Predictors in SPace And Time Graphic Ideas

Curtis C. Bohlen, Casco Bay Estuary Partnership

1/30/2022

Contents

Introduction

A central challenge of making sense of ecological processes complex estuary environments is that conditions in the estuary vary in complex ways in time and space. Mixing of fresh and salt water creates steep three dimensional ecotones of salinity, temperature, and water density. Those ecotones drive local hydrodynamics that can concentrate nutrients or particulates in certain times or places, while diluting them or washing them out of the estuary elsewhere. The ecotones themselves shape the spatio-temporal structure of other "predictors" of zooplankton abundance

In this notebook, I look at ideas for graphically showing the spatio-temporal structure of predictors used in the paper.

A Side Note

Depth-related data from vertical profiles, especially of salinity and temperature (and thus water density) would provide another way of looking at the spatio-temporal structure of the data that is not available to me based on the data I already have.

I wrote a small R package, `tdggraph`, available via the CBEP GitHub page to facilitate making "interpolated" graphics based on vertical profiles. It's still slightly buggy, but it mmay help. . . .

Load Libraries

```
library(tidyverse)
library(readxl)
library(mgcv)      # for GAM models
library(emmeans)   # For extracting useful "marginal" model summaries

theme_set(theme_classic())
```

Load Data

Folder References

I often use folder references to allow limited indirection, thus making code from GitHub repositories more likely to run "out of the box".

```
data_folder <- "Original_Data"
```

Load Data From Source

```
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
station_data <- station_data %>%
  mutate(station = factor(as.numeric(factor(station))))
```

Subsetting to Desired Data

I base selection of predictor variables here on the ones used for the environmental loading in the community analysis. It's not obvious to me that these are the right indicators. It may be easier to understand patterns in terms of time of year or relative location in the estuary.

```
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'))) %>%
  mutate(season = factor(season, levels = c('Spring', 'Summer', 'Fall'))) %>%
  rename(Season = season,
         Temp = ave_temp_c,
         Sal = ave_sal_psu,
         Turb = sur_turb,
         AvgTurb = ave_turb_ntu,
         DOsat = ave_DO_Saturation,
         Chl = ave_chl_microgperl,
         RH = Herring
         ) %>%
  select(Date, Station, Year, Month, Season, DOY, riv_km, Temp, Sal, Turb, AvgTurb,
         DOsat, Chl, RH,
         combined_density,H, SEI,
         Acartia, Balanus, Eurytemora, Polychaete, Pseudocal, Temora) %>%
  arrange(Date, Station)
head(base_data)
#> # A tibble: 6 x 23
#>   Date                Station  Year Month Season   DOY riv_km  Temp     Sal
#>   <dttm>              <fct>   <dbl> <fct> <fct>  <dbl>  <dbl> <dbl>   <dbl>
#> 1 2013-05-28 00:00:00 1        2013 May   Spring   148  22.6  11.7   0.0200
```

```
#> 2 2013-05-28 00:00:00 2        2013 May   Spring   148  13.9   9.40 14.6
#> 3 2013-05-28 00:00:00 3        2013 May   Spring   148   8.12  6.97 24.7
#> 4 2013-05-28 00:00:00 4        2013 May   Spring   148   2.78  9.51 12.7
#> 5 2013-07-25 00:00:00 1        2013 Jul   Summer   206  22.6  18.5  16.0
#> 6 2013-07-25 00:00:00 2        2013 Jul   Summer   206  13.9  13.6  27.0
#> # ... with 14 more variables: Turb <dbl>, AvgTurb <dbl>, DOsat <dbl>,
#> #   Chl <dbl>, RH <dbl>, combined_density <dbl>, H <dbl>, SEI <dbl>,
#> #   Acartia <dbl>, Balanus <dbl>, Eurytemora <dbl>, Polychaete <dbl>,
#> #   Pseudocal <dbl>, Temora <dbl>
```

```
rm(station_data)
```

"Long" Version of the Data

```
long_data <- base_data %>%
  select(-c(H:Temora,  AvgTurb, DOsat)) %>%
  pivot_longer(-c(Date:riv_km, combined_density),
               names_to = 'Predictor', values_to = 'Values') %>%
  mutate(Predictor = factor(Predictor,
                            levels = c('Temp', 'Sal', 'Turb',
                                       'Chl', 'RH'),
                            labels = c("Temperature", "Salinity", "Turbidity",
                                       "Chlorophyll", "River Herring")))
```

