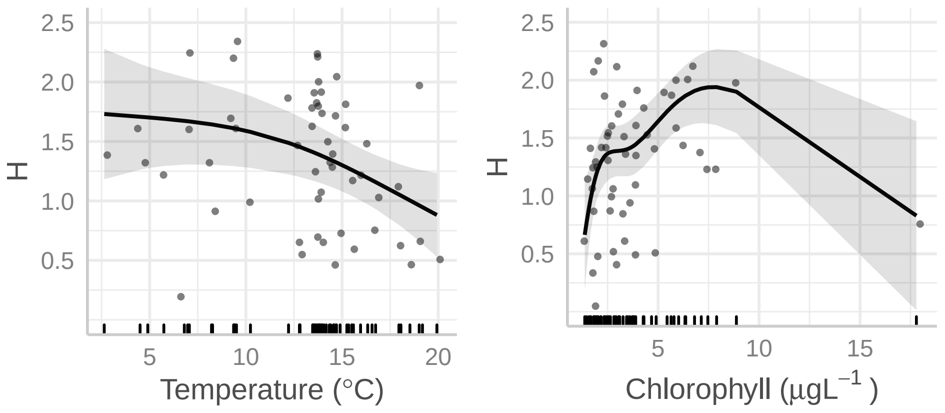
```r
par(oldpar)
```

Combined Graphic

As far as I have been able to tell, you need to build up the combined plots for our mixed model GAMS piecewise. The easiest way is to pull marginal means for each marginal predictor and then assemble them using `grid.arrange()` (for display) or `arrangeGrob()` (for saving with `ggsave()`).

```r
ggp1 <- ggemmeans(shannon_gam_no_low, terms = 'Temp')
ggp2 <- ggemmeans(shannon_gam_no_low, terms = 'Chl')
```

```r
plt1 <- plot(ggp1, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  ggtitle('') +
  xlab(expression("Temperature (" * degree * "C)")) +
  scale_y_continuous( limits = c(0,2.5), breaks = c(1:5*.5))
#> Scale for 'y' is already present. Adding another scale for 'y', which will
#> replace the existing scale.

plt2 <- plot(ggp2, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  xlab(expression("Chlorophyll (" * mu * g * L ^-1 ~")")) +
  ggtitle('') +
  scale_y_continuous( limits = c(0,2.5), breaks = c(1:5*.5))
#> Scale for 'y' is already present. Adding another scale for 'y', which will
#> replace the existing scale.

plt3 <- grid.arrange(plt1, plt2, # alternatives here from cowplot and other pacj=hages,
             ncol = 2, nrow = 1)
```



```
#plt3
```

```
grphc <- arrangeGrob(plt1, plt2, ncol = 2, nrow = 1) #generates graphic
#> Warning: Removed 1 rows containing missing values (geom_point).
#> Removed 1 rows containing missing values (geom_point).

ggsave(file='figures/shannon.png', grphc, type='cairo',
         width = 5, height = 2.5)
ggsave('figures/shannon.pdf', grphc, device = cairo_pdf,
       width = 5, height = 2.5)
```

