Surface Water Analyte Seasonality Analysis

Rebecca Stubbs

February 16, 2018

The Prompt

Background: The attached Excel file (Dataset_C) contains analytical chemistry data from a variety of surface water stations.

Given the attached dataset:

- Develop a tool for plotting time series by station and analyte.
- Symbolize data by season.
- Add a horizontal line for the mean concentration to each plot.
- Which analytes have the most seasonal variability? What tools/approach did you use?
- Produce a pdf of your plots.

The Answer/Analysis

There seem to be two stages of tasks set forth here: the first, to do some data visualization of a multi-dimensional data set, and the second, to determine which of the analytes show strong seasonal variability. Before diving into the seasonality analysis, I first tackled the data visualization of the observations by analyte and station.

NB: This document is as a .pdf rendering of an R Markdown, which allows for the integration of documentation, code, and outputs. What follows is all of the code required to process the data, and generate the outputs for this exercise.

First Forays and Data Explorations

Reading in the data, loading relevant libraries:

```
# Surface Water Data Viz and Seasonality; RStubbs 02/2018
# Generates plots and calculates sesaonally-adjusted data
# to compare to raw data to determine magnitude of seasonality
# Input: Surface water observations by station-analyte

rm(list=ls()) # Clear working environment

library(MapSuite) #mylibrary, has many common libs as dependencies
library(hexbin) # For the hex-bin plots
library(lme4) # Mixed-effects modeling package
library(pander) # tables for R markdown

# Running this Markdown from the same dir as the data, the using relative file paths
sw<-fread("Dataset_C.csv")# Read in surface water observations as data.table sw
```

Next, I calculate various columns describing the date of observations that may be useful later on.

```
# Parse out date information from character date column
  sw[,index:=seq(1:nrow(sw))]
  sw[,Month:=as.numeric(strsplit(SampleDate,"/")[[1]][1]),by=index]
  sw[,Day:=as.numeric(strsplit(SampleDate,"/")[[1]][2]),by=index]
  sw[,Year:=as.numeric(strsplit(SampleDate,"/")[[1]][3]),by=index]
# Create formal column of integer-date
  sw[,Date:=as.IDate(paste0((2000+Year),"-",Month,"-",Day))]
```

In order to symbolize by season, I am lumping observations into seasons by month. It's certainly possible to do this a more sophisticated way, based on cut points of the equinoxes; for now, however, month-by-month approximations seem adequate. December, January and February were assigned winter; March, April, and May were assigned spring; June, July, and August were assigned summer, and September, October, and November were assigned fall.

```
sw[Month %in% c(12,1,2), Season:='Winter']
sw[Month %in% c(3,4,5), Season:='Spring']
sw[Month %in% c(6,7,8), Season:='Summer']
sw[Month %in% c(9,10,11), Season:='Fall']
# Defining Season as a factor variable
sw[,Season:=factor(Season, levels = c("Winter","Spring","Summer","Fall"))]
```

Having generated the variables needed for the analysis, I can proceed to data visualization.

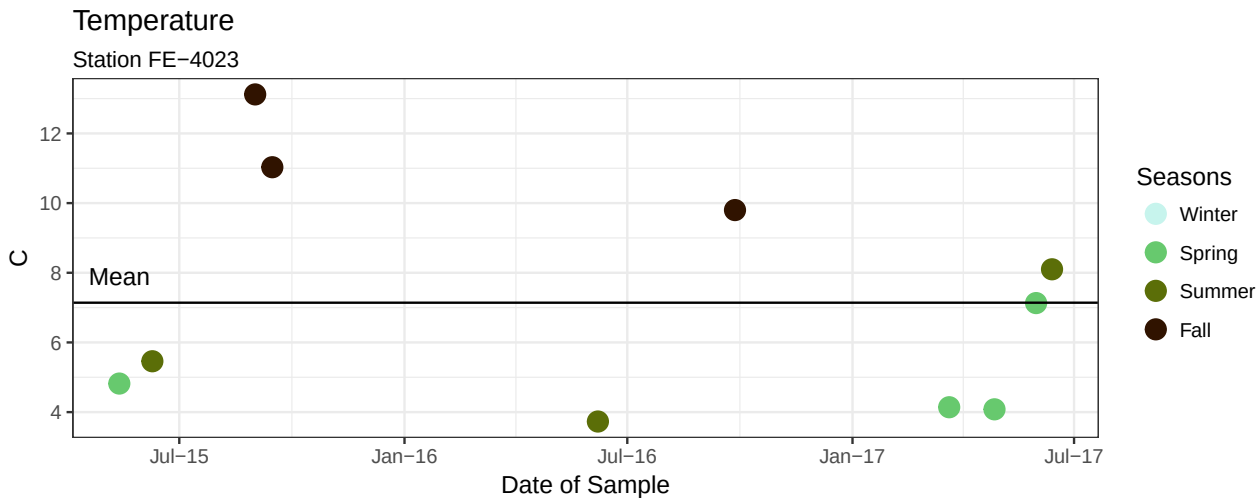
Plotting Each Analyte by Station, Over Time