Add Fancy Labels for Figures

These fancy labels are expressions, which allows me to include non-standard glyphs, greek letters, superscripts, etc.

```
plain_labels = levels = c("Temperature",
                          "Salinity",
                          "Turbidity",
                          "Chlorophyll",
                          "River Herring")

fancy_labels =  c(expression("Temperature (" * degree * "C)"),
                  expression("Salinity" ~ "(PSU)"),
                  expression("Turbidity" ~ "(NTU)"),
                  expression("Chlorophyll (" * mu * g * L ^-1 ~")"),
                  expression("River Herring" ~  "(CPUE)"))

fancy_labels_2 =  c(expression("Temperature (" * degree * "C)"),
                    expression("Salinity" ~ "(PSU)"),
                    expression("Log(Turbidity)" ~ "(NTU)"),
                    expression("Log(Chlorophyll) (" * mu * g * L ^-1 ~")"),
                    expression("Log(1 + River Herring)" ~  "(CPUE)"))

long_data <- long_data %>%
  mutate(fancy_p = factor(Predictor, levels = plain_labels,
                          labels = fancy_labels))
```

Add Transformed Predictors.

```
new_values <- base_data %>%
  select(-c(H:Temora,  AvgTurb, DOsat)) %>%
  mutate(Turb = log(Turb),
         Chl = log(Chl),
         RH = log1p(RH)) %>%
  pivot_longer(-c(Date:riv_km, combined_density),
               names_to = 'Predictor', values_to = 'Values_2') %>%
  pull('Values_2')

long_data <- long_data %>%
  mutate(Transformed_Values = new_values,
         fancy_p_2 = factor(Predictor, levels = plain_labels,
                            labels = fancy_labels_2))
```

Add Mean River Kilometers

We will want a variable that replaces Site with the mean or median river km associated with that site, for graphing purposes.

```
lookup <- base_data %>%
  group_by(Station) %>%
  summarize(mean_riv_km = round(mean(riv_km, na.rm = TRUE),1))

base_data <- base_data %>%
  left_join(lookup, by = 'Station')

long_data <- long_data %>%
  left_join(lookup, by = 'Station')
```

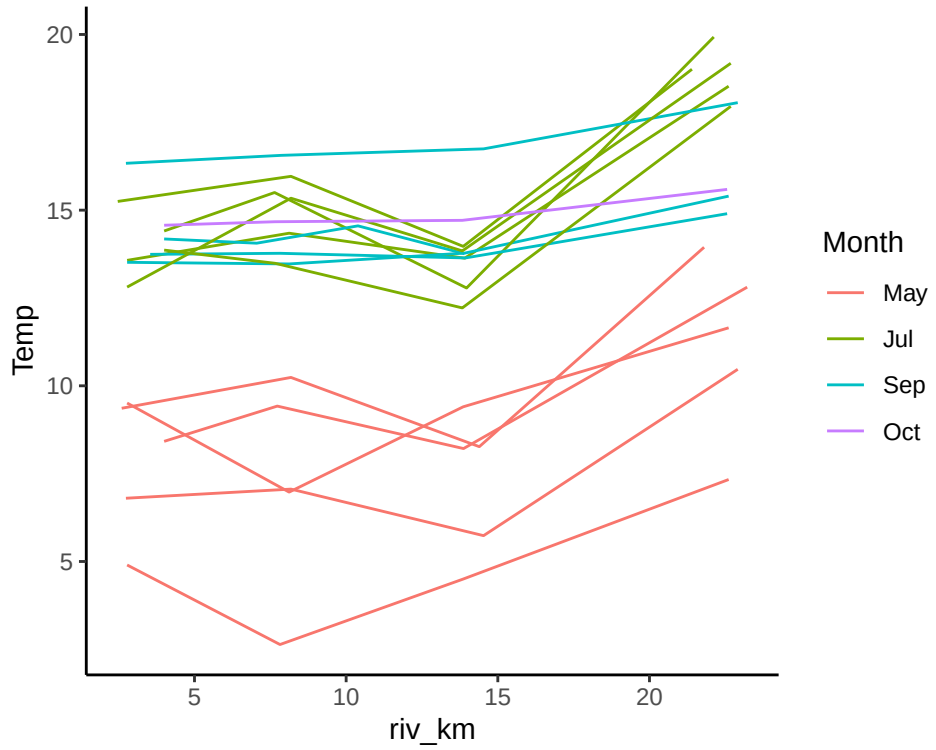
Longitudinal "Slices" Down Estuary

Three Possible Base Graphic Designs

I should be more selective about colors, legends, labels, etc. These preliminary graphics are just to get a sense of each layout.