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))
```

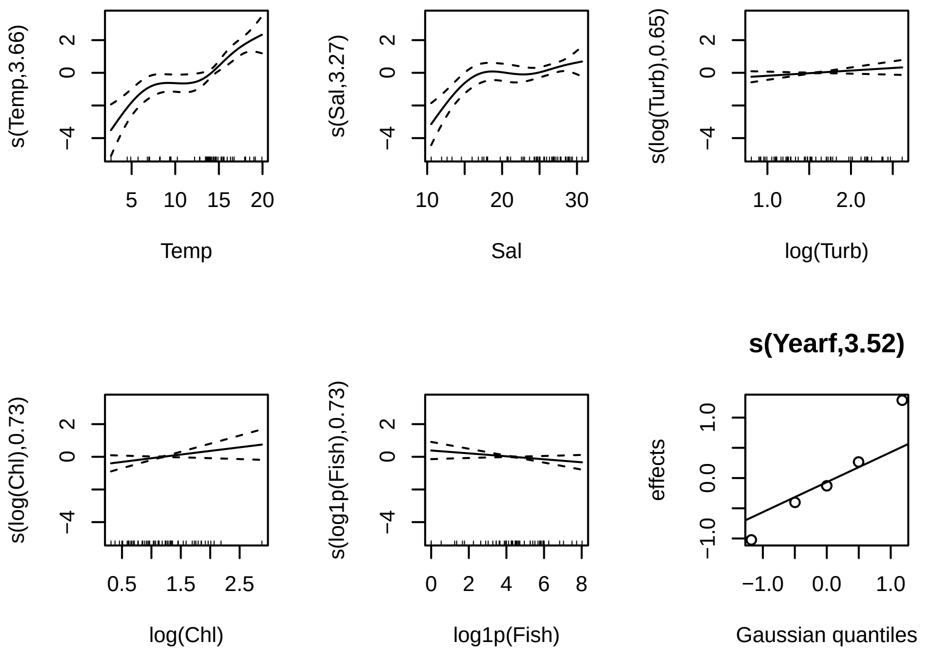
And finally, output the model results. I can do that in a "for" loop, but it's Awkward to look through a long list of output, so I step through each species in turn.

Acartia

```
spp = 'Acartia'
mod <- spp_analysis$gam_mods[spp_analysis$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
```

Plot GAM

```
oldpar <- par(mfrow = c(2,3))
plot(mod)
```



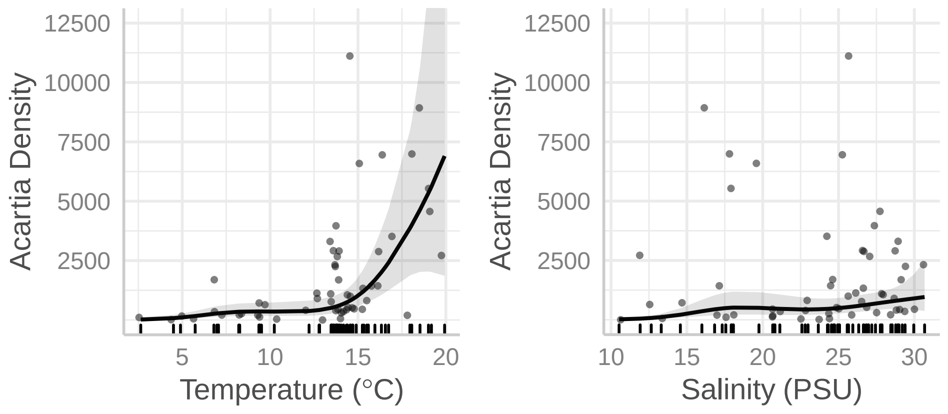
```
par(oldpar)
```

Combined Graphic

```
ggp1 <- ggemmeans(mod, terms = 'Temp')
#> Model has log-transformed response. Back-transforming predictions to original response scale. Standa
ggp2 <- ggemmeans(mod, terms = 'Sal')
#> Model has log-transformed response. Back-transforming predictions to original response scale. Standa
```

```
plt1 <- plot(ggp1, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  ggtitle('') +
  xlab(expression("Temperature (" * degree * "C)")) +
  ylab('Acartia Density') +
  scale_y_continuous(breaks = c(1:5*2500)) +
  coord_cartesian(ylim = c(0, 12500))
#> Scale for 'y' is already present. Adding another scale for 'y', which will
#> replace the existing scale.

plt2 <- plot(ggp2, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  xlab(expression("Salinity" ~ "(PSU)")) +
  ylab('Acartia Density') +
  ggtitle('') +
  scale_y_continuous(breaks = c(1:5*2500)) +
  coord_cartesian(ylim = c(0, 12500))
#> Scale for 'y' is already present. Adding another scale for 'y', which will
#> replace the existing scale.

