Using GAMs to Analyze Plankton Comunity NMDS Data

Curtis C. Bohlen, Casco Bay Estuary Partnership

5/17/2022

Contents

Introduction

This notebook takes the OUTPUT of the NMDS analyses and looks at how well each synthetic NMDS axis can be predicted based on models akin to what we used to analyze total zooplankton abundance, diversity and individual species.

The flow of analyses is as follows:

1. Conduct the NMDS analysis (mimicking Erin's original NMDS plot)
2. Plot the results color coded to show major relationships with predictors
3. Conduct a linear analysis of relationship to each predictors using the `envfit()` function included in the `vegan` package.
4. Conduct GAM analyses of the synthetic axis scores from the NMDS.

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(vegan)
#> Loading required package: permute
#> Loading required package: lattice
#> This is vegan 2.6-2
```

```
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(emmeans)   # For extracting useful "marginal" model summaries
```

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())
```

Folder References

I 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"
```

Input Data

Environmental Data

This is the data source used by Erin. For our purposes, I believe it is identical, except for the dropped May 2015 sample. Code is retained here in case I need it later.

```
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="NMDS Happy", col_types = c("skip", "date",
                                                             "numeric", "text", "skip",
                                                             "numeric", "skip", "numeric",
                                                             "numeric", "numeric", "numeric",
                                                             "numeric", "numeric", "numeric",
                                                             "numeric", "numeric", "numeric",
                                                             "numeric", "numeric", "numeric",
                                                             "numeric", "numeric", "numeric",
                                                             "numeric", "numeric", "numeric",
                                                             "numeric", "numeric", "numeric",
                                                             "numeric", "numeric", "numeric",
                                                             "numeric", "numeric", "numeric",
                                                             "numeric", "numeric", "numeric",
```

```
                                              "numeric", "numeric", "numeric",
                                              "numeric", "numeric", "numeric",
                                              "numeric", "numeric", "numeric",
                                              "numeric", "numeric", "numeric",
                                              "numeric", "numeric", "numeric",
                                              "numeric", "numeric", "numeric",
                                              "numeric", "numeric", "numeric",
                                              "numeric", "numeric", "text")) %>%
  rename_with(~ gsub(" ", "_", .x)) %>%
  rename_with(~ gsub("\\.", "_", .x))
```

The data read in from a different data sheet ("Final"). This is the data sheet I used for all other analyses. I like to use the same environment data for all different analyses.

```
station_data_2 <- read_excel(file_path,
                             sheet="Final", col_types = c("skip", "date",
                                              "numeric", "text", "skip",
                                              "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)) %>%
  filter(! (station == 8 & month == 'May' & year == 2015))
#> New names:
#> * `` -> `...60`
```

Station names are arbitrary, and Ambrose 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_2 %>%
  mutate(station = factor(as.numeric(factor(station)))) %>%
  mutate(season = case_when(month == 'May' ~ 'Spring',
                            month == 'July' ~ 'Summer',
                            TRUE ~ 'Fall')) %>%
  relocate(season, .after = month) %>%
  relocate(station, .after = season)
```

Here I mostly select the depth-averaged water chemistry parameters, create short names that will work in later analyses and graphics and convert some variables to factors to control later analyses.

```
station_data <- station_data %>%
  rename(Date = date,
         Station = station,
         Year = year) %>%
  select(-c(month)) %>%
  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')),
         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,
         RH = Herring
         ) %>%
  select(Date, Station, Year, Yearf, Month, Season, DOY, riv_km, Temp, Sal, Turb, AvgTurb,
         DOsat, Chl, RH) %>%
  arrange(Date, Station)
head(station_data)
#> # A tibble: 6 x 15
#>   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 6 more variables: Sal <dbl>, Turb <dbl>, AvgTurb <dbl>, DOsat <dbl>,
#> #   Chl <dbl>, RH <dbl>
```

Composition Data

```
filename.in <- "Penobscot_Zooplankton and field data_EA_2.13.20.xlsx"
file_path <- file.path(data_folder, filename.in)
zoopl <- read_excel(file_path,
                    sheet = "NMDS Happy",
                    col_types = c("date",
                                  "text", "numeric", "numeric", "text",
                                  "text", "text", "text", "text", "text",
                                  "text", "numeric", "text", "text",
                                  "numeric", "numeric", "numeric",
                                  "text", "text", "text", "numeric",
                                  "numeric", "numeric", "numeric")) %>%
  select(-c(`...20`:`...24`)) %>%
  rename_with(~ gsub(" ", "_", .x))
```

```
#> New names:
#> * `` -> `...20`
#> * `` -> `...21`
#> * `` -> `...22`
#> * `` -> `...23`
#> * `` -> `...24`
```

We renumber the stations here as well. The code is similar.

```
zoopl <- zoopl %>%
  mutate(STATION = factor(as.numeric(factor(STATION))))
zoopl
#> # A tibble: 814 x 19
#>    DATE                Month  Year STATION PHYLUM CLASS `SUB-CLASS` ORDER FAMILY
#>    <dttm>              <chr> <dbl> <fct>   <chr>  <chr> <chr>       <chr> <chr>
#>  1 2015-09-16 00:00:00 Sept~  2015 4       Arthr~ Maxi~ Copepoda    <NA>  <NA>
#>  2 2014-05-02 00:00:00 May    2014 1       Arthr~ Maxi~ Copepoda    Cala~ <NA>
#>  3 2017-07-12 00:00:00 July   2017 3       Unkno~ <NA>  <NA>        <NA>  <NA>
#>  4 2016-07-20 00:00:00 July   2016 4       Unkno~ <NA>  <NA>        <NA>  <NA>
#>  5 2015-09-16 00:00:00 Sept~  2015 3       Unkno~ <NA>  <NA>        <NA>  <NA>
#>  6 2017-10-11 00:00:00 Octo~  2017 1       Unkno~ Unid~ <NA>        <NA>  <NA>
#>  7 2016-07-20 00:00:00 July   2016 3       Unkno~ <NA>  <NA>        <NA>  <NA>
#>  8 2016-05-25 00:00:00 May    2016 2       Unkno~ <NA>  <NA>        <NA>  <NA>
#>  9 2013-09-25 00:00:00 Sept~  2013 3       Unkno~ <NA>  <NA>        <NA>  <NA>
#> 10 2013-07-25 00:00:00 July   2013 4       Unkno~ <NA>  <NA>        <NA>  <NA>
#> # ... with 804 more rows, and 10 more variables: GENUS <chr>, SPECIES <chr>,
#> #   QUANTITY <dbl>, LOWEST_TAXA <chr>, NAME <chr>, `TOTAL_#_ORGANISMS` <dbl>,
#> #   CORRECTED_PERCENT_ABUNDANCE <dbl>, `NET_MESH_SIZE_(MICRONS)` <dbl>,
#> #   NOTES <chr>, Picture_number <chr>
```