Longitudinal Traces for Each Day

```
ggplot(base_data, aes(riv_km, Temp, group = factor(Date))) +
  geom_line(aes(color = Month))
```

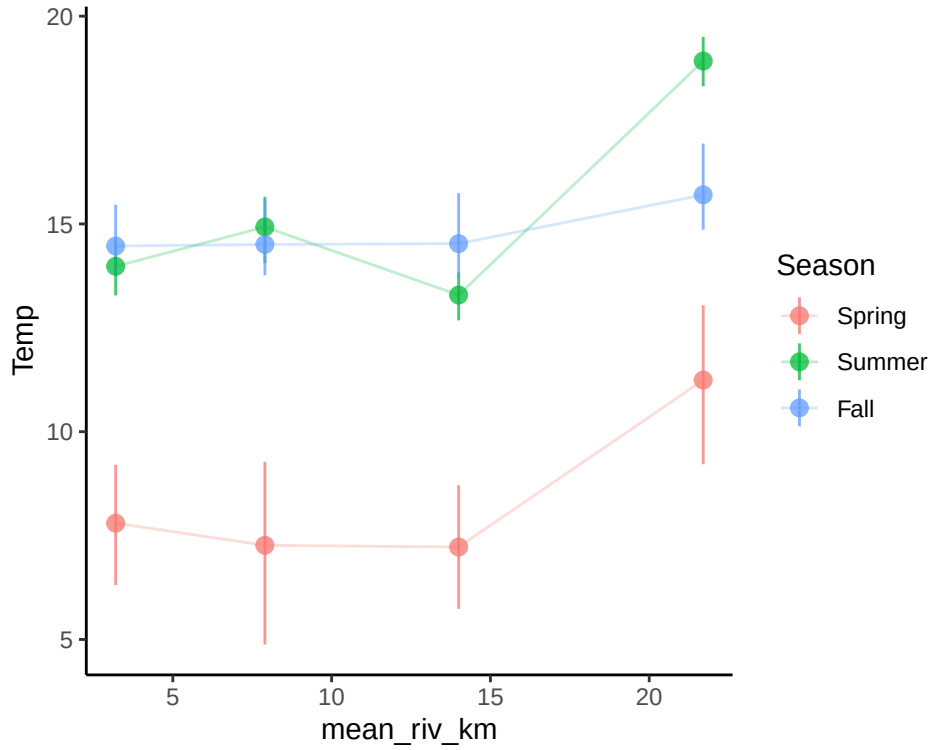


Means by Month and Station

We "recast" Station by the mean river kilometer of that station's data. This shows means plus bootstrapped 95% confidence intervals. An alternative with means and standard errors is also possible. I selected the bootstrapped 95% confidence intervals because they are likely to work even with highly skewed data.

(It occurs to me that I could add "raw" data behind the means and confidence intervals.)

