Graphic for Showing Estuary Conditions in Space And Time

Curtis C. Bohlen, Casco Bay Estuary Partnership

6/21/2022

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 both "predictors" and "responses".

In this notebook, I look at ideas for graphically depicting the spatio-temporal structure of all measured variable. Other notebooks examine the use of sequential models and path models for understanding estuary structure. Here, I focus on graphic exposition only, using R and `ggplot2` as the graphics engine.

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

```
data_folder <- "Original_Data"

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

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.

```
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')),
         Station =factor(Station),
         season = factor(season, levels = c('Spring', 'Summer', 'Fall')),
         Yearf = factor(Year)) %>%
  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,
         Fish = `___61`,
         Zoopl = combined_density,
         Diversity = H
         ) %>%
  select(Date, Station, Year, Yearf, Month, Season, DOY, riv_km, Temp, Sal, Turb, AvgTurb,
         DOsat, Chl, Fish,
         Zoopl, Diversity, SEI,
         Acartia, Balanus, Eurytemora, Polychaete, Pseudocal, Temora) %>%
  arrange(Date, Station)
head(base_data)
#> # A tibble: 6 x 24
#>   Date                Station  Year Yearf Month Season   DOY riv_km  Temp
#>   <dttm>              <fct>   <dbl> <fct> <fct> <fct>  <dbl>  <dbl> <dbl>
#> 1 2013-05-28 00:00:00 1        2013 2013  May   Spring   148  22.6  11.7
#> 2 2013-05-28 00:00:00 2        2013 2013  May   Spring   148  13.9   9.40
#> 3 2013-05-28 00:00:00 3        2013 2013  May   Spring   148   8.12  6.97
#> 4 2013-05-28 00:00:00 4        2013 2013  May   Spring   148   2.78  9.51
#> 5 2013-07-25 00:00:00 1        2013 2013  Jul   Summer   206  22.6  18.5
#> 6 2013-07-25 00:00:00 2        2013 2013  Jul   Summer   206  13.9  13.6
#> # ... with 15 more variables: Sal <dbl>, Turb <dbl>, AvgTurb <dbl>,
#> #   DOsat <dbl>, Chl <dbl>, Fish <dbl>, Zoopl <dbl>, Diversity <dbl>,
#> #   SEI <dbl>, Acartia <dbl>, Balanus <dbl>, Eurytemora <dbl>,
#> #   Polychaete <dbl>, Pseudocal <dbl>, Temora <dbl>
```

```
rm(station_data)
```

"Long" Version of the Data

We are headed towards graphics using `ggplot2`'s "faceting" capabilities. But `facet_wrap()` is designed for use by splitting a data column into facets based o. the value of a factor, while `facet_grid()` requires two factors. That means we have to reshape the data into a "long" format that holds value for the Y axis in one column, with a second column that records which variable is listed.

```
long_data <- base_data %>%
  select(-c(SEI:Temora,  AvgTurb)) %>%
  pivot_longer(-c(Date:riv_km),
               names_to = 'Variable', values_to = 'Values') %>%
  mutate(Variable = factor(Variable,
                           levels = c('Temp', 'Sal', 'Turb',
                                      'Chl', 'DOsat', 'Fish',
                                      'Zoopl', "Diversity"),
                           labels = c("Temperature", "Salinity", "Turbidity",
                                      "Chlorophyll", "Oxygen Saturation",
                                      "Fish Density", "Zooplankton",
                                      "Diversity")))
```

Add Fancy Labels for Figures

These fancy labels are expressions, which allows me to include non-standard glyphs, Greek letters, superscripts, etc. The "plain labels" have to match variable names, but the other two labels are for different possible graphics

```
plain_labels = levels = c("Temperature",
                          "Salinity",
                          "Turbidity",
                          "Chlorophyll",
                          "Oxygen Saturation",
                          "Fish Density",
                          "Zooplankton",
                          "Diversity")

fancy_labels =  c(expression("Temperature (" * degree * "C)"),
                  expression("Salinity" ~ "(PSU)"),
                  expression("Turbidity" ~ "(NTU)"),
                  expression("Chlorophyll (" * mu * g * L ^-1 ~")"),
                  expression("Oxygen Saturation" ~ "(%)"),
                  expression("Fish" ~  "(CPUE)"),
                  expression("Zooplankton (" * m ^-3 ~ ")" ),
                  expression("Diversity" ~ "(H)"))

fancy_labels_2 =  c(expression("Temperature (" * degree * "C)"),
                    expression("Salinity" ~ "(PSU)"),
                    expression("Log(Turbidity)" ~ "(NTU)"),
                    expression("Log(Chlorophyll) (" * mu * g * L ^-1 ~")"),
                    expression("Oxygen Saturation" ~ "(%)"),
                    expression("Log(1 + Fish)" ~  "(CPUE)"),
                    expression("Zooplankton (" * m ^-3 ~ ")" ),
                    expression("Diversity" ~ "(H)"))

