IEP Workshop Training: Generalized Additive Models (GAMs)

Jereme W Gaeta, PhD

3/23/2022

1 Recommended Reference Material

I recommend several books and book chapters to dig deeper into the underlying theory and application of Generalized Additive Models (GAMs) to ecological data. The most thorough book is probably (Zuur 2012); unfortunately, this is also the hardest book to acquire (I recommend inter-library loan). While less thorough, (Zuur et al. 2007) is a great introductory book to a wide range of common statistical analyses; Chapter 7 is a robust introduction to GAMs and Chapters 20, 21, and 22 contain case studies of GAMs being applied to ecological data. Finally, one of my top two go-to resources for advanced regression analyses is (Zuur et al. 2009). I highly recommend this book as an invaluable resource for anyone that regularly analyzes data as a part of their work duties. While this book focuses on mixed effects modeling, Zuur et al. (2009) do a great job of introducing the underlying models, including a wonderful introduction to GAMs in Chapter 3. FYI, my other favorite resource for regression is Gelman and Hill (2006), but they do not cover GAMs; so, I will not be referencing it here.

2 Background

Additive modeling is appropriate when your response and primary predictor variables are continuous and their relationship is non-linear **and** non-linearizable (Figure 1). That is, I always recommend trying to linearize the relationship among variables by trying a suite of data transformations on the response, predictor, or both (i.e., natural logarithmic, square-root, inverse, cubic, etc...). However, when your data clearly exhibit a non-linear pattern, additive modeling is, generally speaking, a great option.

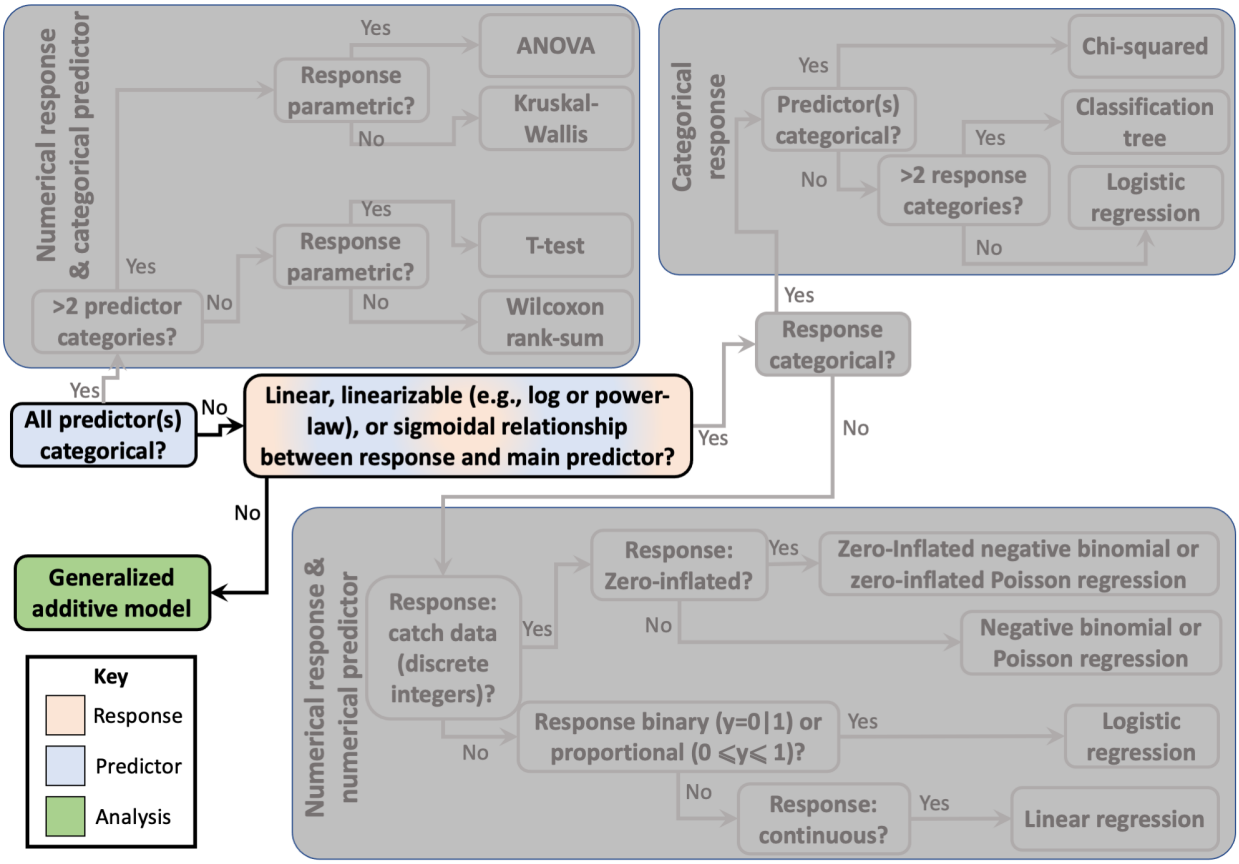

Figure 1: Univariate statistical analysis flowchart with generalized additive modeling highlighted.

2.1 The Assumptions (i.e., requirements) of Regression

Given modern computing power, open source software (e.g., R statistical software), and a plethora of peer-reviewed and non-peer-reviewed resources available on the Googles, we ecologists all too often throw data at an analysis without a deep understanding of the underlying statistical theory. While this micro-training session will not go too deep into the underpinnings of regression, I do want to briefly remind you of the four main assumptions (i.e., requirements) of regression: (1) normality, (2) homogeneity, (3) fixed x, and (4) independence (refer to Zuur et al. (2009) sections 2.3 or Zuur et al. (2007) pages 50-53 for a more thorough discussion).

Disclaimer: It is important to remember that regression is most suited for highly controlled data such as those derived from a physics or engineering laboratory setting. We ecologists will *always* have more variability and uncertainty associated with our data. Therefore, while we should strive to always meet the assumptions of regression, it is understood by quantitative ecologists and peer-reviewers that ecologists will have to *flex* these assumptions.

2.1.1 Normality:

Regression users often believe this assumption means your response variable y must be normally distributed. While that is *part* of the assumption of normality, that is not the core its meaning. This assumption means that the replicates or sub-samples values y would be normally distributed at a given value of x, and this would hold true for **all** values of x. However, we ecologists rarely have numerous sub-samples at each value

of x to test for normality. Therefore, our best option is to build a regression and evaluate the normality of the pooled residuals. This is why you should always include a histogram of model residuals when presenting model results.