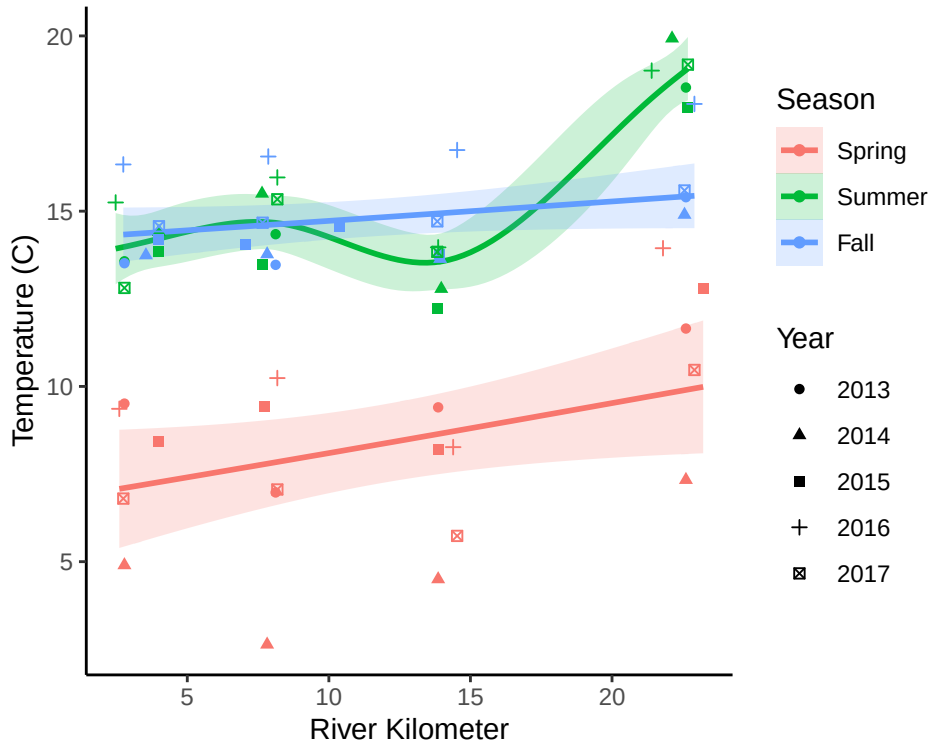
```
ggplot(base_data, aes(mean_riv_km, Temp)) +
    stat_summary(geom = 'line',
                 aes(color = Season),
                 fun = ,
                 alpha = 0.25) +
  stat_summary(aes(color = Season),
               fun.data = mean_cl_boot,
               alpha = 0.75)
#> No summary function supplied, defaulting to `mean_se()`
```



Smoothers

This one could be simplified by suppressing some of the information, such as the year of each observation.

```
ggplot(base_data, aes(riv_km, Temp, color = Season)) +
  geom_smooth(mapping = aes(fill = Season),
              method = 'gam',
              formula = y ~ s(x, bs = "ts"),
              alpha = 0.2) +
  geom_point((aes(shape = factor(Year)))) +

  xlab('River Kilometer') +
  ylab('Temperature (C)') +

  scale_shape(name = 'Year') +

  # All the following is unnecessary here, since  these are mostly defaults, but
  # I Include them as reminders of how to control the legend.
  theme(legend.position = 'right',
        legend.box = 'vertical',
        legend.title.align = 0) +
  guides(color = guide_legend(title.position = "top"),
         shape = guide_legend(title.position = "top"))
```



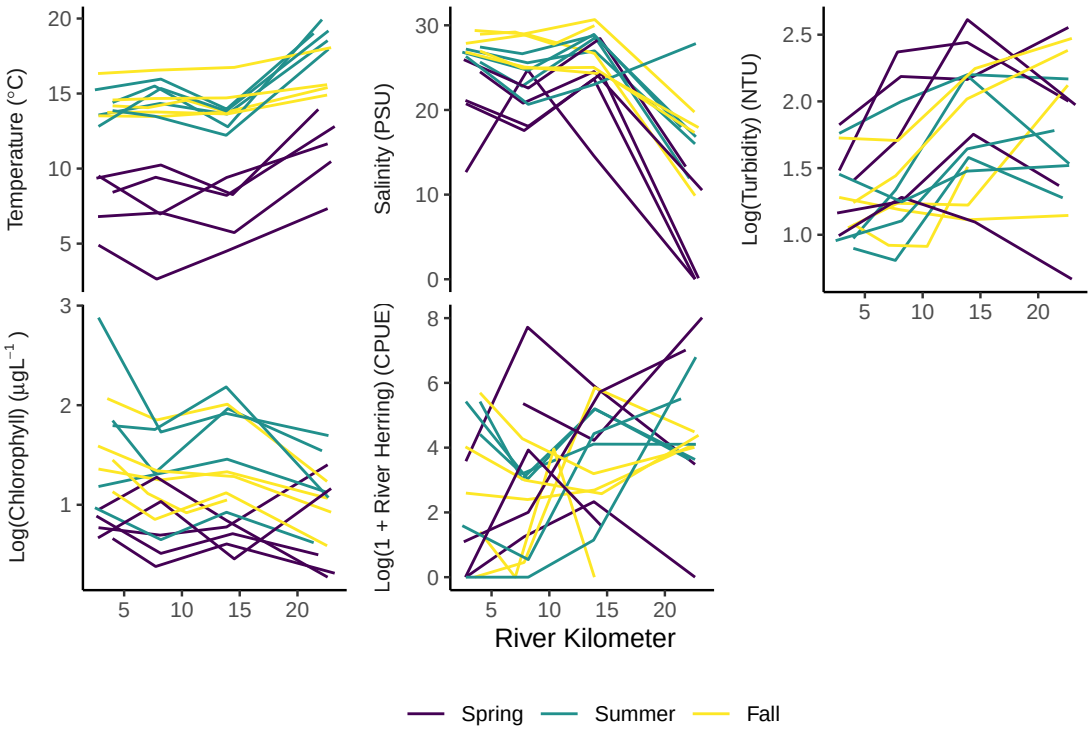
Using `facet_wrap()` to Show Several Predictors

These could all be reshaped into five graphics vertically.