plt3 <- grid.arrange(plt1, plt2, # alternatives here from cowplot and other pacj=hages,
             ncol = 2, nrow = 1)
```

Version 1

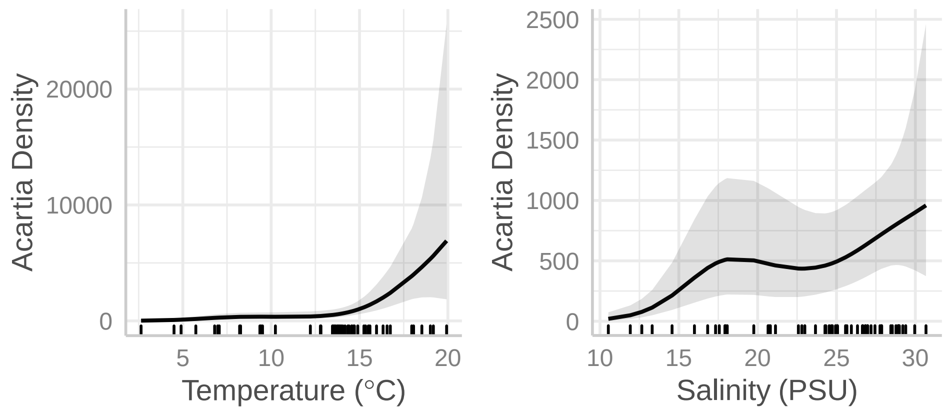


```
#plt3
```

```
grphc <- arrangeGrob(plt1, plt2, ncol = 2, nrow = 1) #generates graphic

ggsave(file='figures/Acartia v1.png', grphc, type='cairo',
         width = 5, height = 2.5)
ggsave('figures/Acartia v1.pdf', grphc, device = cairo_pdf,
       width = 5, height = 2.5)
```

```


plt1 <- plot(ggp1, add.data  = FALSE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  ggtitle('') +
  xlab(expression("Temperature (" * degree * "C)")) +
  ylab('Acartia Density')

plt2 <- plot(ggp2, add.data  = FALSE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  xlab(expression("Salinity" ~ "(PSU)")) +
  ylab('Acartia Density') +
  ggtitle('')

plt3 <- grid.arrange(plt1, plt2, # alternatives here from cowplot and other pacj=hages,
             ncol = 2, nrow = 1)
```

Version 2



```
#plt3
```

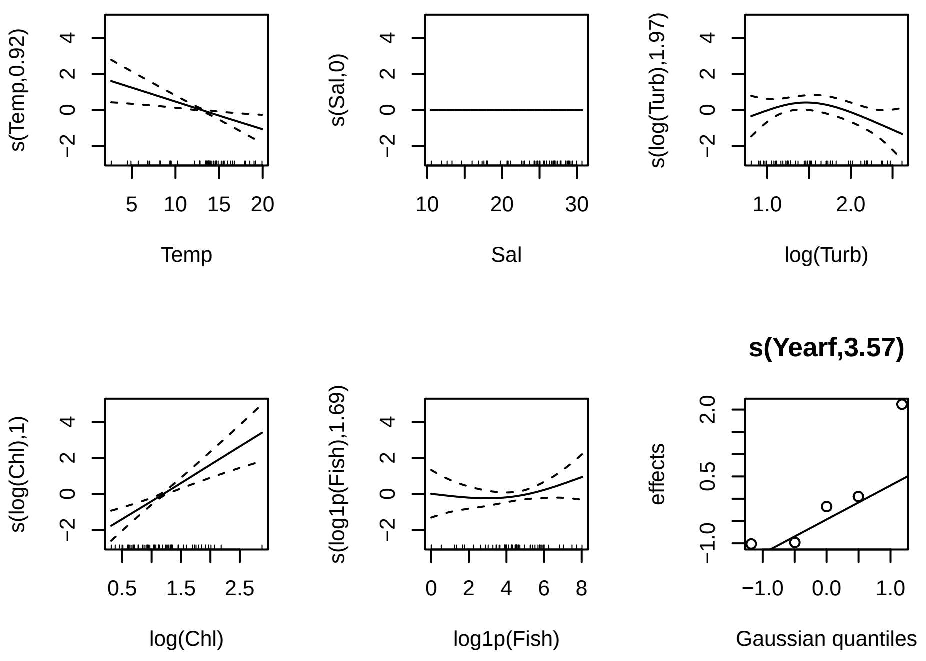
```
grphc <- arrangeGrob(plt1, plt2, ncol = 2, nrow = 1) #generates graphic