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

Add Transformed Predictors

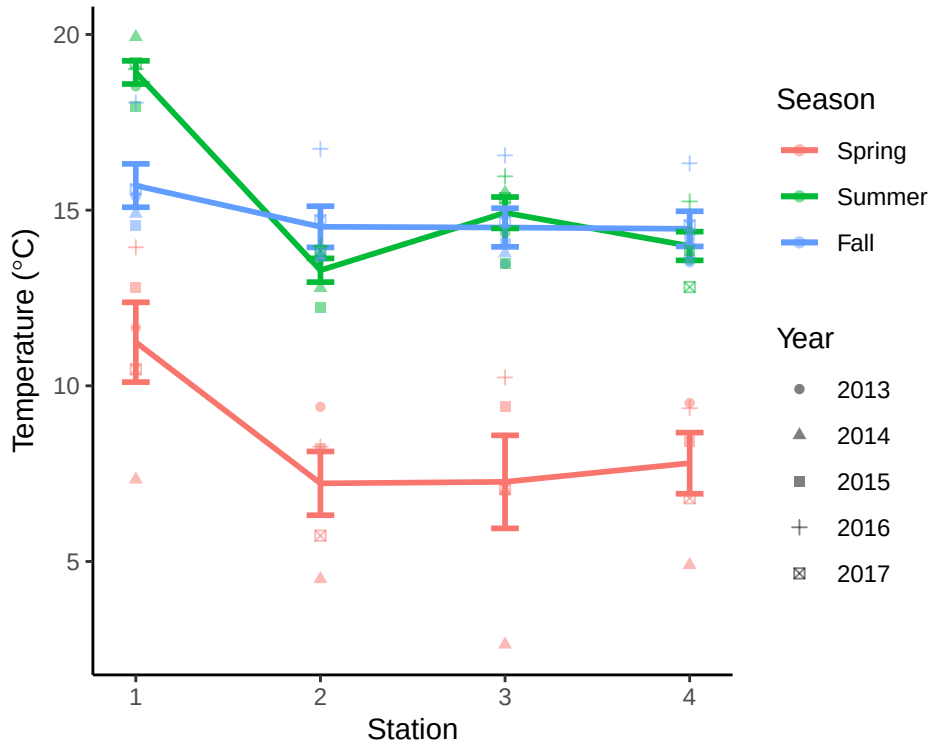
```
new_values <- base_data %>%
  select(-c(SEI:Temora,  AvgTurb)) %>%

  mutate(Turb = log(Turb),
         Chl = log(Chl),
         Fish = log1p(Fish)) %>%
  pivot_longer(-c(Date:riv_km),
               names_to = 'Variable', values_to = 'Values_2') %>%
  pull('Values_2')

long_data <- long_data %>%
  mutate(Transformed_Values = new_values,
         fancy_p_2 = factor(Variable, levels = plain_labels,
                            labels = fancy_labels_2)) %>%
  filter(! is.na(Values))
```

Graphic Development

Base Graphic Design

We represent location only as "Station", not River Kilometer. This has the advantage of placing upstream to the left.

```
ggplot(base_data, aes(as.numeric(Station), Temp, color = Season)) +
  stat_summary(geom = 'line', lwd = 1) +
  stat_summary(geom = 'errorbar', width = 0.15, lwd = 1) +
  geom_point(aes(shape = factor(Year)), alpha = 0.5) +

  xlab('Station') +
  ylab(expression("Temperature (" * degree * "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"))
#> No summary function supplied, defaulting to `mean_se()`
#> No summary function supplied, defaulting to `mean_se()`
```