All Longitudinal Profiles, Transformed Values

The vertical axis labels are now quite long, so I had to shrink the fonts.

```
ggplot(long_data, aes(riv_km, Transformed_Values, group = factor(Date))) +
  geom_line(aes(color = Season)) +
  theme_classic(base_size = 9.5) +
  theme(legend.position="bottom",
        strip.placement = 'outside',
        strip.background = element_blank()) +
  scale_color_viridis_d(name = '') +
  facet_wrap(~fancy_p_2,
             scales = 'free_y',
             nrow = 2,
             strip.position = 'left',
             labeller = label_parsed) +
  ylab('') +
  xlab('River Kilometer')
#> Warning: Removed 1 row(s) containing missing values (geom_path).
```



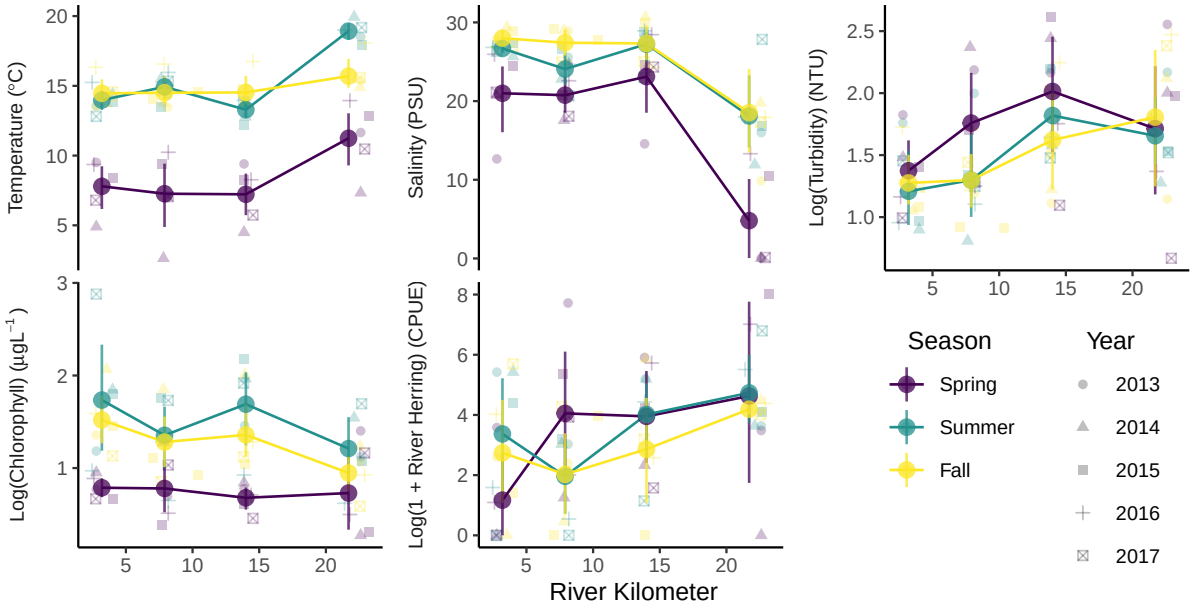
I find that too busy.

All Monthly Summaries (on Transformed Predictors)