Turn Data from Long to Wide

This code generates a total abundance for each taxa by site and date and pivots it to wide format. The code is more compact that what Erin used, but slightly more opaque because it relies on several options of the `pivot_wider()` function.

```
zoopl2 <- zoopl %>%
  pivot_wider(c(DATE, Month, Year, STATION),
              names_from = NAME,
              names_sort = TRUE,
              values_from = CORRECTED_PERCENT_ABUNDANCE,
  values_fn = sum,
  values_fill = 0)
zoopl2
#> # A tibble: 59 x 53
#>    DATE                Month  Year STATION Acartia Amphipod `Arrow worm` Balanus
#>    <dttm>              <chr> <dbl> <fct>      <dbl>    <dbl>        <dbl>   <dbl>
#>  1 2015-09-16 00:00:00 Sept~  2015 4          43.4   0            0         3.28
#>  2 2014-05-02 00:00:00 May    2014 1           6.85  0            0         3.43
#>  3 2017-07-12 00:00:00 July   2017 3          32.5   0            0.0219   22.6
#>  4 2016-07-20 00:00:00 July   2016 4          49.8   0            0.00463   0.422
#>  5 2015-09-16 00:00:00 Sept~  2015 3          49.1   0.00942      4.96      7.66
```

```
#>  6 2017-10-11 00:00:00 Octo~  2017 1         68.6   0            0         0
#>  7 2016-07-20 00:00:00 July   2016 3         49.3   0            0         0
#>  8 2016-05-25 00:00:00 May    2016 2          6.45  0.00466      0         1.38
#>  9 2013-09-25 00:00:00 Sept~  2013 3         45.6   0            0.00236   3.88
#> 10 2013-07-25 00:00:00 July   2013 4         48.2   0            0         9.16
#> # ... with 49 more rows, and 45 more variables: Bivalve <dbl>,
#> #   `Brittle Star` <dbl>, Bryozoan <dbl>, `Calanoid spp` <dbl>, Calanus <dbl>,
#> #   Caligus <dbl>, Centropages <dbl>, Cladoceran <dbl>, `Crab larvae` <dbl>,
#> #   Crangon <dbl>, Ctenophore <dbl>, Cumacean <dbl>, Decapod <dbl>,
#> #   Diacyclops <dbl>, Eucyclops <dbl>, Eurytemora <dbl>, `Fish larvae` <dbl>,
#> #   Gastropod <dbl>, `Halicyclops fosteri` <dbl>, Harpacticoid <dbl>,
#> #   Hermit <dbl>, Hydrozoan <dbl>, Isopod <dbl>, Leptodiaptomus <dbl>, ...
```

Check for Dropped Sample

Erin Ambrose dropped 5/20/15 Station 8 from both datasheets. Environmental and zooplankton data should each have 59 rows, and they do.

Erin notes that there was no zooplankton "sample" (?) only nekton for that sample. I'm not sure if that means no sample was collected or there were no zooplankton in the sample. Anyway, she noted that this sample "threw off calculation of percent abundances."

Note that that sample is one of the Spring "washout" samples that cause trouble on our other analyses as well.

```
sum(! is.na((zoopl2 %>%
  filter((STATION == 4 & Month == 'May' & Year == 2015))))) == 0
#> [1] TRUE
sum(! is.na(station_data %>%
  filter((Station == 4 & Month == 'May' & Year == 2015)))) == 0
#> [1] TRUE
```

Correct Sample Row Alignment

I had some funny artifacts popping up in my initial (re) analyses, which I finally tracked down to the fact that the two data sets as I assembled them were in a different order from how Erin pulled them together. That is just because I used different tools that have different default ordering. Getting the data in alignment from two different datasheets in Excel is critical. SO, I force that alignment here.

Align Data Tables

```
zoopl2 <- zoopl2 %>%
  arrange(DATE, STATION)
station_data<- station_data %>%
  arrange(Date, Station)

head(zoopl2[,c(1,4)])
#> # A tibble: 6 x 2
#>   DATE                STATION
#>   <dttm>              <fct>
```

```
#> 1 2013-05-28 00:00:00 1
#> 2 2013-05-28 00:00:00 2
#> 3 2013-05-28 00:00:00 3
#> 4 2013-05-28 00:00:00 4
#> 5 2013-07-25 00:00:00 1
#> 6 2013-07-25 00:00:00 2
head(station_data[,c(1,2)])
#> # A tibble: 6 x 2
#>   Date                Station
#>   <dttm>              <fct>
#> 1 2013-05-28 00:00:00 1
#> 2 2013-05-28 00:00:00 2
#> 3 2013-05-28 00:00:00 3
#> 4 2013-05-28 00:00:00 4
#> 5 2013-07-25 00:00:00 1
#> 6 2013-07-25 00:00:00 2
```

Matrix of Species for `vegan`

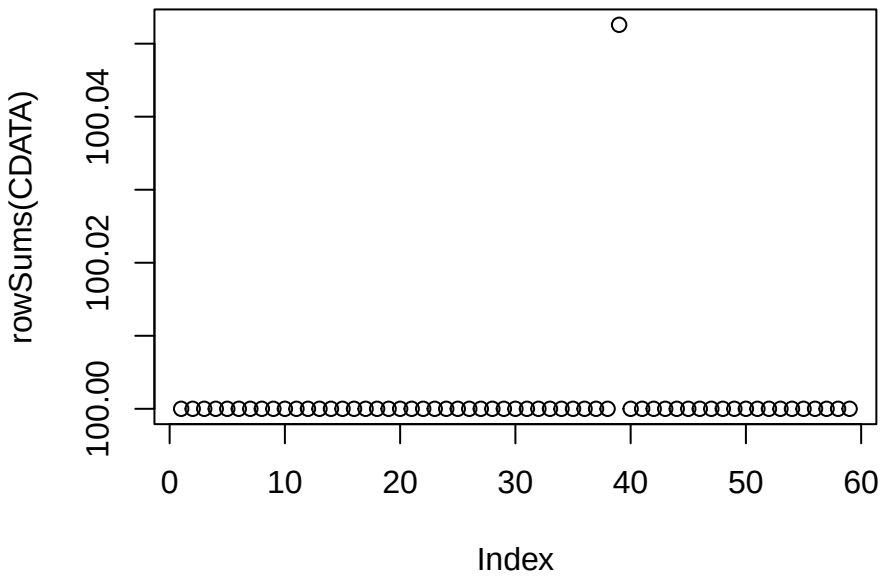
The `vegan` package likes to work with a matrix of species occurrences. Although the matrix can have row names that provide sample identifiers, that was not done here. The "matrix" I produce here is really a data frame with nothing but numeric values. While those are different data structures internally, `vegan` handles the conversion in the background.

```
CDATA <- zoopl2[,-c(1:4)]
```

Data Sanity Checks

We should have no NAs, and row sums should all be 1 (100%), at least within reasonable rounding error.

```
anyNA(CDATA)
#> [1] FALSE
plot(rowSums(CDATA))
```