ggsave(file='figures/Acartia v2.png', grphc, type='cairo',
         width = 5, height = 2.5)
ggsave('figures/Acartia v2.pdf', grphc, device = cairo_pdf,
       width = 5, height = 2.5)
```

Balanus

```
spp = 'Balanus'
mod <- spp_analysis$gam_mods[spp_analysis$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
```

Plot GAM

```
oldpar <- par(mfrow = c(2,3))
plot(mod)
```



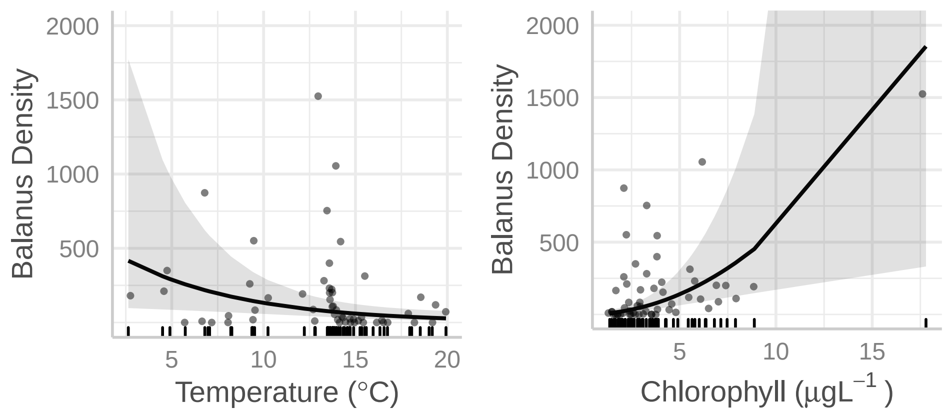
```
par(oldpar)
```

Combined Graphic

```
ggp1 <- ggemmeans(mod, terms = 'Temp')
#> Model has log-transformed response. Back-transforming predictions to original response scale. Standa
ggp2 <- ggemmeans(mod, terms = 'Chl')
#> Model has log-transformed response. Back-transforming predictions to original response scale. Standa
```

```
plt1 <- plot(ggp1, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  ggtitle('') +
  xlab(expression("Temperature (" * degree * "C)")) +
  ylab('Balanus Density') +
  scale_y_continuous(breaks = c(1:4*500)) +
  coord_cartesian(ylim = c(0, 2000))
#> Scale for 'y' is already present. Adding another scale for 'y', which will
#> replace the existing scale.

plt2 <- plot(ggp2, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  xlab(expression("Chlorophyll (" * mu * g * L ^-1 ~")")) +
  ylab('Balanus Density') +
  ggtitle('') +
  scale_y_continuous(breaks = c(1:4*500)) +
  coord_cartesian(ylim = c(0, 2000))
#> Scale for 'y' is already present. Adding another scale for 'y', which will
#> replace the existing scale.

plt3 <- grid.arrange(plt1, plt2, # alternatives here from cowplot and other pacj=hages,
             ncol = 2, nrow = 1)
```

Version 1