Again, the location is the mean river kilometers of all sampling events associated with each Station. The error bars are 95% bootstrap confidence intervals.

```
ggplot(long_data, aes(mean_riv_km, Transformed_Values)) +
  geom_point(aes(x = riv_km, y = Transformed_Values,
                 shape = factor(Year), color = Season),
             alpha = 0.25) +
  stat_summary(geom = 'line',
               aes(color = Season),
               fun=mean) +
  stat_summary(aes(color = Season),
               fun.data = mean_cl_boot,  # could also try mean_se
               alpha = 0.75) +

  facet_wrap(~fancy_p_2,
             scales = 'free_y',
             nrow = 2,
             strip.position = 'left',
             labeller = label_parsed) +
  ylab('') +
  xlab('River Kilometer') +

  scale_shape(name = 'Year') +
  scale_color_viridis_d(name = 'Season') +
```

```
  theme_classic(base_size = 9.5) +
  theme(strip.placement = 'outside',
        strip.background = element_blank(),
        legend.position = c(.85, .175),
        legend.box = 'horizontal',
        legend.title.align = 0.5,)
#> Warning: Removed 2 rows containing non-finite values (stat_summary).

#> Warning: Removed 2 rows containing non-finite values (stat_summary).
#> Warning: Removed 2 rows containing missing values (geom_point).
```



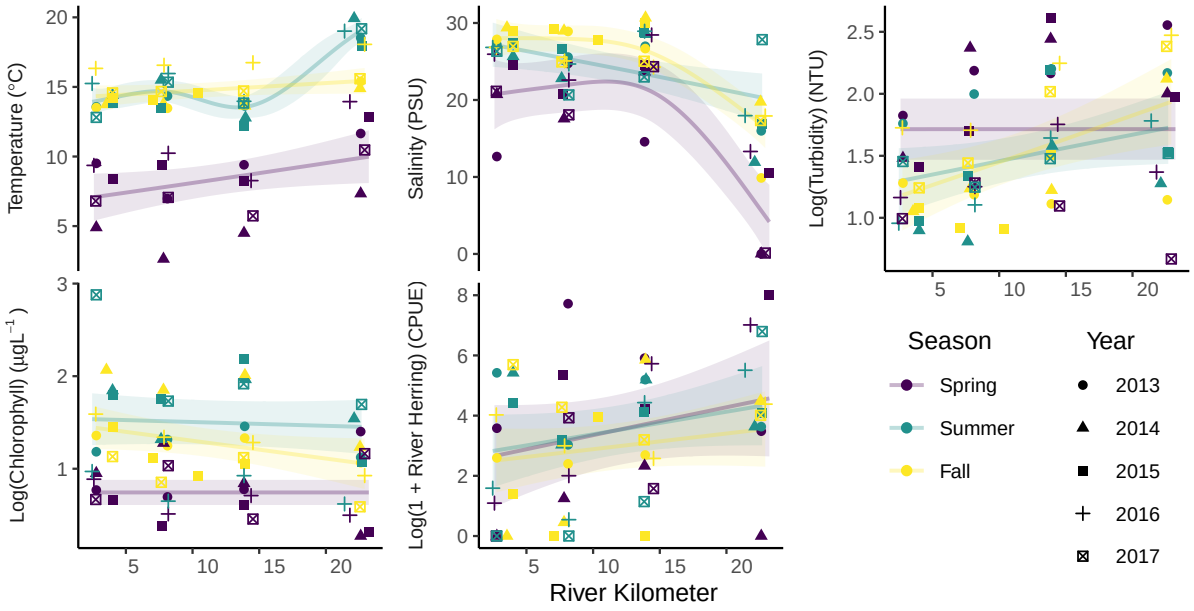
All Smoothers

This is a big, complex graphic, with many custom adjustments...

1. We have to split geom_smooth() into parts so we can independently control transparency of line and background. I am controlling the smoother carefully, by specifying the nature of the smooth, here using a "GAM" smoother, based on a thin plate spline, which tends to fit a fairly conservative.
2. I show both the smoothing line and the 95% confidence band around each line. That feels like it is maybe too much.
3. There are a lot of "custom" features here, mostly regarding layout, like whether to include alegend, what to call it, and how to label the "facets".

```
ggplot(long_data, aes(riv_km, Transformed_Values, color = Season)) +

  # First the error bands -- perhaps too much
  stat_smooth(geom = 'ribbon',
              mapping= aes(fill = Season),
```