In order to make plotting by each station and analyte straightforward, I have written a function that takes a station name, and analyte name as input, and returns a plot of the points, color-coded by season, with a black line representing the mean of the analyte at that station across the full time period.

```
# Define a color pallette for the factor variable, Season
SeasonColors<-wpal("foliage",noblack=T,n=4) # grab colors from MapSuite's pallettes
names(SeasonColors) <- c("Winter","Spring","Summer","Fall")

# Define function to generate plot
  MakeAnalyteTSPlot<-function(a,s){

    p<-ggplot(sw[StandardAnalyte==a & StationName==s],
      aes(x= Date, y=StandardResult, color=Season)) + geom_point(size=4) +
      xlab("Date of Sample") +
      scale_x_date(labels = function(x) format(x, "%b-%y")) +
      ylab(sw[StandardAnalyte==a & StationName==s]$StandardUnit[1]) +
      ggtitle(paste0(a), subtitle=paste0("Station ",s)) + theme_bw()  +
      scale_colour_manual(name = "Seasons",values = SeasonColors, drop=F) +
      geom_hline(yintercept = mean(sw[StandardAnalyte==a &
      StationName==s]$StandardResult)) +
      annotate("text", min(sw[StandardAnalyte==a & StationName==s]$Date),
               mean(sw[StandardAnalyte==a &StationName==s]$StandardResult),
               vjust = -1, label = "Mean")

  return(p)
  }
```

Before iterating over all of the analytes and stations, and saving them to PDFs, I will test out the plotting function on one analyte, and one station. For these plots, I have left the Y scale free to roam with the minimum and maximum of each station, with the idea of focusing on within-station variation–another flavor of these graphs would include a "set scale" that remained constant for all stations.

```
# Make and print plot
  ts<-MakeAnalyteTSPlot(a="Temperature",s="FE-4023")
  print(ts)
```



Each plot can now be written to a .pdf for each analyte, by station.

```
# Not run every time the markdown is created; file generation takes 10-15 min
for (a in unique(sw$StandardAnalyte)){
  print(a) # Keep track of what analyte you are on
  # Start a PDF document of results
    pdf(paste0("/Users/stubbsrw/Documents/git_code/stubbs_repo/",
        "fe_problems/surface_water/station_analyte_time_series/",a,".pdf"))
     # Generate plot for each Analyte over time, for each station.
      for (s in unique(sw[StandardAnalyte==a]$StationName)){
        ts<-MakeAnalyteTSPlot(a=a,s=s) # Get inputs from UI, use function
        print(ts)  # Print it to the PDF
      }
  dev.off() # Close PDF for each
}
```

Interactive Visualization

Rather than simply having .PDFs, I sometimes find it useful to create a quick and dirty interactive visualization, with options to select and sub-set data. This won't work in a static file format like .PDF, but you can check out an interactive version of this plot, and some of the plots below,for each station and analyte, at this URL: **https://rebeccastubbs.shinyapps.io/surface_water_app/**