2.1.2 Homogeneity:

This assumption is tightly linked with the assumption of normality. Specifically, this is the assumption that the distribution of your residuals hold constant across x (see the green box in the next section and Figure 2 for more information). This is why you should include a plot of model residuals (y-axis) against observed values of your predictor variable (x-axis). The plot should look like a rectangle with the density of residuals increasing as you approach zero along the y-axis. A clear violation of this assumption would be your residuals changing shape as you move left to right (imagine a traffic cone on its side or visible curve).

2.1.3 Fixed X:

This assumption is really tricky for ecologists. Remember, regression is most suited for highly controlled data such as those derived from a physics or engineering laboratory setting. The assumption of fixed X has two components: (1) that you *a priori* know each value of x (e.g., you design an aquarium experiment and *a priori* decide the water temperatures for the experiment) and (2) you know the exact value of x (i.e., no measurement error). Neither of these components are ever *truly* met in field ecology. So, it is understood that, generally speaking, we ecologists slightly bend this assumption. You should really only be concerned if the measurement error around x is very large relative to the range of x in your study (e.g., if you are aging Delta Smelt that are usually $\leqslant 3$ years old and your measurement error is ± 2 years).

2.1.4 Independence:

The final assumption of regression is that your observation of y at a given value of x cannot be influenced by or related to another observation of y at a different value of x. This assumption is most commonly violated by time series data and spatial data. Given a lot of ecological data are temporally or spatially related, we can account for this by including model terms that set the correlation structure. Run "*?nlme::corClasses*" in your R console for more information.

2.2 Model Notation

Personally, I believe you cannot fully understand a model without understanding the model notation. Do not run away yet, I do not mean you need to truly understand the math to the point of doing the analysis by hand on graph paper, I just mean the underlying equations.

For example, let's take that old equation from grade school:

$$y = mx + b \tag{1}$$

where y is your response variable, x is your predictor variable, m is the slope, and b is the intercept of a simple linear regression (are you having flashbacks of drawing a line on graph paper yet?). We can re-write equation 1 using the following standard mathematical notation:

$$\begin{gathered} y = \alpha + \beta x + \varepsilon \\ \varepsilon \sim N(0, \sigma^2) \end{gathered} \tag{2}$$

where y and x are your predictor and response variables, respectively, α and β are model (i.e., equation) coefficients (the same as b and m in equation 1, respectively), and ε is the remaining uncertainty not

represented by the model. Specifically, ε is the residual variance (i.e, the difference between the observations and the regression line created by the coefficients). The model uncertainty (ε) is described by the second line of the equation: the residual variance is normally distributed (N) with a mean of 0 and a variance of σ^2.

A Deeper Look at ε and the Assumption of Homogeneity

While ε may seem a little confusing, this is a simple, yet critical aspect of regression ecologists should understand. Specifically, this is related to the second assumption of regression: homogeneity. This assumption means the uncertainty, or noise, around the mean (i.e., where the mean is the linear regression model or prediction) must be constant across your predictor variable x. Consider the regression prediction (i.e., $\hat{y}$) at a given value of x, the observations of y must be normally distributed around $\hat{y}$ such that 68.3% of observations at that value of x must be $\hat{y} \pm 1 * \sigma^2$, 95.5% of observations at that value of x must be $\hat{y} \pm 2 * \sigma^2$, and 99.7% of observations at that value of x must be $\hat{y} \pm 3 * \sigma^2$. Because a linear regression model only produces one residual variance term (i.e., σ^2; the second part of equation 2), our observations must vary consistently around $\hat{y}$ for all values of x; that is, our data must have homogeneous variance (see Figure 2 for an illustration of this concept; refer to Zuur et al. (2009) sections 2.2 and 2.3.3 or Zuur et al. (2007) pages 50-57 for a more thorough discussion).

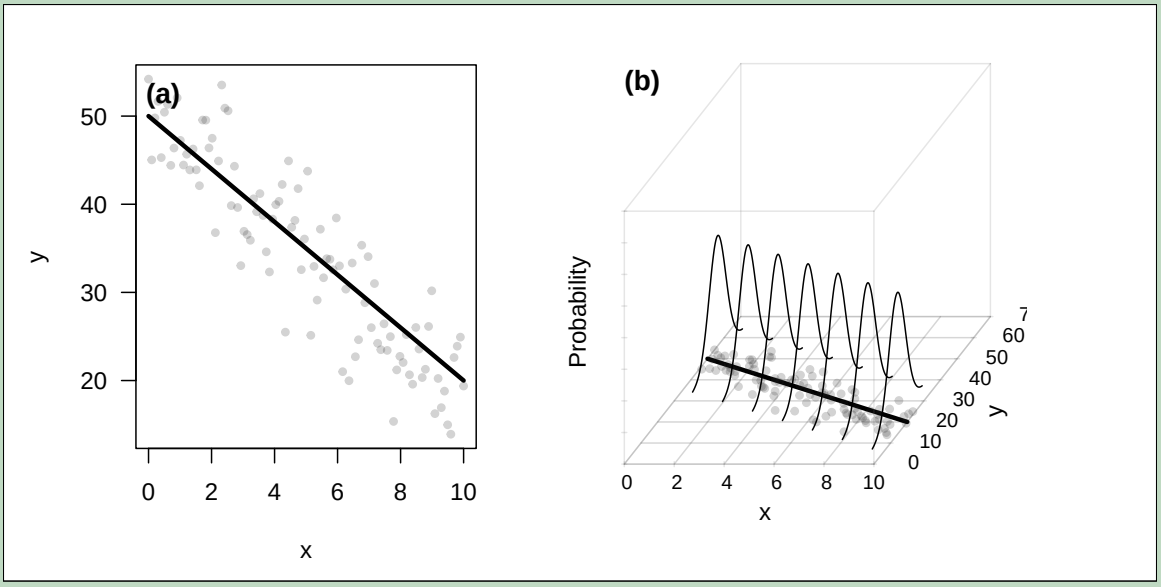

Figure 2: (a) A hypothetical dataset (points) shown with a linear regression model (black line). (b) The same hypothetical dataset (points) and linear regression model (black line), but shown with the several normal distributions across the range of x, each of which represents ε: a normal distribution around the regression line (i.e., a mean of zero) with a variance of σ^2.

In simple linear regression, α (i.e., the intercept coefficient) is the value of the response variable (y) when the predictor variable (x) is zero. β is the rate of change of y given a change in x of 1 (i.e., the slope coefficient). However, things get more complicated when the relationship between x and y is non-linear; we have to use a different model with a slightly different equation.