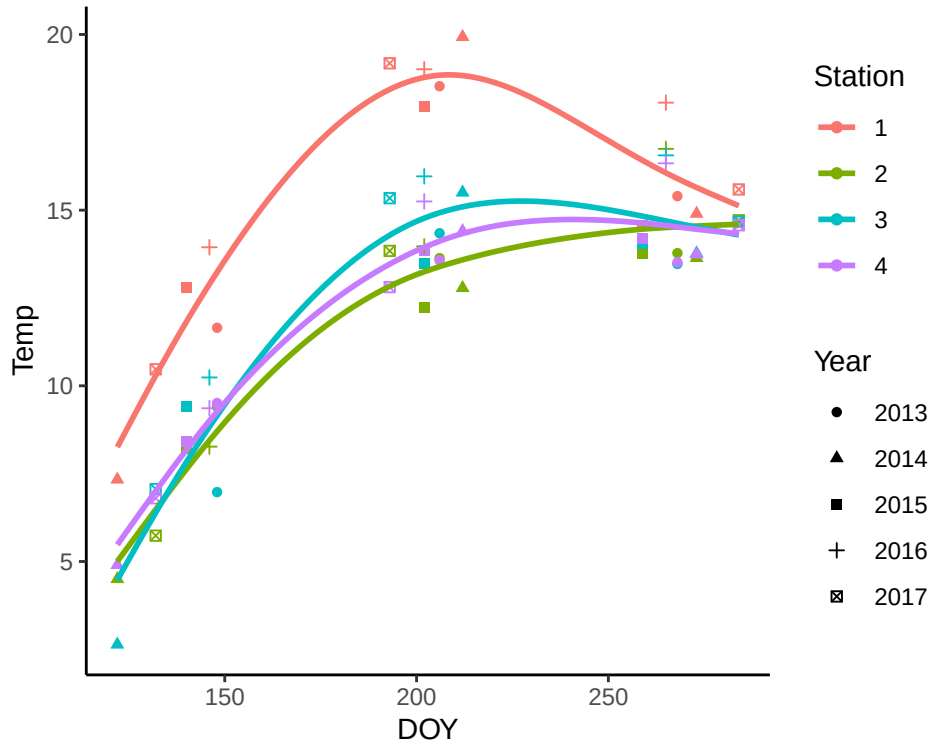
```
              method = 'gam',
              formula = y ~ s(x, bs = "ts"),
              alpha = 0.1,
              color = NA) +
  # Then the line smoother
  stat_smooth(geom = 'line',
              method = 'gam',
              formula = y ~ s(x, bs = "ts"),
              alpha = 0.3,
              size = .75) +
  # Add points ON TOP of the smoothers, to make them more visble
  geom_point((aes(shape = factor(Year)))) +

  facet_wrap(~fancy_p_2,
             scales = 'free_y',
             nrow = 2,
             strip.position = 'left',  # So they look like Y axis labels
             labeller = label_parsed) +  # To read formulae
  ylab('') +
  xlab('River Kilometer') +

  # Now, control all of the scales used, setting names, colors, and labeling
  scale_shape(name = 'Year') +
  scale_color_viridis_d(name = 'Season') +
  scale_fill_viridis_d(name = 'SSeason') +

  # And chose where to show the legends
  guides(color = guide_legend(title.position = "top"),
         fill = 'none',
         shape = guide_legend(title.position = "top")) +

  # Make the fonts slightly smaller than the default
  theme_classic(base_size = 9.5) +

  # and control various details
  theme( strip.placement = 'outside',          # like axis label
         strip.background = element_blank(),  # aesthetic choice
         legend.position = c(.85, .175),      # in the "blank" space
         legend.box = 'horizontal',           # legends side by side
         legend.title.align = 0.5,            # Centered over legend
         axis.text.x = element_text(hjust = -.2))  # Put labels right of ticks
#> Warning: Removed 2 rows containing non-finite values (stat_smooth).

#> Warning: Removed 2 rows containing non-finite values (stat_smooth).
#> Warning: Removed 2 rows containing missing values (geom_point).
```



Temporal "Slices" at Each Station

Initial Concept

```
ggplot(base_data, aes(DOY, Temp, color = Station)) +
  geom_point((aes(shape = factor(Year)))) +
  geom_smooth(method = 'gam', formula = y ~ s(x, bs = "cs"), se = FALSE) +
  scale_shape(name = 'Year')
```