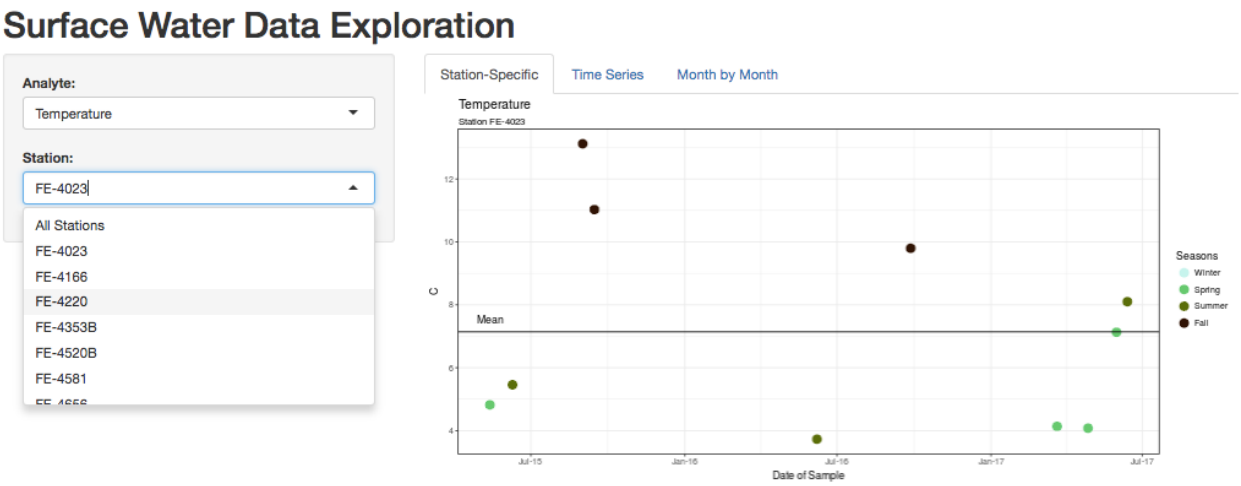


Figure 1: Screen-Capture of Interactive Tool Available Online, at https://rebeccastubbs.shinyapps.io/surface_water_app/

Quantifying Seasonality

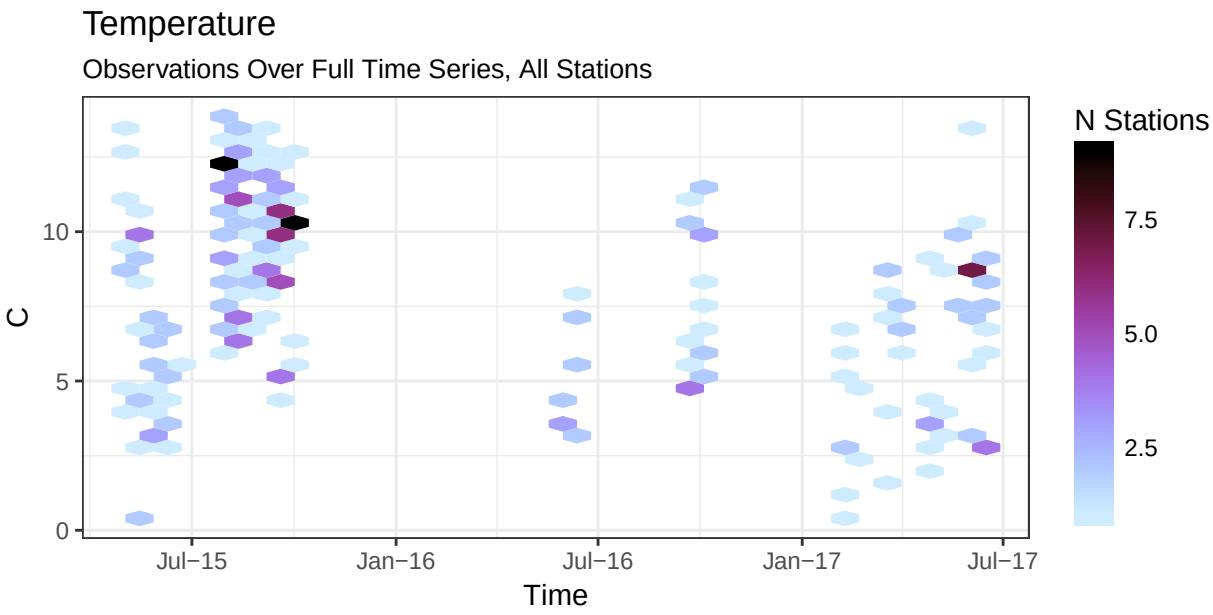
The fundamental question here seems to be, "To what extent is the variability in the data due to seasonal effects, rather than annual or other differences?" Unfortunately, there aren't very many data points per station within this data set, and the time period of observation only lasts a few years' time. This makes disentangling variation in observations for each site difficult–the change in observed data could be due to measurement error, a product of seasonal variation, inter-annual variation, random noise, the effect of a recent event, or the influence of other unknown factors.