The notation for a GAM is as follows:

$$\begin{aligned} y &= \alpha + f(x) + \varepsilon \\ \varepsilon &\sim N(0, \sigma^2) \end{aligned} \tag{3}$$

where y and x are still your predictor and response variables, respectively, α is the model intercept (the same coefficient as in equation 2; although it no longer represents the value when x is zero, but rather the mean value of y), and ε is the remaining uncertainty that the model did not capture (i.e., residual variance).

$f(x)$ is a smoother function; essentially, it represents numerous underlying equations that vary depending on the *type* of smoother function you are using (e.g., cubic regression spline or thin-plate spline) and several other parameters. While this can seem daunting, R does all the work for you, leaving $f(x)$ as a black box (we will shine a little light into this black box in a moment). This black box, however, is one of the biggest drawbacks of additive models: additive models do not yield coefficients and, therefore, do not yield a useful equation like most regression models, meaning we can ***only plot*** model outputs (Zuur et al. 2009). That is, we cannot compare model coefficients to other studies nor can we report an equation that others could use to generate predictions. We can develop predictions, but we need the underlying data and code to do so.

To summarize, GAMs are a form of regression with a suite of unreportable coefficients in a 'black box' component of the model. However, this approach is still very powerful when the relationship you are analyzing is non-linear.

3 Analysis

I recommend always taking a question-driven approach to statistical and mechanistic modeling. With a good question comes brainstorming about the best data to address your question and a plan for the ideal statistical or mechanistic modeling approach. Regardless of the question, statistical method, or mechanistic modeling, you should always follow the same overarching framework presented in Figure 3: explore your data; apply a model; evaluate, validate, and optimize your model; and, when validation and optimization are complete, focus on a clear, concise interpretation and effective presentation. While every step in this general framework may not be relevant for every project or every analysis, returning to this framework will ensure you produce defensible and publishable science.

While, unfortunately, we will not have time during the workshop to walk through each step of the modeling framework presented in Figure 3, I will return to the figure and bring your attention to these concepts throughout this micro-training.

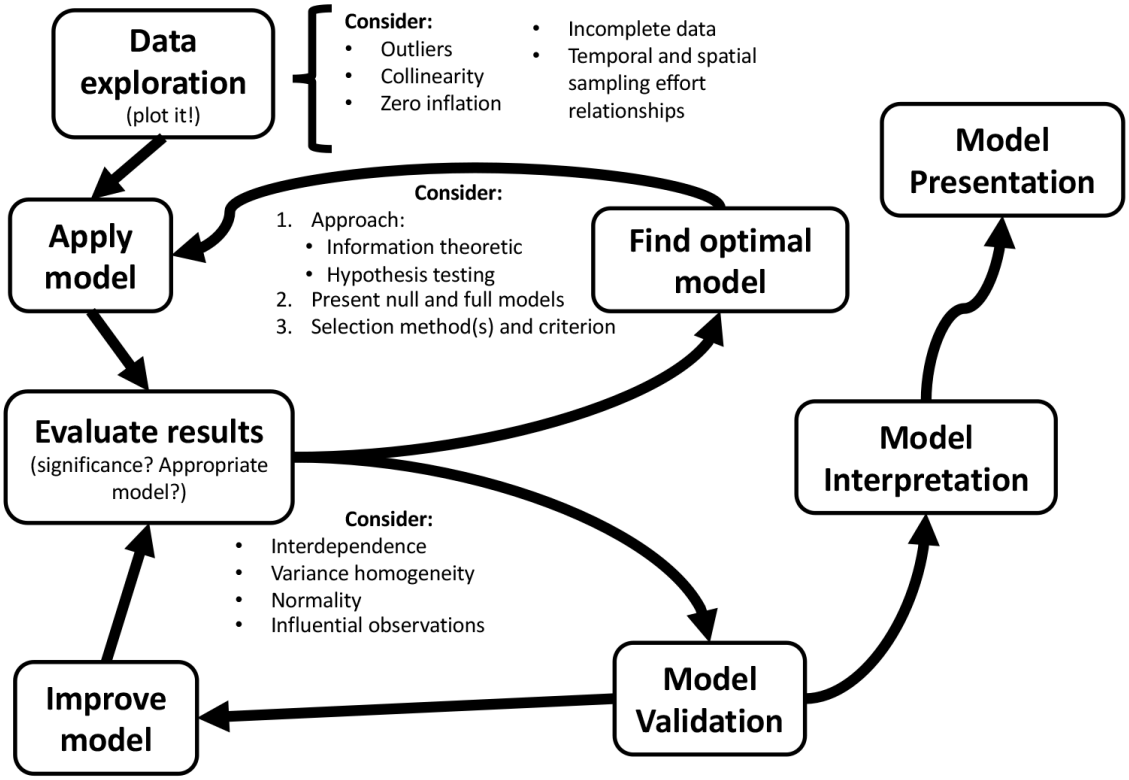

Figure 3: A general modeling framework I recommend following (at least loosely) for all analyses.

3.1 Load the R Packages, our Dataset, and Custom Functions

Prior to any analyses in R, you need to make sure you have the necessary packages loaded into the environment. Equally important, however, is making sure you do not have conflicting packages loaded. The ***gam*** package and ***mgcv*** package are both used for generalized additive modeling and both have a *gam()* function. However, the arguments, outputs, and flexibility of these two functions differ. We will be using the ***mgcv*** package for this micro-training session. However, we want to make sure the ***gam*** package is not loaded into your R environment, so let's detach this package prior to loading the others. NOTE: you will get an error if ***gam*** is not currently loaded; disregard this error.

```
#~ Load packages
detach(package:gam)
library(mgcv)
library(itsadug)
library(lubridate)
library(data.table)

#~ A custom function to calculate day of water year
water.day = function(x, start.month = 10L){
  start.yr = year(x) - (month(x) < start.month)
  start.date = make_date(start.yr, start.month, 1L)
  as.integer(x - start.date + 1L)
}

#~ A custom function to add a label to the same location on every panel
```

```
plot_label = function(lab="(a)", x_prop=0.08, y_prop=0.92,
                      font_type=2, fcex=1.15, usr=par('usr')){
  x_val = usr[1]+(usr[2]-usr[1])*x_prop
  y_val = usr[3]+(usr[4]-usr[3])*y_prop
  text(x = x_val, y = y_val, labels = lab, font = font_type, cex=fcex)
}

#~ Load the dataset
df = read.csv(file = "Yolo_tow_drain_drift_cpue.csv", header=T)
```

3.2 Case Study Data

I selected a publicly available dataset generated by the DWR's Yolo Bypass Fish Monitoring Program (YBFMP), which is available via the Environmental Data Initiative (IEP et al. 2021). The YBFMP collects aquatic macroinvertebrate drift data at the base of the Toe Drain. One of the variables they report are taxa-specific CPUE. For this micro-training, we will evaluate total aquatic macroinvertebrate CPUE across the day of water year (i.e., where the first day of water year is October 1).