Using `facet_wrap()` to Show All Variables

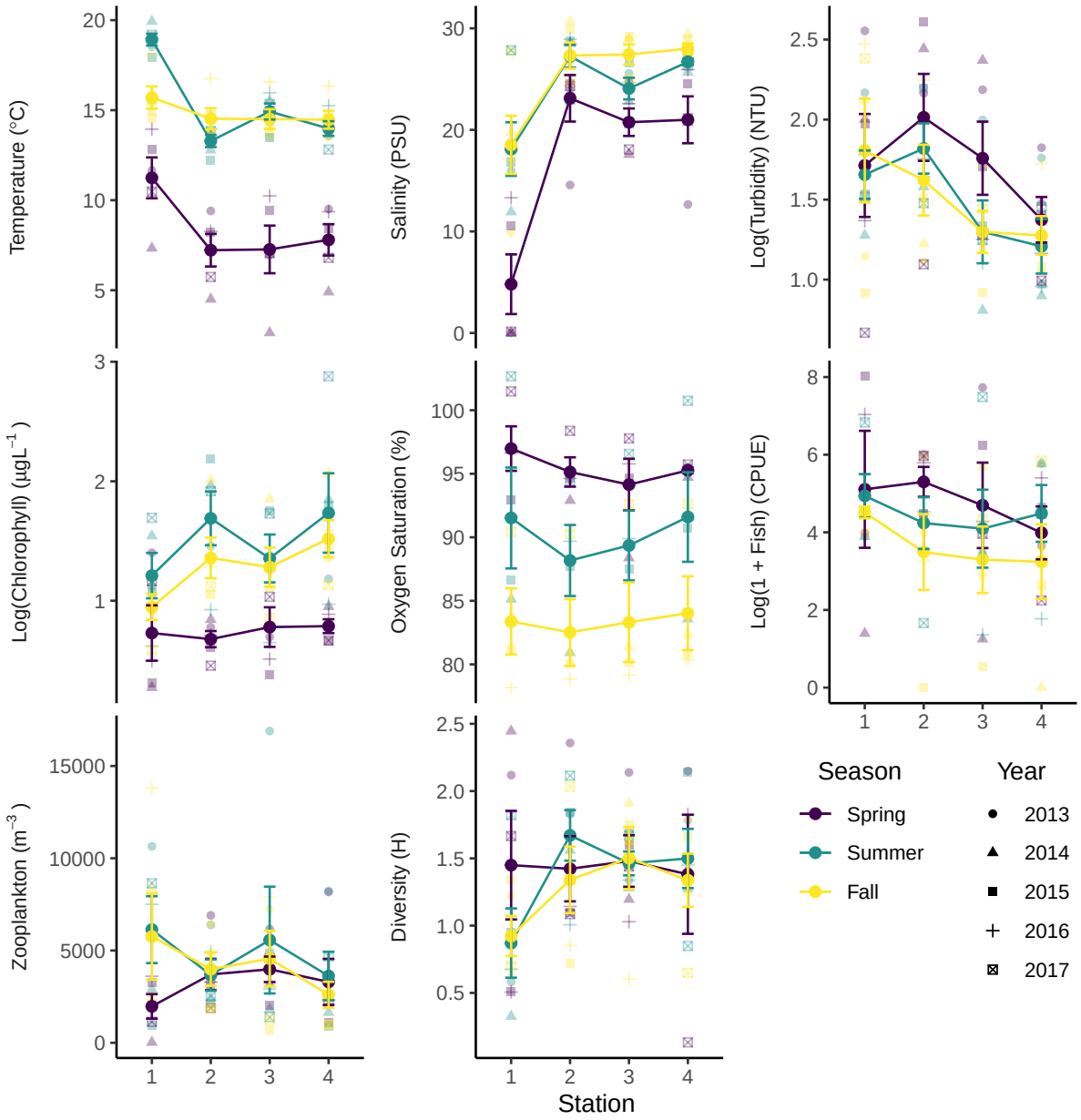
Earlier versions of this graphic were based on bootstrapped 95% confidence intervals, but confidence interval functions (`mean_cl_boot`, `mean_cl_normal`, `mean_sdl`) are giving me an error

"Warning: Computation failed in `stat_summary()`:"

This Graphic provides mean +/- the standard error of the mean..

```
ggplot(long_data, aes(Station, Transformed_Values, color = Season)) +
  geom_point(aes(shape = factor(Year)),
             alpha = 0.35) +
  stat_summary(aes(group = Season), geom = 'line', fun = mean) +
  stat_summary(aes(group = Season), geom = 'point', size = 2, fun = mean) +
  stat_summary(mapping = aes(group = Season), geom = 'errorbar', width = 0.2,
               fun.data = mean_se
               ) +

  facet_wrap(~fancy_p_2,
             scales = 'free_y',
             nrow = 3,
             strip.position = 'left',
             labeller = label_parsed) +
  ylab('') +
  xlab('Station') +

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