Without knowing the spatial location of any of the stations, I assumed that the stations would be nearby one another, and subject to (at least roughly) the same weather and insolation patterns— if these stations were far apart, it would be even more difficult to measure the "seasonal component" of each analyte, since the magnitude of seasonal changes could be subject to variables such as latitude and elevation.

Rather than experimenting on a chemical analyte with which I am less familiar, I show examples below of the process first undertaken with temperature, a measured (but not chemically analyzed) feature of the data set. Looking at the results of this analysis for temperature also might give me a sense for how strong the seasonal effect might be, since temmperature is dependent on time of year, but to greater and lesser extents depending on the location of interest.

As a first pass, I made a frequency plot that showed the observed values across the entire time period, for all stations.

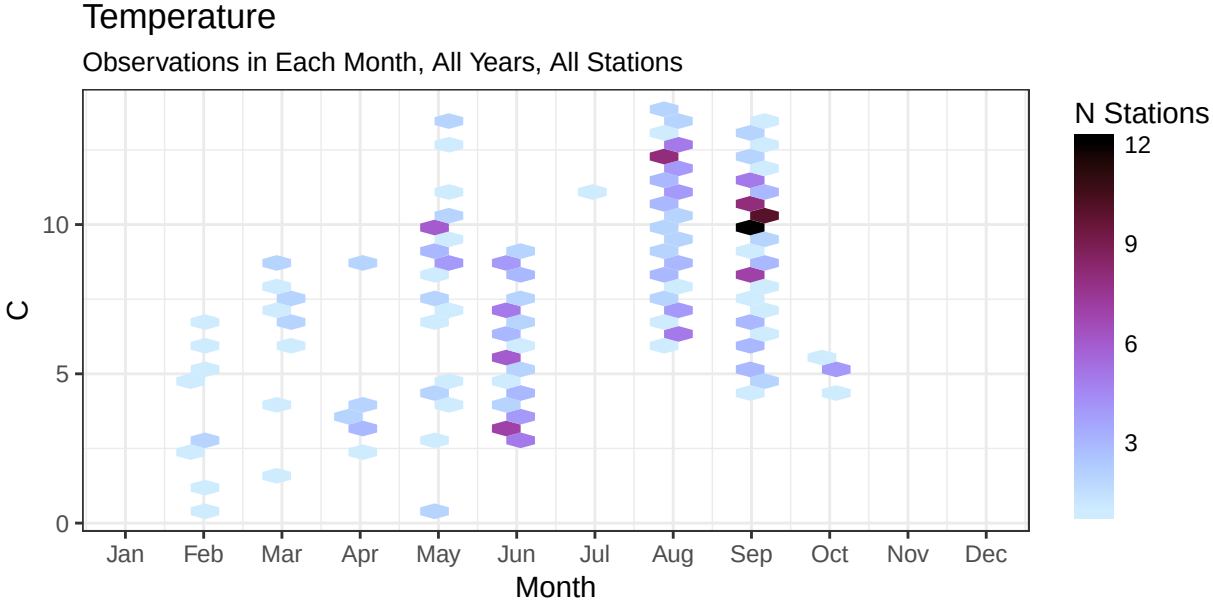
```
# Plotting using the HexBin Frequency graphics, where the number of
#stations with an observation in that category is essentially heat-mapped
plot_full_ts<-function(a){
p<-ggplot(sw[StandardAnalyte==a], aes(x=Date, y=StandardResult)) +
  geom_hex() + # honeycomb-plot geometry
  scale_x_date(labels = function(x) format(x, "%b-%y")) + xlab("Time") +
  ylab(sw[StandardAnalyte==a]$StandardUnit[1]) +
  ggtitle(paste0(a),
          (subtitle="Observations Over Full Time Series, All Stations")) +
  theme_bw() + scale_fill_gradientn(colors=wpal("berries")) +
  guides(fill=guide_colourbar(title="N Stations",
                              title.position="top", barheight=10, barwidth=1,
                              label=TRUE, ticks=FALSE, direction="vertical"))
return(p)
}
plot_full_ts("Temperature")
```



This graph highlights the extent to which surface water temperature varied both from year to year, and for any given month– this surprises me somewhat, I would have expected to see a clearer trend for something that is directly associated with the seasons.