Let's take a look at the dataset:

```
head(df)
```

```
##   FlowdiffAdj Secchi Conductivity VolumeAdj Inundation Station   WY water_doy
## 1       42366     NA           NA  130.1309       TRUE    STTD 1998       127
## 2       44093     NA           NA  135.4355       TRUE    STTD 1998       134
## 3       37812     NA           NA  116.1428       TRUE    STTD 1998       140
## 4       37871     NA           NA  116.3241       TRUE    STTD 1998       147
## 5       68946     NA           NA  211.7736       TRUE    STTD 1998       155
## 6       68946     NA           NA  211.7736       TRUE    STTD 1998       162
##   water_year WYClass         cpue fInun
## 1       1997       W 0.0227924425     1
## 2       1997       W 0.0184811241     1
## 3       1997       W 0.0744342011     1
## 4       1997       W 0.0222567865     1
## 5       1997       W 0.0049959004     1
## 6       1997       W 0.0002927654     1
```

Abbreviated metadata for the variables are as follows:

- *Inundation*: whether the flood plain was inundated
- *Station*: Station ID; the data has been reduced to include just the "STTD" station, which is the location of the rotary screw trap at the Toe Drain
- *water_doy*: day of water year; water year begins on October 1
- *water_year*: water year
- *cpue*: total CPUE of aquatic macroinvertebrates collected on a given day
- *fInun*: the *Inundation* variable converted into a binary factor

3.3 Study Question

The study question we will address in this micro-training is ***"does the amount of aquatic macroinvertebrate drift differ throughout the water year and, if so, does it vary with floodplain inundation?"***

The overarching goal, therefore, of our analysis is to test whether aquatic macroinvertebrate CPUE varies throughout the water year and, if so, whether the relationship differs if the floodplain is inundated. As per our modeling framework (Figure 3), let's start by exploring the distribution of the response variable as this will inform our analytical options. The response variable is strongly skewed (Figure 4a), but a $\log_e$-transformation normalized the variable (Figure 4b). Since the data are normalizable (i.e., can be transformed into a normal distribution), we can proceed with relative ease. If our data were not normalizable, we would need to explore alternative distributions, such as gamma, negative binomial, etc...

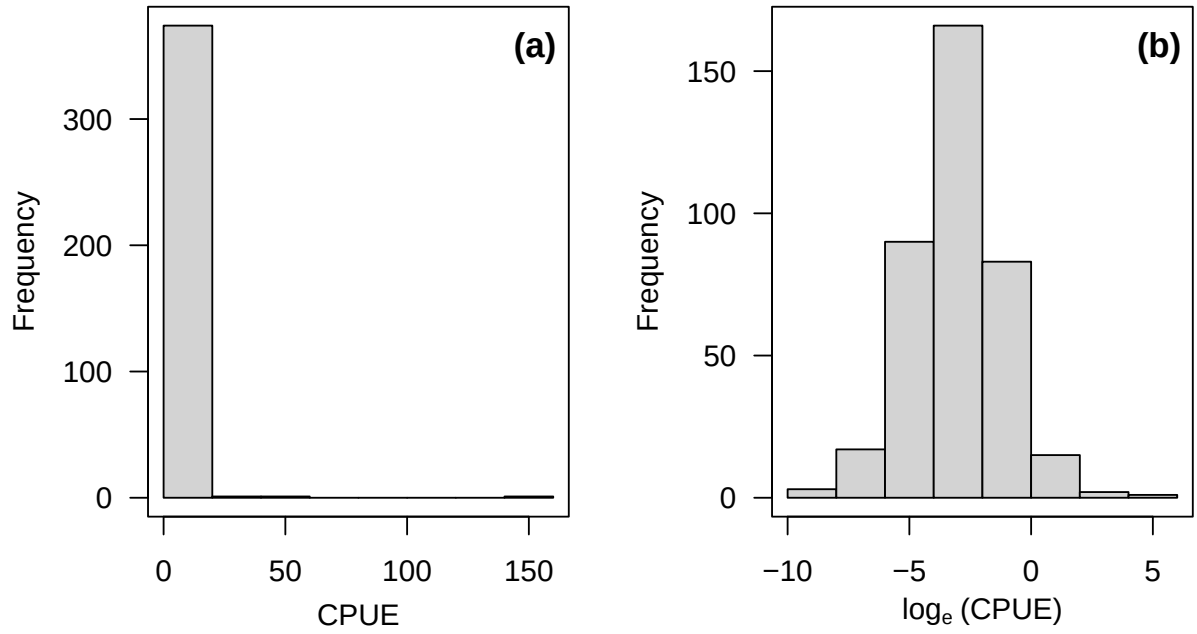

Figure 4: Macroinvertebrate drift sample CPUE at the Yolo Bypass Toe Drain. Shown as (a) raw data and (b) $\log_e$-transformed.

To address the first part of the analysis goal, let's look at the CPUE data across day of water year (Figure 5a). First, you can see that $\log_e$-transformation of the CPUE data really is necessary (Figure 5a & b). Since linear regression is a better option than a GAM, I applied a linear model to the data and evaluated the results *a la* Figure 3. To do so, I plotted the prediction of the linear model onto a plot of the data (black line in Figure 5b). A plot of the residuals across the predictor variable reveals strong pattern in the residuals, indicated the data in their current form are not suitable for a linear regression model (at least not without additional covariates; Figure 5c).

Following the general modeling framework (Figure 3), this tells us we need to apply a different modeling approach. In our case, we will pursue a GAM.