```
guides(shape = guide_legend(override.aes = list(alpha = 1 ) ) ) +



theme_classic(base_size = 11) +
theme(strip.placement = 'outside',
      strip.background = element_blank(),
      legend.position = c(.85, .175),
      legend.box = 'horizontal',
      legend.title.align = 0.5,)
```

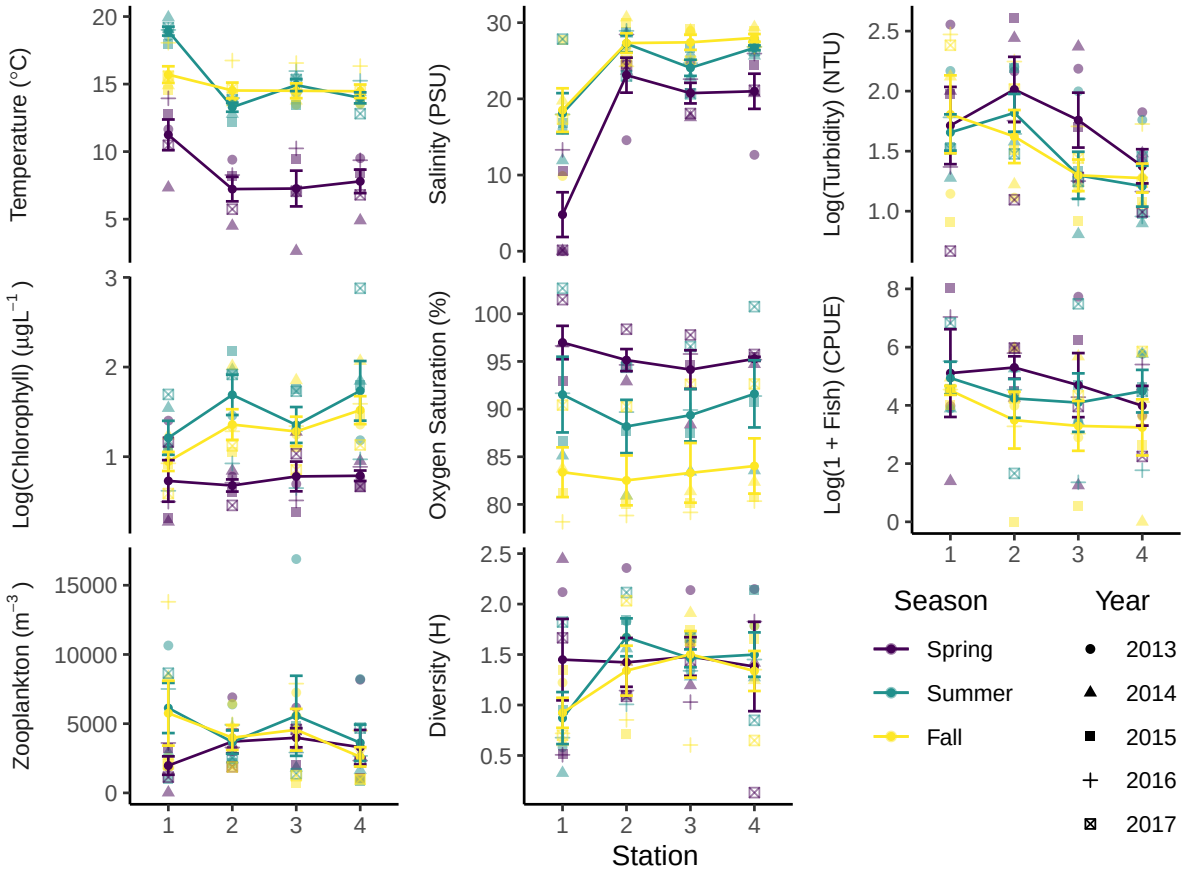


Final Graphic

The instructions to authors suggests figure widths should line up with columns, and proposes figure widths should be: 39, 84, 129, or 174 mm wide, with height not to exceed 235 mm. Presumably that corresponds to 1,2,3,or 4 columns wide?

Unfortunately RMarkdown / `knitr` likes figure dimensions in inches. 174 mm is about 6.85 inches. While a square layout like the previous graphic works, we don't really need so much height, so we can make the graphic slightly shorter

```
ggplot(long_data, aes(Station, Transformed_Values, color = Season)) +
  geom_point(aes(shape = factor(Year)),
             alpha = 0.5) +
  stat_summary(aes(group = Season), geom = 'line', fun = mean) +
  stat_summary(aes(group = Season), geom = 'point', size = 1, fun = mean) +
  stat_summary(mapping = aes(group = Season), geom = 'errorbar', width = 0.2,
               fun.data = mean_se
               ) +

  facet_wrap(~fancy_p_2,
             scales = 'free_y',
             nrow = 3,
             strip.position = 'left',
             labeller = label_parsed) +
  ylab('') +
  xlab('Station') +

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

  guides(shape = guide_legend(override.aes = list(alpha = 1 ) ) ) +


  theme_classic(base_size = 11) +
  theme(strip.placement = 'outside',
        strip.background = element_blank(),
        legend.position = c(.85, .11),
        legend.box = 'horizontal',
        legend.title.align = 0.5,)
```




```
ggsave('figures/env_variables.png', type='cairo',
         width = 6.85, height = 4.9)
ggsave('figures/env_variables.pdf', device = cairo_pdf,
       width = 6.85, height = 4.9)
```