```
#plt3
```

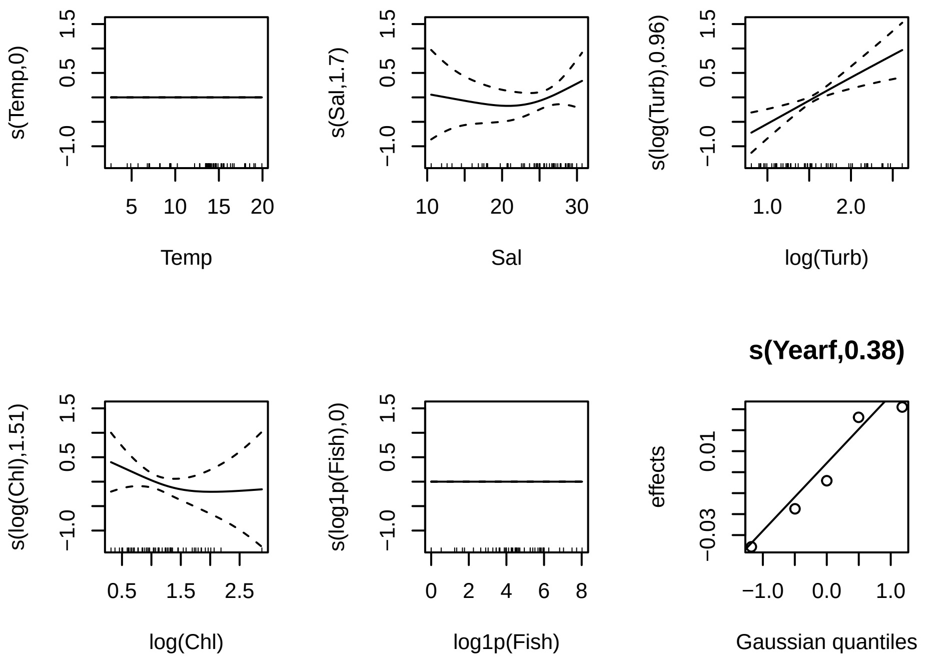
```
grphc <- arrangeGrob(plt1, plt2, ncol = 2, nrow = 1) #generates graphic

ggsave(file='figures/Balanus v1.png', grphc, type='cairo',
         width = 5, height = 2.5)
ggsave('figures/Balanus v1.pdf', grphc, device = cairo_pdf,
       width = 5, height = 2.5)
```

Eurytemora

```
spp =  "Eurytemora"
mod <- spp_analysis$gam_mods[spp_analysis$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
```

Plot GAM

```
oldpar <- par(mfrow = c(2,3))
plot(mod)
```

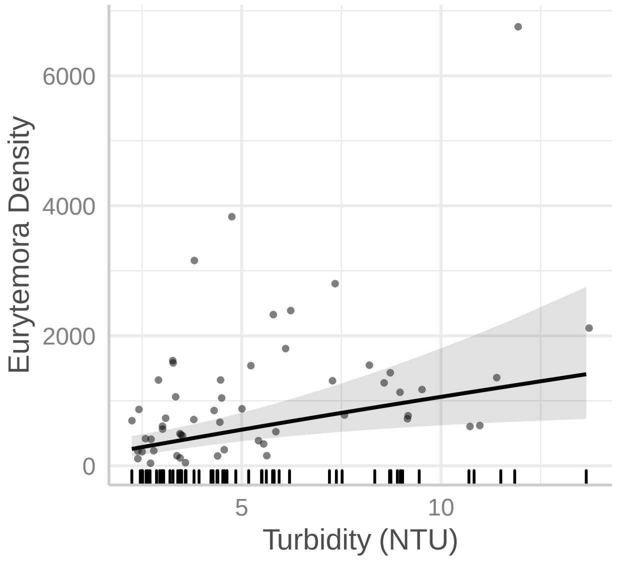


```
par(oldpar)
```

Requesed gRaphic

```
ggp1 <- ggemmeans(mod, terms = 'Turb')
#> Model has log-transformed response. Back-transforming predictions to original response scale. Standa
```

```
plt1 <- plot(ggp1, add.data  = TRUE,
     dot.alpha = 0.5, dot.size = 1) +
  geom_rug(aes(y = NULL)) +
  ggtitle('') +
  xlab(expression("Turbidity" ~ "(NTU)")) +
  ylab('Eurytemora Density')
plt1
```



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