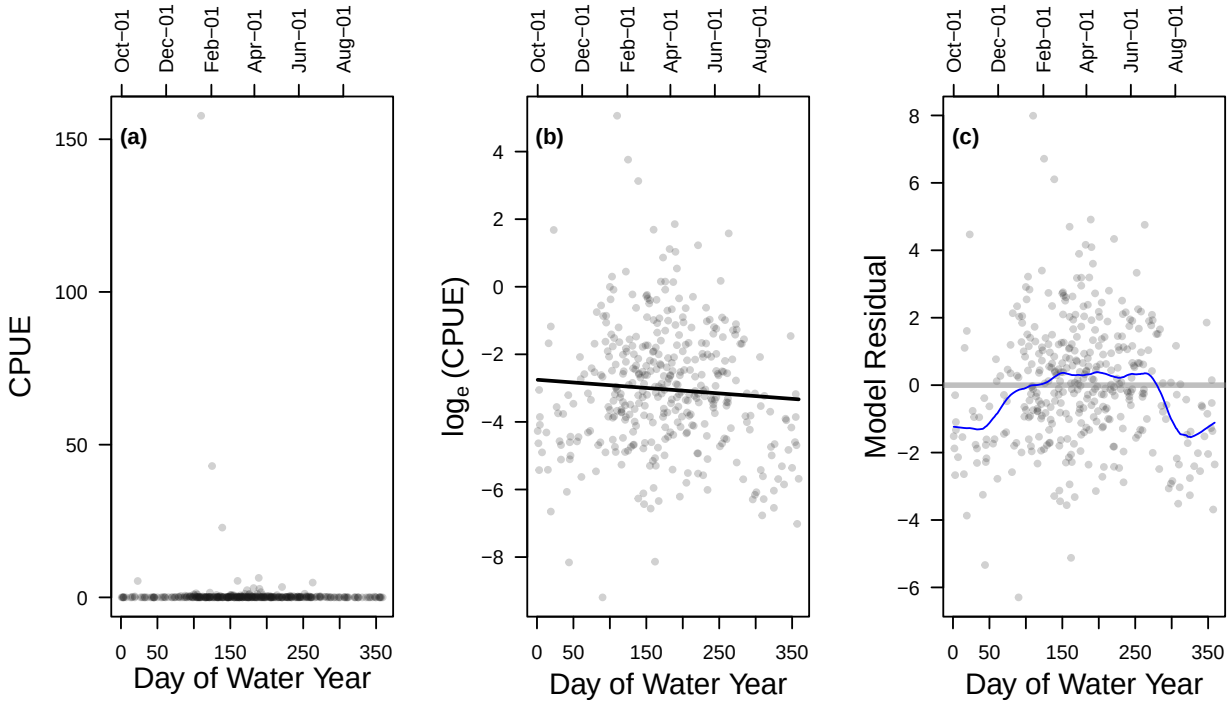

Figure 5: Macroinvertebrate drift sample CPUE across day of water year in the Yolo Bypass Toe Drain. Shown as (a) raw data, (b) $\log_e$-transformed CPUE data with linear regression model (black line), and (c) linear regression residuals (shown with an automated smoother; blue line).

Now that we know the *cpue* variable needs to be $\log_e$-transformed prior to analysis, let's create and add a variable called *ln_cpue* to the dataset:

```
df$ln_cpue = log(df$cpue)
```

3.4 Analysis Time Out: GAMs Under the Hood

When I first started exploring GAMs, my first question was simply *"how do they do that?"* While the details vary among algorithms, the overall concept is universal. First, the models subset the data across the predictor variable (x-axis), the model then builds a regression for each subset of data (Figure 6) and then "stitches" those regression subsets together into a smooth line (Figure 7).

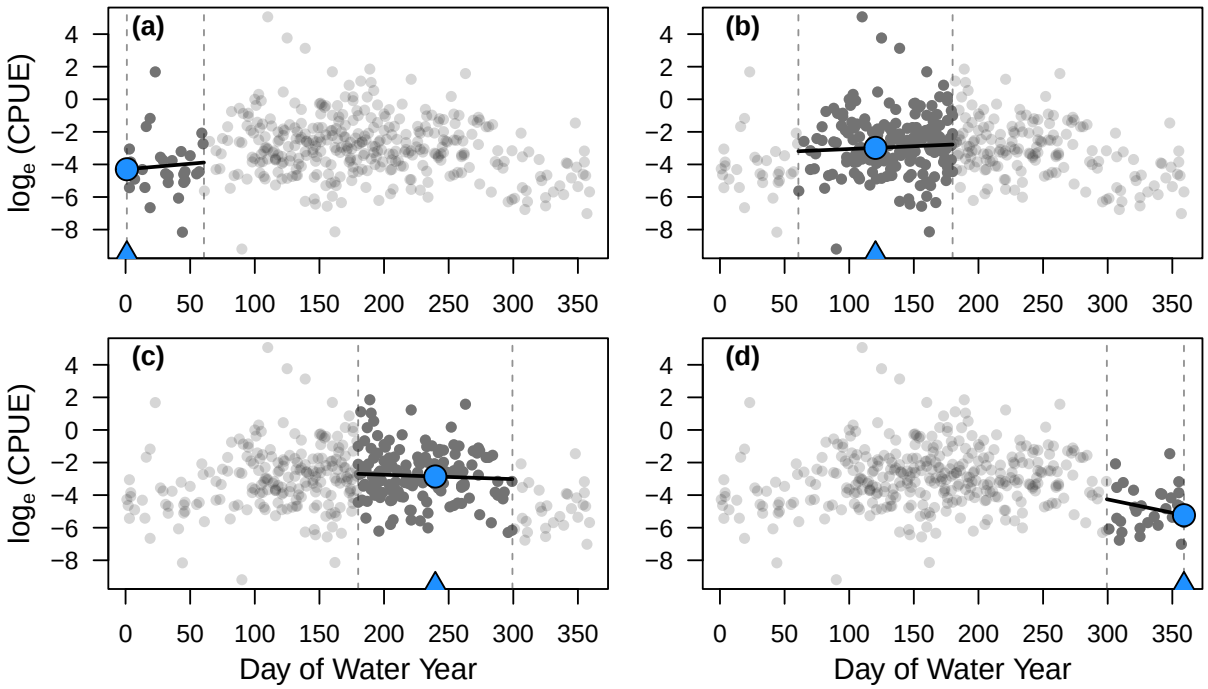

Figure 6: A simplified illustration of how Generalized Additive Models create non-linear patterns by breaking the predictor variable (i.e., Day of Water Year) into sections based on window length (a-d) and generating regressions with the subsets. The vertical dash lines denote the subset window ranges, the dark points indicate the data within the subset, the black line is the subset regression, the blue trangle is the x-axis mid-point of the window, and the blue point is the regression estimate at the mid point.