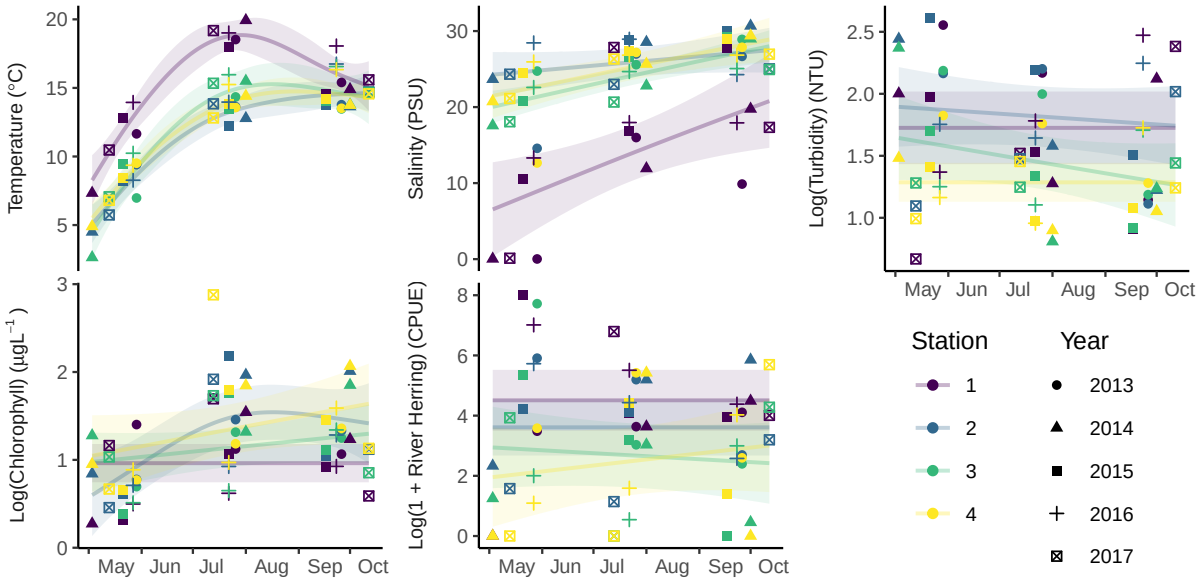
All Temporal Slices

To make this look good, I need to create a month by month axis for Days of the Year by Month.

```
days = c(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
cum_days = c(0, cumsum(days))[1:12]
names(cum_days) <- month.abb
cum_days
#> Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
#>   0  31  59  90 120 151 181 212 243 273 304 334
```

```
ggplot(long_data, aes(DOY, Transformed_Values, color = Station)) +

  # First the error bands -- perhaps too much
  stat_smooth(geom = 'ribbon',
              mapping= aes(fill = Station),
              method = 'gam',
              formula = y ~ s(x, bs = "ts"),
              alpha = 0.1,
              color = NA) +
  # Then the line smoother
  stat_smooth(geom = 'line',
              method = 'gam',
              formula = y ~ s(x, bs = "ts"),
              alpha = 0.3,
              size = .75) +
  # Add points ON TOP of the smoothers, to make them more visible
  geom_point((aes(shape = factor(Year)))) +
```

```
  facet_wrap(~fancy_p_2,
             scales = 'free_y',
             nrow = 2,
             strip.position = 'left',
             labeller = label_parsed) +
  ylab('') +
  xlab('') +

  # Now, control all of the scales used, setting names, colors, and labeling
  scale_shape(name = 'Year') +
  scale_color_viridis_d(name = 'Station') +  #viridis palettes are colorblind friendly
  scale_fill_viridis_d(name = 'Station') +
  scale_x_continuous(breaks = cum_days, labels=month.abb) +

  # And chose legends, where to put names
  # you can do many other things with a call to `guides()`
  guides(color = guide_legend(title.position = "top"),
         fill = 'none',
         shape = guide_legend(title.position = "top")) +

  # Make the fonts slightly smaller than the default
  theme_classic(base_size = 9.5) +

  # and control various details
  theme( strip.placement = 'outside',          # Outside of axis, like axis label
         strip.background = element_blank(),  # An aesthetic choice
         legend.position = c(.85, .175),      # Located in the "blank" area
         legend.box = 'horizontal',           # Place two legends side by side
         legend.title.align = 0.5,            # Centered over the legend
         axis.text.x = element_text(hjust = -.25))  #Put labels right of ticks.
#> Warning: Removed 2 rows containing non-finite values (stat_smooth).

#> Warning: Removed 2 rows containing non-finite values (stat_smooth).
#> Warning: Removed 2 rows containing missing values (geom_point).
```


Temperature (°C)
Salinity (PSU)
Log(Turbidity) (NTU)
Log(Chlorophyll) (µgL^{-1})
Log(1 + River Herring) (CPUE)
May
Jun
Jul
Aug
Sep
Oct
Station
1
2
3
4
Year
2013
2014
2015
2016
2017