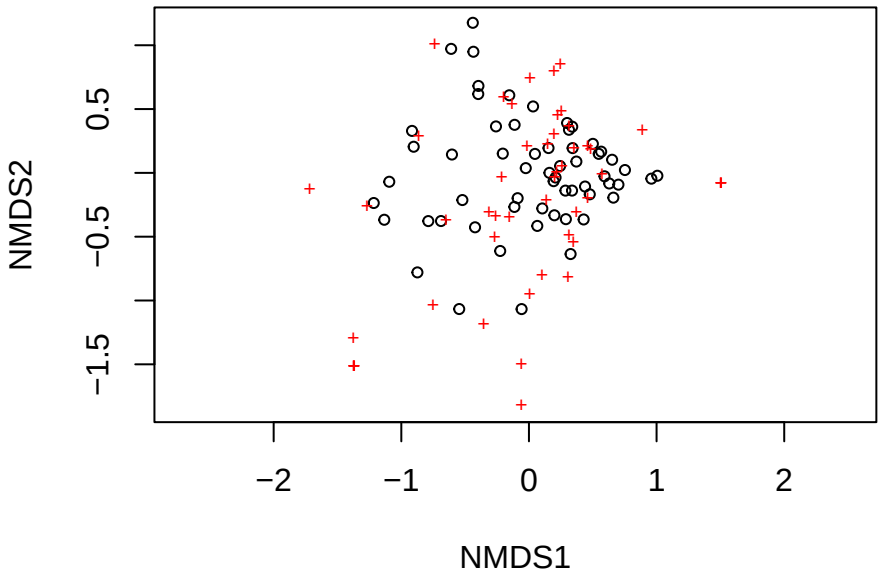
NMDS Analyses

```
NMDSE <- metaMDS(CDATA, autotransform = FALSE, k = 2, trymax = 75)
#> Run 0 stress 0.1509449
#> Run 1 stress 0.1790177
#> Run 2 stress 0.2035611
#> Run 3 stress 0.1746273
#> Run 4 stress 0.2091166
#> Run 5 stress 0.1742828
#> Run 6 stress 0.1790177
#> Run 7 stress 0.1505672
#> ... New best solution
#> ... Procrustes: rmse 0.00650441  max resid 0.03819226
#> Run 8 stress 0.1795435
#> Run 9 stress 0.1743709
#> Run 10 stress 0.1496383
#> ... New best solution
#> ... Procrustes: rmse 0.0237743  max resid 0.1595679
#> Run 11 stress 0.2105025
#> Run 12 stress 0.1493367
#> ... New best solution
#> ... Procrustes: rmse 0.006252744  max resid 0.03526922
#> Run 13 stress 0.1634485
#> Run 14 stress 0.1577084
#> Run 15 stress 0.1747259
#> Run 16 stress 0.198593
```

```
#> Run 17 stress 0.1505044
#> Run 18 stress 0.1729591
#> Run 19 stress 0.1808228
#> Run 20 stress 0.2044686
#> Run 21 stress 0.1747258
#> Run 22 stress 0.1808227
#> Run 23 stress 0.1669607
#> Run 24 stress 0.1658234
#> Run 25 stress 0.1505044
#> Run 26 stress 0.1496383
#> ... Procrustes: rmse 0.006264638  max resid 0.03552729
#> Run 27 stress 0.15446
#> Run 28 stress 0.1505044
#> Run 29 stress 0.15446
#> Run 30 stress 0.180383
#> Run 31 stress 0.1493367
#> ... New best solution
#> ... Procrustes: rmse 1.009933e-05  max resid 4.100708e-05
#> ... Similar to previous best
#> *** Solution reached
NMDSE
#>
#> Call:
#> metaMDS(comm = CDATA, k = 2, trymax = 75, autotransform = FALSE)
#>
#> global Multidimensional Scaling using monoMDS
#>
#> Data:     CDATA
#> Distance: bray
#>
#> Dimensions: 2
#> Stress:     0.1493367
#> Stress type 1, weak ties
#> Two convergent solutions found after 31 tries
#> Scaling: centring, PC rotation, halfchange scaling
#> Species: expanded scores based on 'CDATA'
```

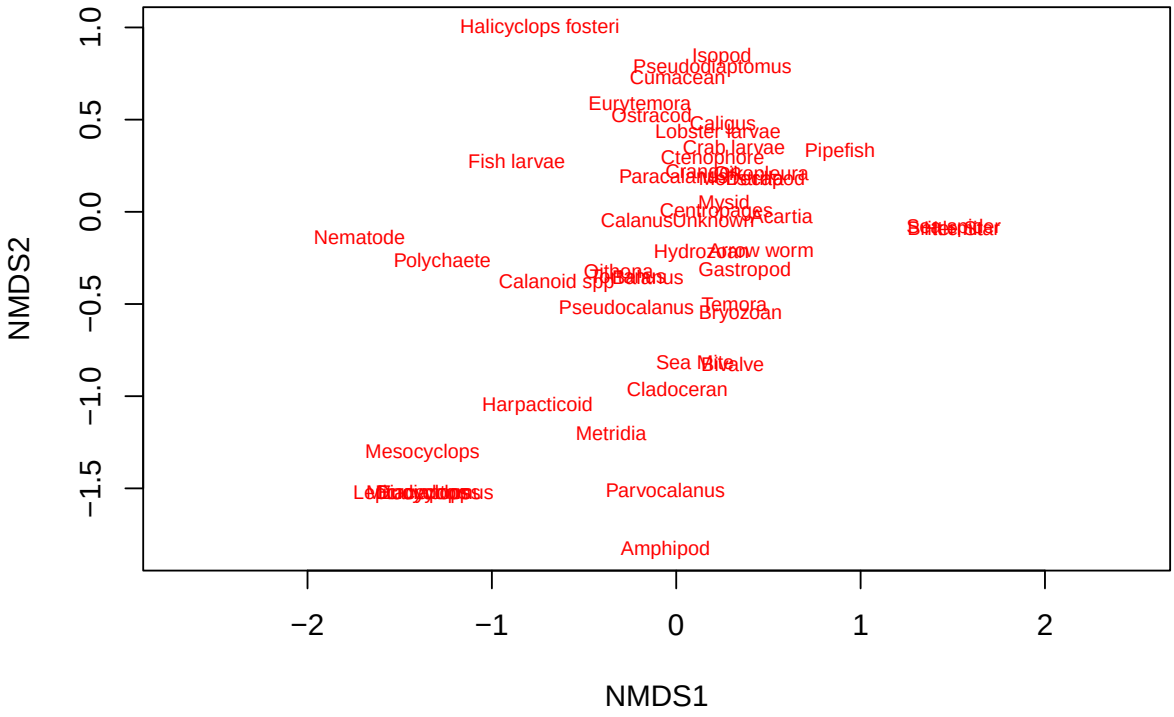
Plot

```
plot(NMDSE, type = 'p')
```



Plot Species

```
plot(NMDSE, 'species', type = 't')
```



Combining the NMDS Results with Environmental Data

I want to use the names of these variables as labels in graphics later. I capitalize variable names here, so they will appear capitalized in graphics without further action on my part.