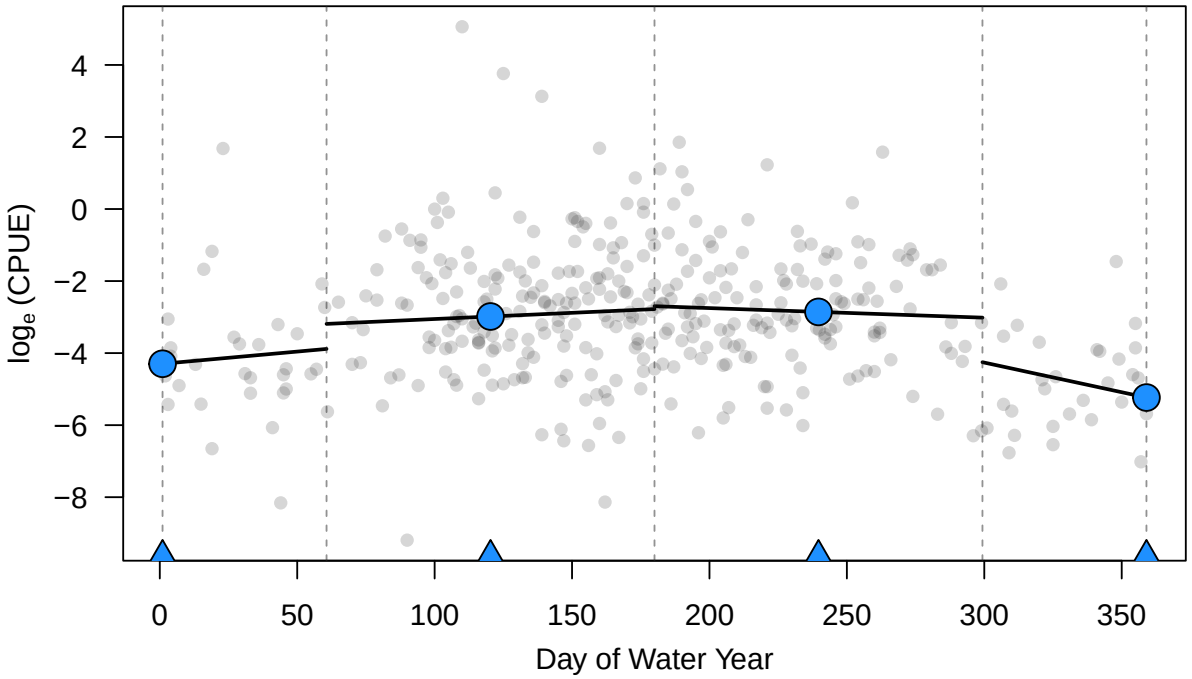

Figure 7: A simplified illustration of how Generalized Additive Models create non-linear patterns by breaking the predictor variable (i.e., Day of Water Year) into sections based on window length (see the previous figure) with the subset regressions stitched together. The vertical dash lines denote the subset window ranges, the black line is the subset regression, the blue trangle is the mid-point of the window, and the blue point is the regression estimate at the mid point.

You can control the underlying algorithm by changing the type of regression that is used and, in some cases, by controlling the polynomial order. For example, Figure 6 & 7 use a first-order polynomial (AKA linear regression; equations 1 & 2). The equations for a second- (equation 4) and a third-order (equation 5) polynomial regression would be as follows (respectively):

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \varepsilon \tag{4}$$
$$\varepsilon \sim N(0, \sigma^2)$$

$$y = \alpha + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \varepsilon \tag{5}$$
$$\varepsilon \sim N(0, \sigma^2)$$

where the predictor variable (x; day of water year, in our case) would be squared (equation 4) or squared and cubed (equation 5), resulting in increased "wiggliness" (see Figure 8).

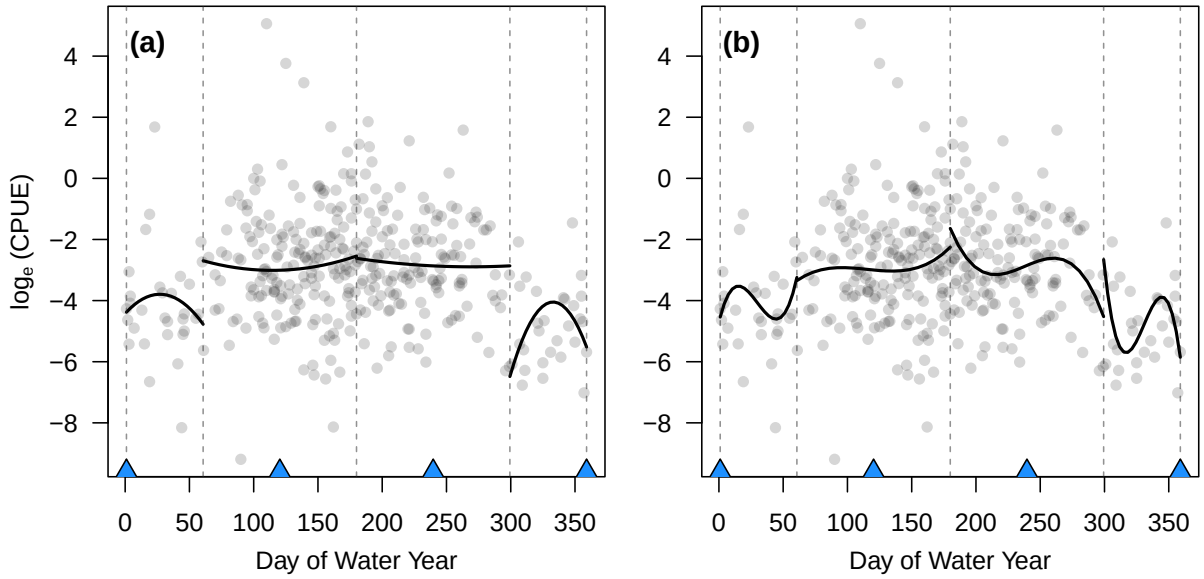

Figure 8: A simplified illustration of how Generalized Additive Models create non-linear patterns by breaking the predictor variable (i.e., Day of Water Year) into sections based on window length (see the previous figures) with the subset regressions stitched together. In this case, the subset regressions are (a) second-order polynomials and (b) third-order polynomials. The vertical dash lines denote the subset window ranges, the black line is the subset regression, and the blue trangle is the mid-point of the window.

In addition to the kind of polynomial the model uses (e.g., first-, second-, or third-order polynomial), the window length can also influence how the model fits the data (Figure 9).