Pooling observations across stations and also years could give a better perspective, so I also generate a plot in which observations from all stations are combined for each month of the year:

```
# Plotting by Month, for all years, all stations
plot_by_month<-function(a){
p<-ggplot(sw[StandardAnalyte==a], aes(x= Month, y=StandardResult)) + geom_hex() +
  ggtitle(paste0(a),
          subtitle="Observations in Each Month, All Years, All Stations") +
  scale_x_continuous(limits=c(1,12),breaks=seq(1,12),
                     labels=c("Jan","Feb","Mar","Apr",
                     "May","Jun","Jul","Aug",
                     "Sep","Oct","Nov","Dec")) +
  ylab(sw[StandardAnalyte==a]$StandardUnit[1]) + theme_bw() +
    scale_fill_gradientn(colors=wpal("berries")) +
  guides(fill=guide_colourbar(title="N Stations", title.position="top",
                              barheight=10, barwidth=1, direction="vertical",
                              label=TRUE, ticks=FALSE))
return(p)
}
plot_by_month("Temperature")
```



There are some interesting results here– it appears that the temperature data is relatively noisy across the different sites, and that even within the same month, there is a large degree of variation in observed surface temperatures.

It looks like there is a trend there to the casual eye, but it is interesting to note that there are very, very few observations during the winter months for this analyte. Now that the architecture to generate these plots has been vetted, I put versions of these plots for each analyte as part of the interactive data visualization tool rather than generating a booklet of PDFs. **https://rebeccastubbs.shinyapps.io/surface_water_app/**.

Using a mixed-effects model to tease out seasonality from the data

To determine the impact of seasonality on the measurements separate from the influence of variation across years, I fit a mixed-effects model[1] with parameters for an intercept (representing the expected/mean observed value) with respect to all of the data, and deviations from that intercept (modeled as random effects) for each year and season. This kind of model fits intercept shifts for each sub-group (such as season) of the data, based on the idea that each season's deviation from the global/data-wide mean is pulled from a mean 0, normal distribution, with the standard deviation representing the season's variance away from that global mean as a modeled parameter. From a theory standpoint, this model expresses that the differences seen in the observations due to seasons are the product of a stochastic process we have not observed, and that by default, these subgroups are likely to be deviations from an overall pattern (the 'global' intercept)– without evidence from the data collected, the model's "assumption" is that a given subgroup is no different than the mean.

The task at hand is to determine which analytes have "the **most** seasonal variability"– however, the analytes are all in different units! To be able to compare model parameters from one analyte to another, I converted all of the observed values to z-scores, re-scaling the data such that each recorded observation is represented as deviations away from the mean.

By extension, the random effect values returned from the model for each season represent the amount each season's expected value is different from the global mean, in units of sigma for that specific analyte or measured quantity.

First, I calculate the "z-score" transformed version of the measured observations for each analyte, across all stations and for all observations in time:

```
# Calculate z-scores
  # scale() is equivalent to (x-mean)/sd, calc. for each analyte
  sw[,z_score:=scale(StandardResult), by=StandardAnalyte]
```

Next, I fit the model (tested on Temperature) using the lme4 package, with z-score as the dependent variable, and a random intercept on year and season.

```
  analyte<-sw[StandardAnalyte=="Temperature"]
  mod <- lme4::lmer(z_score  ~ 1+ (1|Year)+(1|Season),data=analyte) # build model
  summary(mod) # show summary of model fit parameters
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: z_score ~ 1 + (1 | Year) + (1 | Season)
##    Data: analyte
##
## REML criterion at convergence: 677.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.8791 -0.7695  0.1593  0.7642  2.5436
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Season   (Intercept) 0.2676   0.5173
##  Year     (Intercept) 0.1899   0.4358
##  Residual             0.7823   0.8845
## Number of obs: 255, groups:  Season, 4; Year, 3
##
```