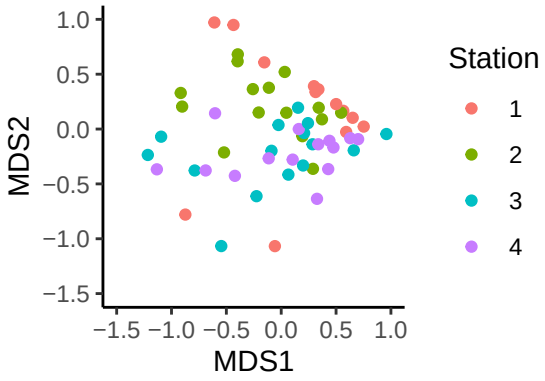
```
envNMDS <- station_data %>%
  select(-Date, -Month, -DOY, -riv_km, -AvgTurb) %>%
  mutate(Turb2 = log(Turb),
         Chl2 = log(Chl),
         RH2 = log1p(RH)) %>%
  mutate(sample_seq = as.numeric(Season) + (Year-2013)*3,
         sample_event = factor(sample_seq)) %>%
  cbind(as_tibble(NMDSE$points))
```

Plotting

These plots are intended principally to help us understand the NMDS from a more intuitive perspective. The idea is to plot the ordination, but colored by various predictor variables.

By Station

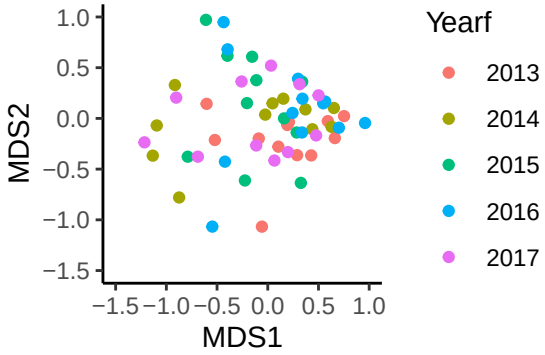
```
ggplot(envNMDS, aes(MDS1, MDS2)) +
    geom_point(aes(color=Station)) +
  xlim(c(-1.5,1)) +
  ylim(c(-1.5,1)) +
  theme(aspect.ratio=1)
#> Warning: Removed 2 rows containing missing values (geom_point).
```



Note that station 1 is split into a group along the upper edge and two points along the lower edge. The stations don't segregate fully, but there are trends. Other than those two spring samples, Station 1 is upper edge. Station 2 is upper zone as well. I suspect those two samples are "washout" event samples.

By Year

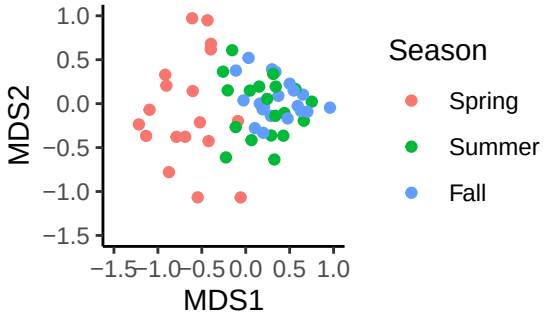
```
ggplot(envNMDS, aes(MDS1, MDS2)) +
  geom_point(aes(color=Yearf)) +
  xlim(c(-1.5,1)) +
  ylim(c(-1.5,1)) +
  theme(aspect.ratio=1)
#> Warning: Removed 2 rows containing missing values (geom_point).
```



MAYBE 2016 is towards the upper edge, but it's not clear at all. I don't see a robust pattern here.

By Season

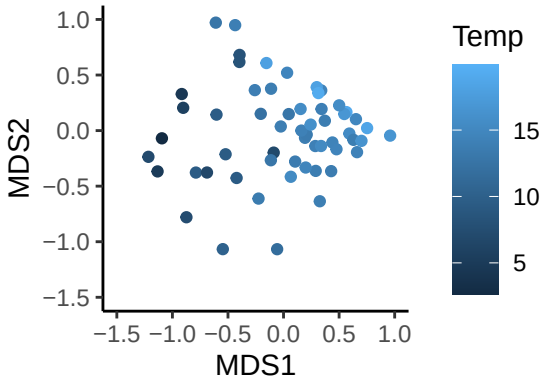
```
ggplot(envNMDS, aes(MDS1, MDS2)) +
  geom_point(aes(color=Season)) +
  xlim(c(-1.5,1)) +
  ylim(c(-1.5,1)) +
  theme(aspect.ratio=1)
#> Warning: Removed 2 rows containing missing values (geom_point).
```



Note the VERY strong association here, with Spring samples all to the left on the plot. Summer and Fall plots are fairly mixed up, but all to the left. That means Axis 1 can be interpreted as largely a "season" signal.

By Temperature

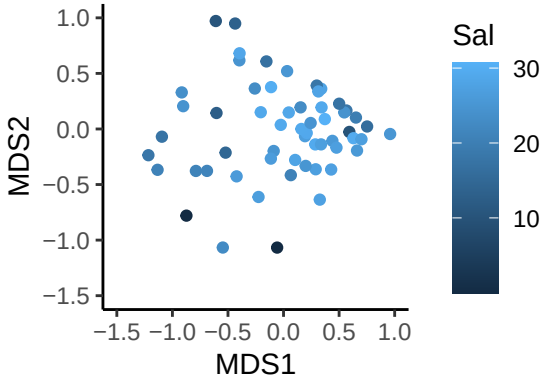
```
ggplot(envNMDS, aes(MDS1, MDS2)) +
  geom_point(aes(color=Temp)) +
  xlim(c(-1.5,1)) +
  ylim(c(-1.5,1)) +
  theme(aspect.ratio=1)
#> Warning: Removed 2 rows containing missing values (geom_point).
```



This reveals the same pattern as the last graphic, only filtered through the correlation between season and temperature. Cool temperatures in spring to the left.

By Salinity

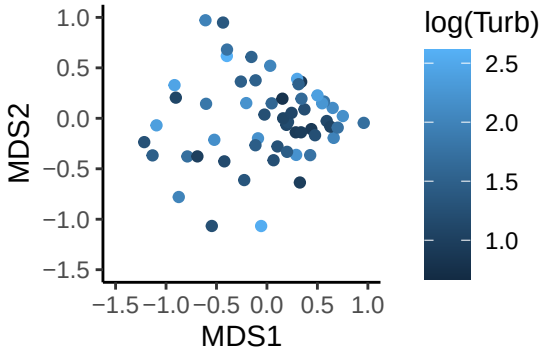
```
ggplot(envNMDS, aes(MDS1, MDS2)) +
  geom_point(aes(color=Sal)) +
  xlim(c(-1.5,1)) +
  ylim(c(-1.5,1)) +
  theme(aspect.ratio=1)
#> Warning: Removed 2 rows containing missing values (geom_point).
```



This one is hard to interpret. What jumps out at me here is the two VERY low salinity sites at the bottom, and the tendency for other lower salinity samples to fall to the left (spring) and along the upper edge (Station 1).