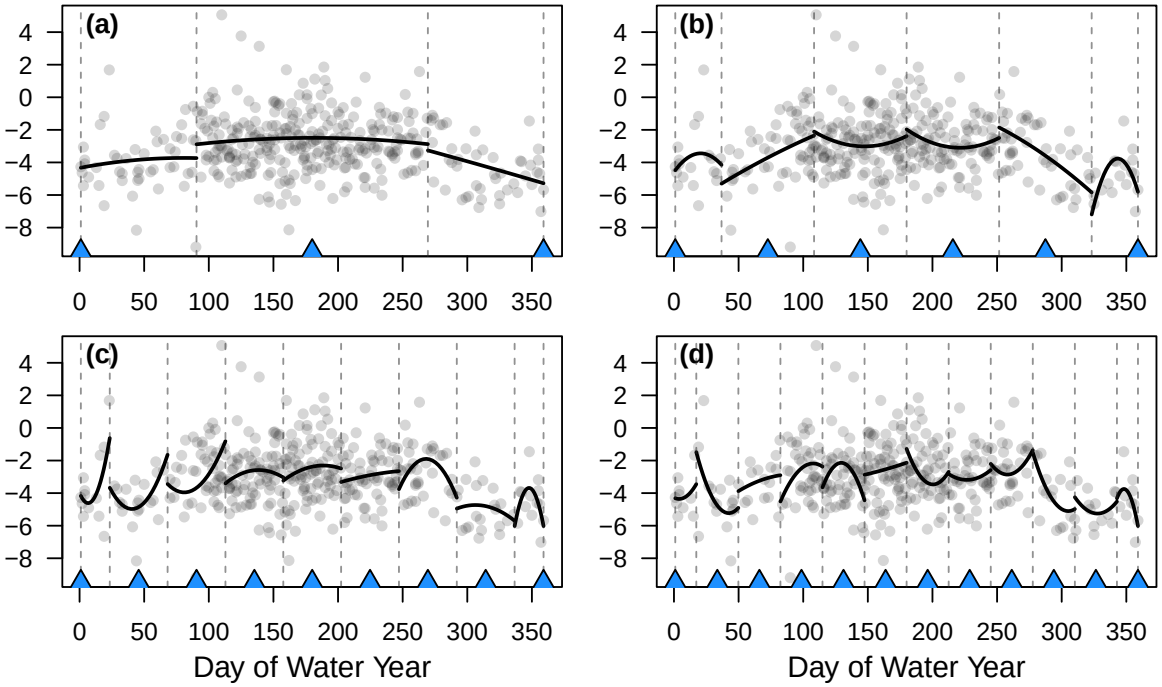

Figure 9: A simplified illustration of how Generalized Additive Models create non-linear patterns by breaking the predictor variable (i.e., Day of Water Year) into sections based on window length with the subset regressions stitched together. In this example, the subset regressions are second-order polynomials with varying window lengths of (a) 179 days, (b) 72 days, (c) 45 days, or (d) 33 days. The vertical dash lines denote the subset window ranges, the black line is the subset regression, and the blue trangle is the mid-point of the window.

Take Note:

At this point, I hope you have noticed three important aspects of this process:

1) The ends of the subset regression lines in Figures 7, 8, & 9 do not align perfectly
2) The window length can strongly influence the observed pattern
3) The ends of the data x-axis range have far less data informing the regressions

Regarding 1) above, the GAM uses a little calculus (derivatives) to force the lines to converge at the breaks (please see Zuur et al. (2009) Chapter 3, pages 46-51 for more information). In GAM-speak, these break points (i.e., vertical dashed lines in Figures 6, 7, 8, & 9) are called knots (argument k= in the ***gam()*** function). Occasionally, you may get a warning asking you to reduce k; this is usually due to a low sample size (i.e., a small dataset). Otherwise, you will usually want to leave k at the default setting of *k=-1*. This tells the algorithm to use cross-validation to optimize k. For more information on this topic, see pages 51-53 in Zuur et al. (2009).

Regarding 2) above, the window length is directly driven by k. As stated above, the algorithm uses cross-validation to optimize k, so you do not need to worry about this argument unless the model "throws" a warning. Again, see pages 51-53 in Zuur et al. (2009) for more information on cross-validation. To highlight the effect of k (even though you I recommend you not manually set k unless necessary), see Figure 10.

Regarding 3) above (i.e., less data at the extremes), this is why the uncertainty in GAM predictions expands at the data ranges: less data = less confidence in the prediction (e.g., see the inflation in the confidence interval at the extremes of the x-axis in Figure 10). The literature, therefore, strongly recommends caution when discussing GAM model predictions at data extremes. Generally, I will not even show more predictions at the data extremes by constraining predictions from the 5^{th}- to the 95^{th}-percentile of the range of the predictor variable (sometimes I even constrain from the 10^{th}- to the 90^{th}-percentile).

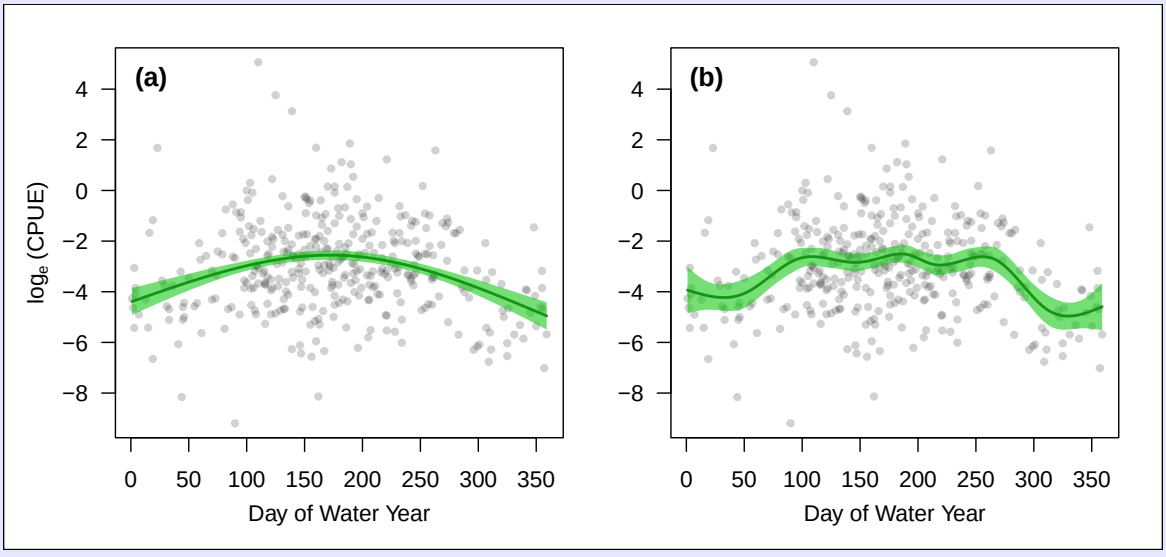

Figure 10: A generalized additve model fit using (a) 3 and (b) 12 knots.