[1] Note that this modeling technique is called something different by almost every discipline... Wikipedia informs me that these models are also called multilevel models, hierarchical linear models, nested data models, random coefficient, random-effects models, random parameter models, or split-plot designs.

```
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  -0.4277     0.3719   -1.15
```

Taking a look at the random effect parameters for each Season, we can see the estimated values of the random intercepts based on each year and season category:

```
# Capture the model's seasonal random effect values
season_re<-data.table(Analyte="Temperature",
                      Season=levels(analyte$Season),
                      ranef(mod)$Season)
setnames(season_re,"(Intercept)","Intercept_shift")
pander(season_re)
```

Analyte	Season	Intercept_shift
Temperature	Winter	-0.6714
Temperature	Spring	0.02116
Temperature	Summer	0.1591
Temperature	Fall	0.4912

This can be interpreted to mean that observations taken in the fall are estimated to be .49 standard deviations higher than the average across the whole time period, regardless of the year of observation, and random noise. Adding together the absolute value of these estimated parameters gives one concise number that describes the magnitude of seasonality for this analyte.

Now, I undertake the same process for each of the analytes. I am excluding the analytes from this analysis for which fewer than 4 seasons were observed.

```
  season_random_effects<-list() # Emtpy list to hold seasonal RE values
  mods<-list() # A list to explore the model attributes for each analyte
  skipped<-c() # keep a list of skipped analytes, for further examination if needed

  analytes<-unique(sw$StandardAnalyte)  # Determine list of analytes
   # Exclude dissolved mercury; that model is unidentifable apparently
     analytes<-analytes[!analytes %in% c("Mercury, dissolved")]

  for (a in analytes){ # For each analyte
    # Check to see if all seasons are observed, and proceed, else skip
    if(length(unique(sw[StandardAnalyte==a]$Season))==4){
      analyte<-sw[StandardAnalyte==a] # Subset data
      mod <- lmer(z_score  ~ 1+ (1|Year)+(1|Season),data=analyte) # Fit model
      mods[[a]]<-mod

    # Save random effect values
      season_random_effects[[a]]<-data.table(Analyte=a,
                      Season=levels(analyte$Season),
                      ranef(mod)$Season)

    }else{
      skipped<-c(skipped,a)
    }
  }
```

Having estimated the random effect values for each of the analytes, the sum of the absolute value of the seasonal intercept shifts will indicate which analytes are the most seasonally driven:

```
  season_random_effects<-rbindlist(season_random_effects) # Combine to 1 data frame
  setnames(season_random_effects,"(Intercept)","Intercept_shift") # Clarify colname
# Create sum for each analyte
  seasonal_summary<-season_random_effects[,list(Seasonality=sum(abs(Intercept_shift))),
                        by="Analyte"][order(-Seasonality)]
```

The results: a rank-ordered table of analyte or measured quantity, by seasonality

```
# Make rank-order table
pander(seasonal_summary,
caption=paste0("Rank-ordered magnitude of seasonal effect for each analyte ",
               "represented by the sum of \n the absolute values ",
               "of each season's contribution to analyte variation,\n ",
               "in units of analyte-specific standard deviation."))
```