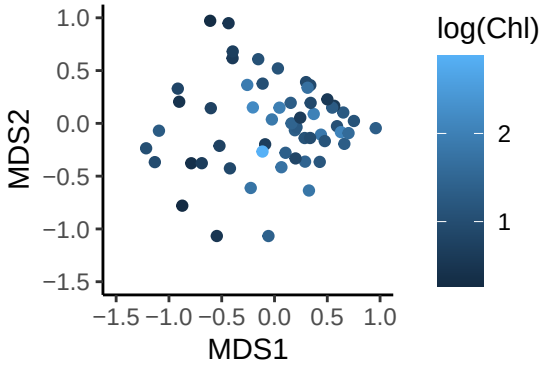
By Turbidity

```
ggplot(envNMDS, aes(MDS1, MDS2)) +
  geom_point(aes(color=log(Turb))) +
  xlim(c(-1.5,1)) +
  ylim(c(-1.5,1)) +
  theme(aspect.ratio=1)
#> Warning: Removed 2 rows containing missing values (geom_point).
```



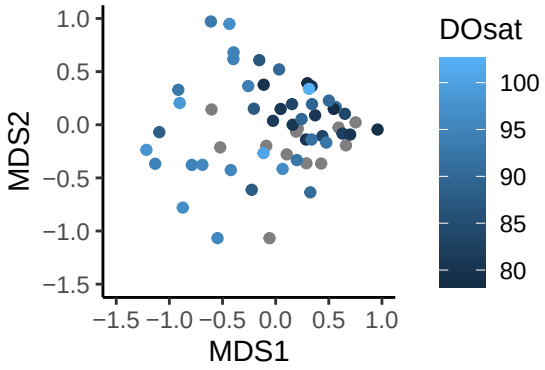
By Chlorophyll

```
ggplot(envNMDS, aes(MDS1, MDS2)) +
  geom_point(aes(color=log(Chl))) +
  xlim(c(-1.5,1)) +
  ylim(c(-1.5,1)) +
  theme(aspect.ratio=1)
#> Warning: Removed 2 rows containing missing values (geom_point).
```



By Oxygen Saturation

```
ggplot(envNMDS, aes(MDS1, MDS2)) +
  geom_point(aes(color=DOsat)) +
  xlim(c(-1.5,1)) +
  ylim(c(-1.5,1)) +
  theme(aspect.ratio=1)
#> Warning: Removed 2 rows containing missing values (geom_point).
```



Note that highest DO is to the left, providing an alternate "explanation" to considering axis 1 a seasonal axis.

Using `envfit` to Estimate Correlations

The `envfit()` function is fitting linear predictors to the two NMDS axes jointly. The related help file says "The environmental variables are the dependent variables that are explained by the ordination scores, and each dependent variable is analyzed separately." The model is always linear, which is different from our GAM models.

Despite that description, the R squared terms don't match R squared from linear models, so something else is going on here.

That means this is NOT a single statistical test, but a separate statistical fit for each predictor variable. Coefficients are the coordinates of a unit-length vector that points along the "direction" in ordination space that shows maximum correlation with the NMDS scores. So, if one goes up, the other necessarily goes down. The R2 term "is a"goodness of fit statistic" like the one from multiple regression models. The higher the number, the better the ability of the ordination scores to predict environmental variables

It's worth remembering that these results are based on randomization methods, so results change somewhat between repeated model runs. The relatively high number of permutations specified here helps keep those effects small.

Create Working Environmental Data

```
ef <- envfit(NMDSE, envNMDS[,c(1, 3:12)], permu = 9999, na.rm = TRUE)
ef
#>
#> ***VECTORS
#>
#>           NMDS1    NMDS2     r2 Pr(>r)
#> Temp    0.97606  0.21751 0.7597 0.0001 ***
#> Sal     0.98677 -0.16212 0.0979 0.1121
#> Turb   -0.33502  0.94221 0.1477 0.0329 *
#> DOsat -0.97723 -0.21220 0.3353 0.0001 ***
#> Chl     0.72746 -0.68615 0.0782 0.1709
#> RH     -0.22967  0.97327 0.2436 0.0010 ***
#> Turb2 -0.34246  0.93953 0.1501 0.0302 *
#> Chl2    0.85551 -0.51778 0.1660 0.0199 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> Permutation: free
#> Number of permutations: 9999
#>
#> ***FACTORS:
#>
#> Centroids:
#>                 NMDS1   NMDS2
#> Station1       0.1458  0.3011
#> Station2      -0.1540  0.3187
#> Station3      -0.1657 -0.2507
#> Station4       0.0648 -0.2421
#> Yearf2014     -0.0620 -0.0441
#> Yearf2015     -0.1255  0.1201
#> Yearf2016      0.1830  0.0678
#> Yearf2017     -0.1447 -0.0129
#> SeasonSpring -0.7443  0.0037
#> SeasonSummer  0.1803  0.0012
#> SeasonFall     0.3760  0.0872
#>
#> Goodness of fit:
#>              r2 Pr(>r)
#> Station 0.1959 0.0061 **
```

```
#> Yearf    0.0441 0.6842
#> Season   0.4689 0.0001 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> Permutation: free
#> Number of permutations: 9999
#>
#> 13 observations deleted due to missingness
```

Note 13 observations deleted due to missingness. Those must mostly be the 2013 data, which lack DO data. We can refit to include those data by dropping DO as a predictor.

Extracting Vector Information

The `ef` object is an `envfit` S3 object, with three named slots. The vector information we need to plot the environment arrows is available in `vectors`. But that object is itself also an S3 object, with five named items. The help page for `envfit()` tells us that the information we need for the direction of the arrows is in the `arrows` component. We are told that arrows contain "Arrow endpoints from vectorfit. The arrows are scaled to unit length."

```
ef$vectors$arrows
#>            NMDS1      NMDS2
#> Temp    0.9760582  0.2175096
#> Sal     0.9867717 -0.1621163
#> Turb   -0.3350237  0.9422097
#> DOsat  -0.9772263 -0.2121996
#> Chl     0.7274633 -0.6861466
#> RH     -0.2296717  0.9732682
#> Turb2  -0.3424577  0.9395332
#> Chl2    0.8555123 -0.5177826
#> attr(,"decostand")
#> [1] "normalize"
```

The information we need to determine the magnitude of those vectors is in the `r` component of the `vectors` component. We scale each of the arrows by the square root of the related r squared value.

```
arrows <- ef$vectors$arrows
rsq    <- ef$vectors$r
scaled_arrows <- as_tibble(arrows*sqrt(rsq)) %>%
  mutate(parameter = rownames(arrows))
```

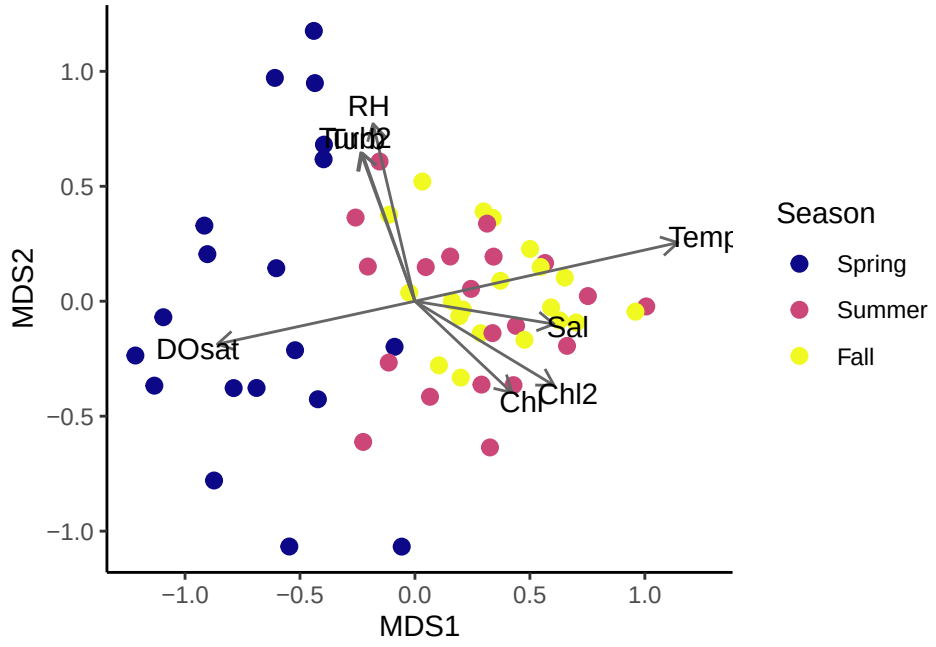
While we are creating vectors, we also want to create points for placing the annotations identifying each vector. We want to space the labels so they are a fixed distance beyond the end of each vector. We do that with a little vector addition.

```
scale_factor = 0.3   # Fraction of unit length beyond arrow to place annotation

scaled_arrows <- scaled_arrows %>%
  mutate(ann_xpos = NMDS1 + arrows[,1] * scale_factor,
         ann_ypos = NMDS2 + arrows[,2] * scale_factor)
```

Plotting `envfit()` Information

```
plt <- ggplot(data = envNMDS, aes(MDS1, MDS2)) +
  geom_point(aes(color = Season), size = 2.5) +
  geom_segment(data=scaled_arrows,
               mapping = aes(x=0,xend=ann_xpos,y=0,yend=ann_ypos),
               arrow = arrow(length = unit(0.25, "cm")) ,colour="grey40") +
  geom_text(data=scaled_arrows,
            mapping = aes(x=1.1 * ann_xpos,
                          y=1.1 * ann_ypos,label=parameter),
            size=4, nudge_x =0, nudge_y = 0, hjust = .5)+
  scale_color_viridis_d(option = 'C', name = 'Season') +
  coord_fixed()
plt
```



This is where the fact that the `envfit()` analysis is parameter by parameter gets a bit irritating. Temperature and dissolved oxygen are negatively correlated, for purely physical reasons.

It's also interesting that the turbidity and river herring arrows are nearly perpendicular to the temperature / DO arrows, suggesting nearly independent relationships to the community data.

- Axis 1 is season, which is highly correlated with temperature and dissolved oxygen, and less strongly associated with salinity.
- Axis 2 is LARGELY River Herring and Turbidity (which are themselves weakly correlated). Another vision is to see this axis as more or less a skewed upstream-downstream pattern. I am uncertain how to interpret the Chlorophyll association.

GAM Analysis

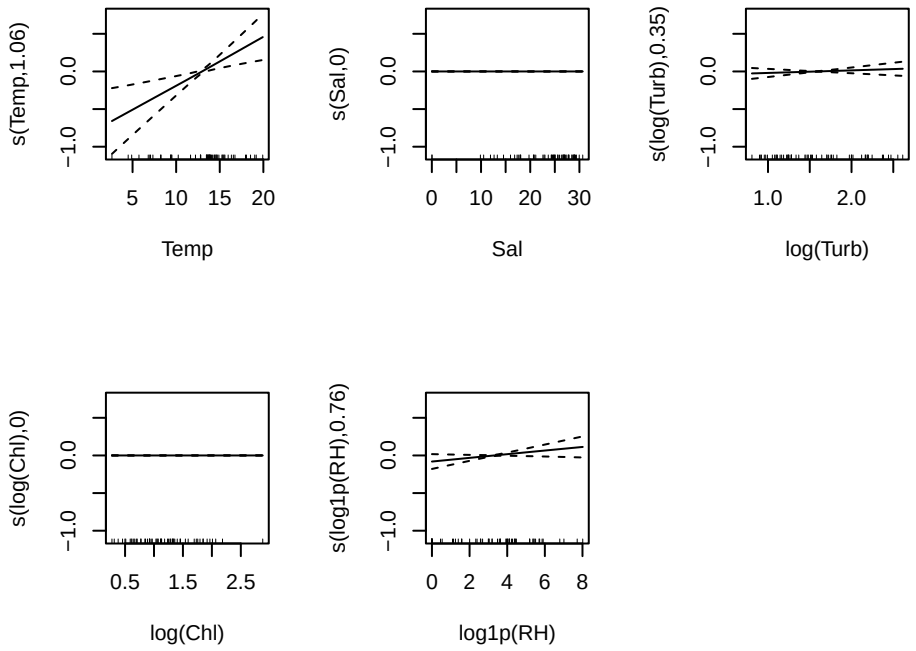
Axis 1

```
gam_1 <- gamm(MDS1 ~
               Station +
               Season +
               s(Temp, bs="ts") +
               s(Sal, bs="ts") +
               s(log(Turb), bs="ts") +
               s(log(Chl), bs="ts") +
               s(log1p(RH),bs="ts"),
              random = list(Yearf = ~ 1, sample_event = ~ 1),
             data = envNMDS, family = 'gaussian')
summary(gam_1$gam)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> MDS1 ~ Station + Season + s(Temp, bs = "ts") + s(Sal, bs = "ts") +
#>     s(log(Turb), bs = "ts") + s(log(Chl), bs = "ts") + s(log1p(RH),
#>     bs = "ts")
#>
#> Parametric coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -0.39930    0.10966  -3.641 0.000645 ***
#> Station2     -0.02135    0.11527  -0.185 0.853814
#> Station3      0.02310    0.11013   0.210 0.834710
#> Station4      0.11884    0.11594   1.025 0.310311
#> SeasonSummer  0.46736    0.16479   2.836 0.006584 **
#> SeasonFall    0.61728    0.15824   3.901 0.000288 ***
#> ---
#> 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)      1.056e+00      9 1.168 0.00314 **
#> s(Sal)       5.016e-09      9 0.000 0.56892
#> s(log(Turb)) 3.520e-01      9 0.070 0.23282
#> s(log(Chl))  2.864e-09      9 0.000 0.87903
#> s(log1p(RH)) 7.564e-01      9 0.338 0.07363 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.769
#>   Scale est. = 0.049746  n = 58
```

```
anova(gam_1$gam)
#>
#> Family: gaussian
#> Link function: identity
```