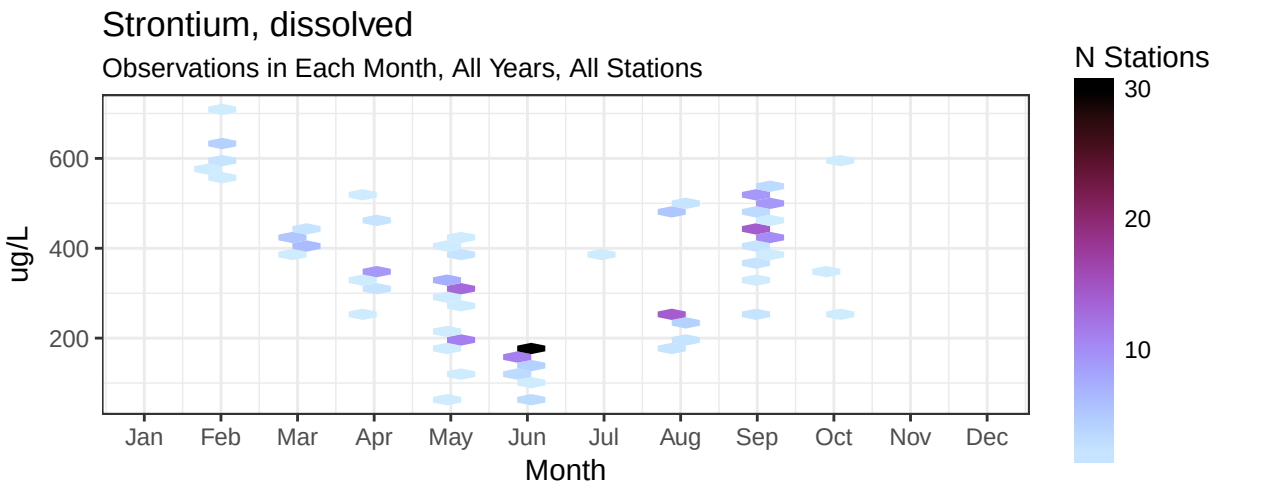
Table 2: Rank-ordered magnitude of seasonal effect for each analyte represented by the sum of the absolute values of each season's contribution to analyte variation, in units of analyte-specific standard deviation.

Analyte	Seasonality
Strontium, dissolved	3.907
Hardness as CaCO3, D	3.818
Antimony, total	3.298
Selenium, total	3.276
Silver, total	3.229
Sodium, dissolved	3.167
Vanadium, total	3.122
Calcium, total	2.966
Sodium, total	2.94
Silica, dissolved	2.729
Barium, total	2.643
Potassium, dissolved	2.516
Total Alkalinity	2.381
Calcium, dissolved	2.32
TDS, measured	2.299
Arsenic, total	2.129
Thallium, total	2.032
Thallium, dissolved	1.97
Vanadium, dissolved	1.891
Arsenic, dissolved	1.782
Iron, dissolved	1.676
Molybdenum, dissolved	1.583
Barium, dissolved	1.396
Molybdenum, total	1.349
Temperature	1.343
Beryllium, total	1.328
Magnesium, dissolved	1.325
Beryllium, dissolved	1.263
Magnesium, total	1.26
Antimony, dissolved	1.108
Selenium, dissolved	1.091
Nickel, total	1.05
Silver, dissolved	0.9882
Chromium, dissolved	0.9464
pH	0.9158
Potassium, total	0.7995
Specific Conductance	0.4925
Lead, dissolved	0.4497

Analyte	Seasonality
ORP	0.4384
Chromium, total	0.3878
Lead, total	0.3876
Turbidity	0.3119
Iron, total	0.2076
Zinc, dissolved	0.1607
Copper, total	0.101
Zinc, total	0.07999
Cadmium, dissolved	0.07878
Copper, dissolved	0.06025
Cobalt, dissolved	0.04927
Cobalt, total	0.04807
Lithium, dissolved	0.04093
Manganese, dissolved	0.02261
Oxygen, Dissolved	1.917e-15
Aluminum, dissolved	0
Aluminum, total	0
Cadmium, total	0
Manganese, total	0
Nickel, dissolved	0

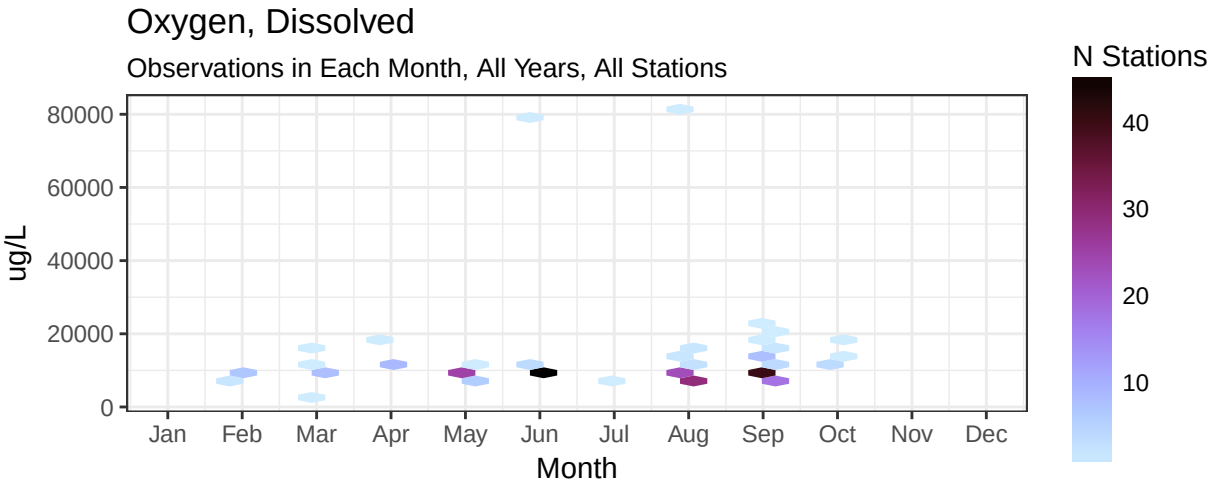
To sanity-check this process, let's take a look at the plots of observations in each month of the year, for a highly seasonal analyte:

```
plot_by_month("Strontium, dissolved")
```



And, on the other side, an analyte with lite evidence of seasonal variation:

```
plot_by_month("Oxygen, Dissolved")
```



Based on the above plots, this analysis passes the gut-check for me– dissolved strontium, the leader in seasonality using this index, seems to have a strong pattern of lower values in the summer months, while dissolved oxygen seems to mostly hover around the same values year round, with noise distributed throughout.

This analysis has limitations. Although the model fit for most of the analytes, some of the model resulted in 'singular fits', in which the estimated variance for the season and year random effects was 0. This stats.stacksexchange post[2] goes into this circumstance in detail– from the brief investigations I did, it appears this probably resulted from small numbers of random effect categories, and sparse data with each category. Better estimates of the magnitude of seasonal variation for each analyte could be gleaned from a data set that spanned more seasonal cycles, and had better observations evenly across the full time period. With only 4 observations in the winter months for many of the analytes, and some analytes without observations in all 4 seasons, modeling options like periodic functions and cubic splines seemed sub-optimal, but might have yielded different results in a more complete data set. This analysis does not describe how the variance/standard deviation in analytes might change with the seasons, and is currently limited to a narrow view of what constitutes "seasons"- for any given analyte, it's possible that different groups of months would be more appropriate and better represent trends within a year than the months I have chosen.

An initial stab at this model also included the sampling station such that the model would have a random intercept for each station as well– however, insufficient data for many of the observation sites (on the order of 2-3 observations for the full time period for Temperature, for instance) makes inference about the individual station's expected deviation from the mean dubious, and including them seemed like an excessive number of parameters to fit for this exercise. Knowing that certain sites group together might be one way to improve this– for instance, if all of the "FE-ET" stations could be considered together, it would likely improve the model's estimate, and would better disentangle variation due to site-specific conditions rather than seasonality or annual differences.

Other potential strategies and false starts along the way

To determine how much seasons "matter" for each analyte, I originally compared the observed data with and "without" seasonality, by de-trending the data of its seasonal component. This could be achieved by subtracting the model estimated intercept shifts for each season from the observed data points—functionally feeding the model new data to 'predict' values, in which the new, season-free data belongs to none of the seasonal sub-groups that the model was fit on. Calculating a summary measure, like the mean percent difference between the raw and seasonally-adjusted data, would have provided a measure of how much seasonality impacted the specific measurements taken within this dataset. However, this method would be sensitive to how many data points were observed in each season– for example, if there was a strong seasonal effect for the spring, but the data were mostly collected in the summer and fall, with few points in the spring

[2] https://stats.stackexchange.com/questions/115090/why-do-i-get-zero-variance-of-a-random-effect-in-my-mixed-model-despite-some-va

and winter, that analyte would appear to be far less seasonal than an analyte with moderate seasonality, but focused in summer, when much of the data were taken.

I also considered other model types to capture both long-term (annual) and short-term (seasonal) trends. Fitting models including some kind of periodic function, or using cubic splines, seemed initially attractive– however, without observations in winter for many of the analytes, this did not seem like the right path.