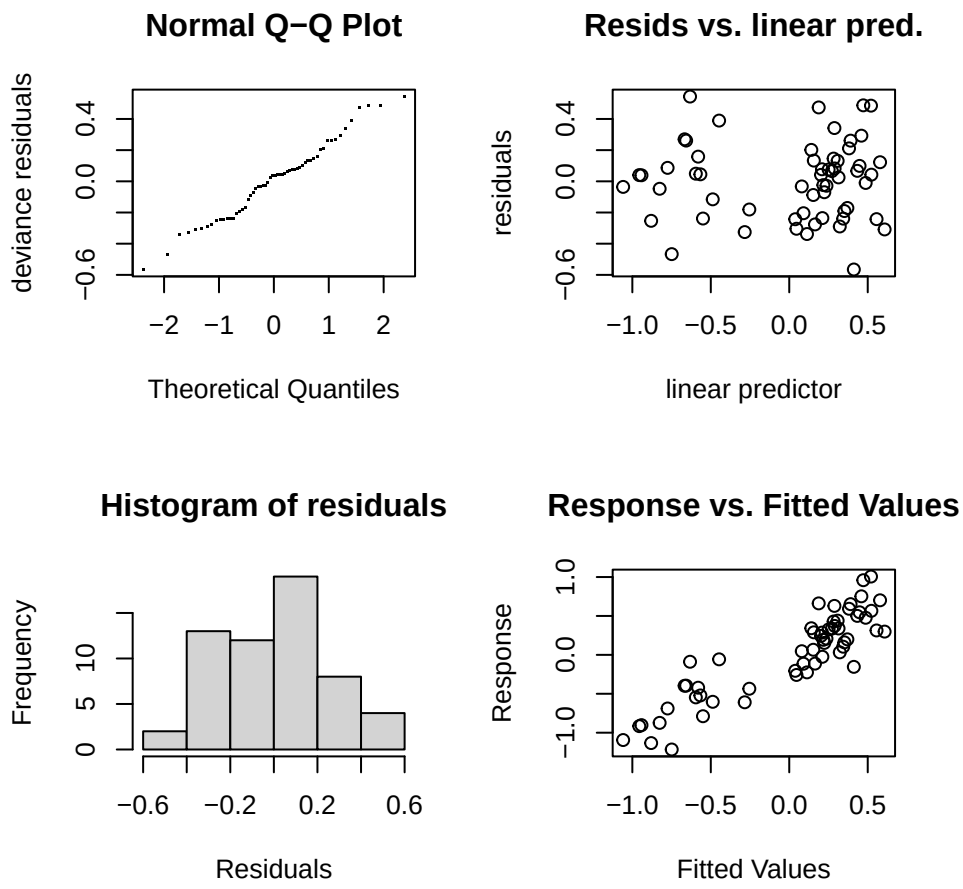
```
#>
#> Formula:
#> MDS1 ~ Station + Season + s(Temp, bs = "ts") + s(Sal, bs = "ts") +
#>     s(log(Turb), bs = "ts") + s(log(Chl), bs = "ts") + s(log1p(RH),
#>     bs = "ts")
#>
#> Parametric Terms:
#>         df     F  p-value
#> Station  3 0.852 0.471974
#> Season   2 8.606 0.000616
#>
#> Approximate significance of smooth terms:
#>                    edf    Ref.df     F p-value
#> s(Temp)      1.056e+00 9.000e+00 1.168 0.00314
#> s(Sal)       5.016e-09 9.000e+00 0.000 0.56892
#> s(log(Turb)) 3.520e-01 9.000e+00 0.070 0.23282
#> s(log(Chl))  2.864e-09 9.000e+00 0.000 0.87903
#> s(log1p(RH)) 7.564e-01 9.000e+00 0.338 0.07363
```

```
oldpar <- par(mfrow = c(2,3))
plot(gam_1$gam)
par(oldpar)
```



As we saw from the `envfit()` analysis, Axis 1 is associated with the SEASON, with spring clearly separated from Summer and Fall. Temperature is also important in GAM analysis (note that the relationship is essentially linear). Last, I see a possible connection to River Herring. Essentially, low Axis 1 Scores are spring, with lower temperatures.

```
oldpar <- par(mfrow = c(2,2))
gam.check(gam_1$gam)
```




```
#>
#> 'gamm' based fit - care required with interpretation.
#> Checks based on working residuals may be misleading.
#> Basis dimension (k) checking results. Low p-value (k-index<1) may
#> indicate that k is too low, especially if edf is close to k'.
#>
#>                    k'      edf k-index p-value
#> s(Temp)      9.00e+00 1.06e+00    1.01   0.500
#> s(Sal)       9.00e+00 5.02e-09    1.12   0.790
#> s(log(Turb)) 9.00e+00 3.52e-01    0.87   0.075 .
#> s(log(Chl))  9.00e+00 2.86e-09    1.04   0.525
#> s(log1p(RH)) 9.00e+00 7.56e-01    1.07   0.620
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
par(oldpar)
```

The model is fairly well behaved. No problems obvious here.

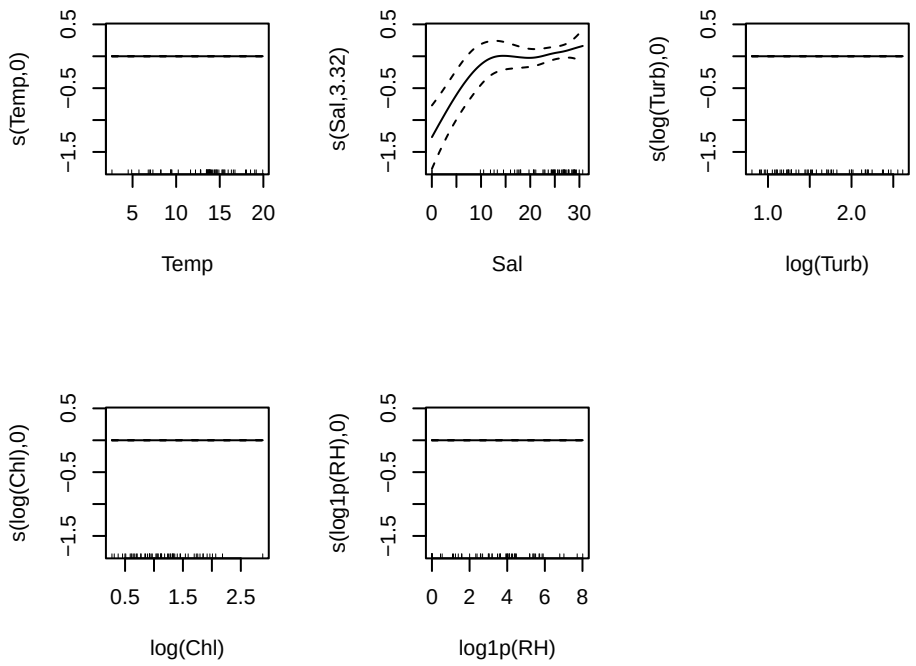
Axis 2

```
gam_2 <- gamm(MDS2 ~
               Station +
               Season +
               s(Temp, bs="ts") +
               s(Sal, bs="ts") +
               s(log(Turb), bs="ts") +
               s(log(Chl), bs="ts") +
               s(log1p(RH),bs="ts"),
              random = list(Yearf = ~ 1, sample_event = ~ 1),
             data = envNMDS, family = 'gaussian')
#> Warning in lme.formula(y ~ X - 1, random = rand, data = strip.offset(mf), : nlminb problem, converge
#>   message = singular convergence (7)
summary(gam_2$gam)
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> MDS2 ~ Station + Season + s(Temp, bs = "ts") + s(Sal, bs = "ts") +
#>     s(log(Turb), bs = "ts") + s(log(Chl), bs = "ts") + s(log1p(RH),
#>     bs = "ts")
#>
#> Parametric coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   0.44037    0.15845   2.779 0.007721 **
#> Station2     -0.22146    0.15732  -1.408 0.165568
#> Station3     -0.62923    0.14664  -4.291 8.42e-05 ***
#> Station4     -0.64146    0.15205  -4.219 0.000107 ***
#> SeasonSummer -0.16278    0.11255  -1.446 0.154504
#> SeasonFall   -0.08603    0.12167  -0.707 0.482889
#> ---
#> 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)      4.058e-09      9 0.000    0.427
#> s(Sal)       3.317e+00      9 3.849 3.73e-05 ***
#> s(log(Turb)) 6.399e-09      9 0.000    0.944
#> s(log(Chl))  7.512e-09      9 0.000    0.483
#> s(log1p(RH)) 1.467e-08      9 0.000    0.809
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> R-sq.(adj) =  0.501
#>   Scale est. = 0.075246  n = 58
```

```
anova(gam_2$gam)
#>
#> Family: gaussian
#> Link function: identity
#>
```

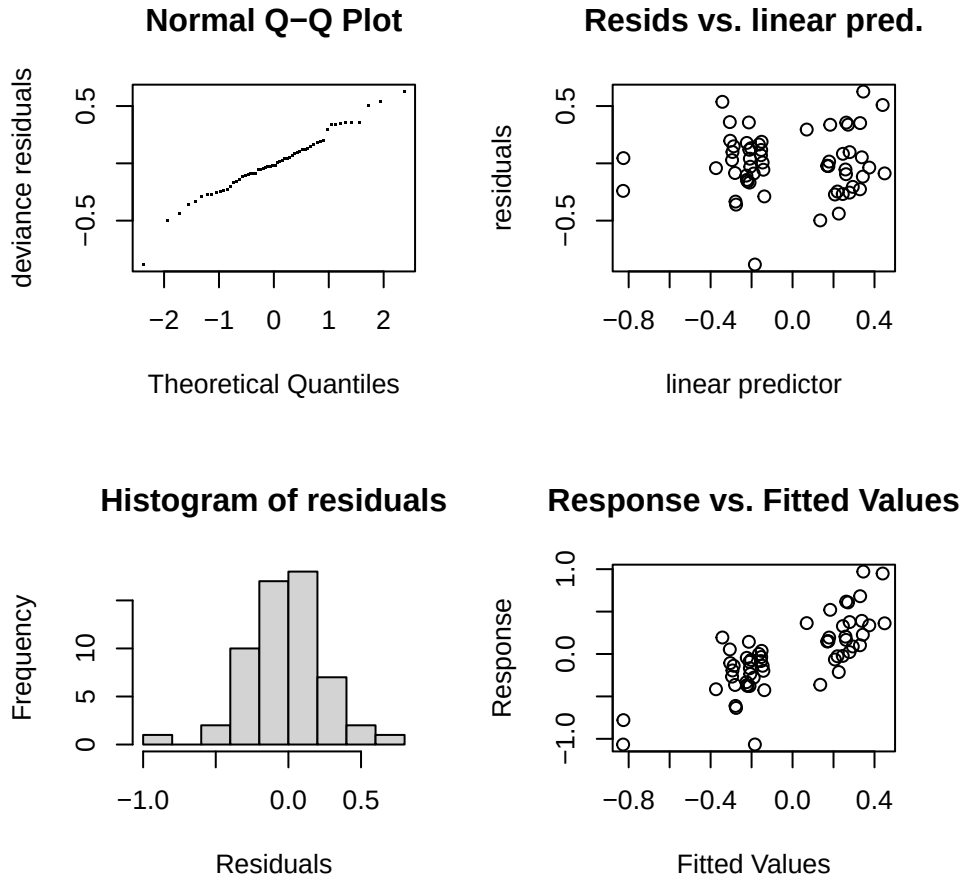
```
#> Formula:
#> MDS2 ~ Station + Season + s(Temp, bs = "ts") + s(Sal, bs = "ts") +
#>     s(log(Turb), bs = "ts") + s(log(Chl), bs = "ts") + s(log1p(RH),
#>     bs = "ts")
#>
#> Parametric Terms:
#>         df      F  p-value
#> Station  3 11.812 6.23e-06
#> Season   2  1.113    0.337
#>
#> Approximate significance of smooth terms:
#>                     edf    Ref.df     F  p-value
#> s(Temp)      4.058e-09 9.000e+00 0.000    0.427
#> s(Sal)       3.317e+00 9.000e+00 3.849 3.73e-05
#> s(log(Turb)) 6.399e-09 9.000e+00 0.000    0.944
#> s(log(Chl))  7.512e-09 9.000e+00 0.000    0.483
#> s(log1p(RH)) 1.467e-08 9.000e+00 0.000    0.809
```

```
oldpar <- par(mfrow = c(2,3))
plot(gam_2$gam)
par(oldpar)
```



Axis 2 shows the effect of a couple of low salinity samples, which are very low on Axis # 2. But the big story here is the connection with station. It will be worth comparing pairwise marginal means.

```
oldpar <- par(mfrow = c(2,2))
gam.check(gam_2$gam)
```

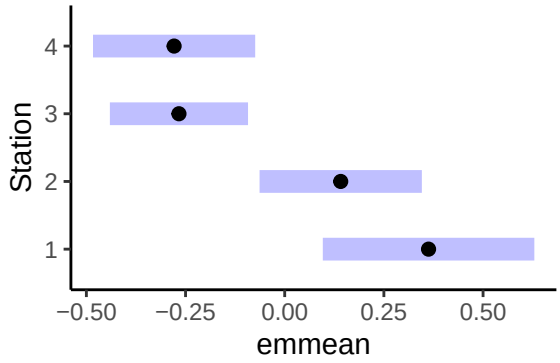



```
#>
#> 'gamm' based fit - care required with interpretation.
#> Checks based on working residuals may be misleading.
#> Basis dimension (k) checking results. Low p-value (k-index<1) may
#> indicate that k is too low, especially if edf is close to k'.
#>
#>                    k'      edf k-index p-value
#> s(Temp)      9.00e+00 4.06e-09    1.11    0.74
#> s(Sal)       9.00e+00 3.32e+00    1.09    0.67
#> s(log(Turb)) 9.00e+00 6.40e-09    0.88    0.16
#> s(log(Chl))  9.00e+00 7.51e-09    1.24    0.93
#> s(log1p(RH)) 9.00e+00 1.47e-08    1.16    0.86
par(oldpar)
```

Again, it looks like the low salinity samples have a disproportionate effect. Otherwise, this model looks fairly robust.

Pairwise Comparisons

```
Sta_emms <- emmeans(gam_2, ~Station, type = 'response',
                    data = envNMDS)
plot(Sta_emms)
```




```
pairs(Sta_emms, adjust ='bonferroni')
#>  contrast            estimate    SE   df t.ratio p.value
#>  Station1 - Station2   0.2215 0.157 48.7   1.408  0.9934
#>  Station1 - Station3   0.6292 0.147 48.7   4.291  0.0005
#>  Station1 - Station4   0.6415 0.152 48.7   4.219  0.0006
#>  Station2 - Station3   0.4078 0.110 48.7   3.702  0.0033
#>  Station2 - Station4   0.4200 0.109 48.7   3.845  0.0021
#>  Station3 - Station4   0.0122 0.111 48.7   0.110  1.0000
#>
#> Results are averaged over the levels of: Season
#> P value adjustment: bonferroni method for 6 tests
```

The big story is that Axis 2 is associated with the differences between Stations 1 and 2 (high values) and Stations 3 and 4. It is curious that this model does not pull out differences according to Turbidity, since Turbidity also differs the same way.