3.5 Building a GAM with a Single Smoother

Let's return to the general modeling framework (Figure 3). We tried a linear regression and realized this was not the optimal model for the data at hand (Figure 5b & c). Now that you have a general idea about what the GAM algorithm is doing under the hood, let's apply our data to a GAM! We will be using the *gam()* function from the "***mgcv***" package to develop our GAM model. The function has three main components:

1. The formula of the model you want to develop, including the smoother function
    a. The smoother function, *s()*, will have several components:
        i. the variable across which to smooth (*water_doy* in our case)
        ii. the kind of smoother you wish to use, defined by the *bs=* argument
            * folks commonly use *tp* for thin plate spline or *cr* for cubic spline
        iii. (used later) the grouping variable of which to develop unique smoother functions; defined by the *by=* argument
2. The dataset from which the variables are pulled
    a. This is defined by the argument *data=*
3. The type of distribution your data follows, defined by the *family=* argument
    a. This would be necessary if our data were not normalizable
    b. Fortunately, the $\log_e$-transformation normalized the response variable (Figure 4), so we can use the default of *family=gaussian()* where "Gaussian" is mathematician speak for "normally distributed" or "bell-shaped curve"

Reminder: a package called "***gam***" also has a *gam()* function and I recommend running *detach(package:gam)* prior to loading the "*mgcv*" package.

```
mod = gam(ln_cpue ~ s(water_doy, bs = "cr"), data = df)
summary(mod)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## ln_cpue ~ s(water_doy, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.03509    0.09033   -33.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df     F p-value
## s(water_doy) 8.141  8.775 6.346  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.122   Deviance explained = 14.1%
## GCV = 3.1522  Scale est. = 3.0758    n = 377
```

The *summary()* function yields an output with several components including "**Parametric coefficients**" and "**Approximate significance of smooth terms**." The **Parametric coefficients** includes only one parameter: the intercept. Where the intercept in linear regression is the value of y when x is zero, the intercept for a GAM is the mean value of y in the dataset. The **Approximate significance of smooth terms** tells us information about the "black box" smoother. Specifically, the "*edf*" is the effective degrees of freedom, or, in layperson's terms, the wiggliness of the smoother function where the greater the *edf* the greater the wiggle. The second to last row of information in the summary output reports the "*Deviance explained*," which is the GAM equivalent of an R^2. The final line reports the "*Scale est.*"; this is the GAM value for residual variance. Applying the model summary to equation 3 would result in the following:

$$y = -3.04 + f(x) + \varepsilon \tag{6}$$
$$\varepsilon \sim N(0, 3.08)$$

3.6 Evaluating Results of a GAM with a Single Smoother

Now that we have applied a model to our data, we need to evaluate the results (Figure 3).The first step when evaluating model results should be to evaluate fit visually via various residual plots. However, the first thing I usually do is to plot the model outcome (because I shake presents at Christmas). Let's look at the default ***mgcv*** package plot via the *plot()* function:

```
plot(mod)
```

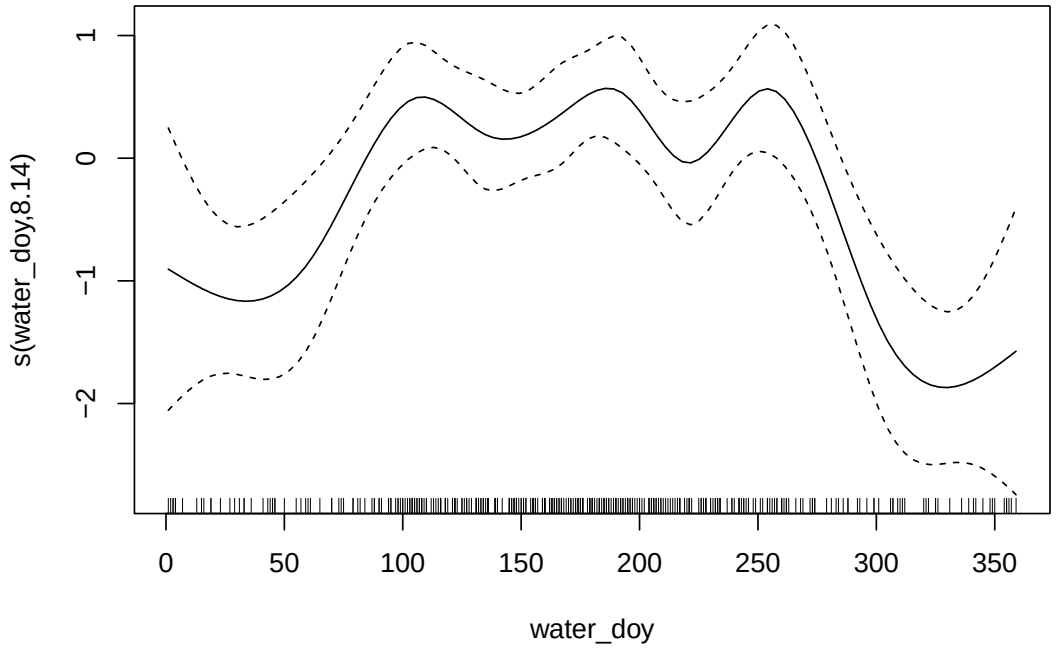

Figure 11: Default GAM plot produced in R showing model prediction (relative to the intercept) across day of water year.

Default GAM plots are unique in a few ways:

1. The plot automatically adds a rug plot to the bottom of the y axis. These are little ticks inside the plot that represent the x-vales (water day of year or 'water_doy' in our case) at which we have observations.

2. The y-axis is centered around the intercept coefficient (i.e., α in 3)
3. The y-axis label conveys the smoother term and includes a number that is the *edf* or "*wiggliness*"

Now that our curiosity is satiated, let's evaluate model fit properly. This includes a plot of observations across predicted values (Figure 12a) and several residual plots. Residuals are the deviation of the observation from the model prediction. To evaluate the assumption of normality, we need to plot a histogram of the residuals (Figure 12b), and we need to plot the residuals across the predictor variable to evaluate the assumption of homogeneity (Figure 12c):

```
layout(mat = matrix(c(1,2,3,3), 2, 2, byrow=TRUE))
par(mar=c(4.5,4.5,1.5,1.5)+0.1, oma=rep(0,4))
plot(df$ln_cpue ~ predict(mod), las=1, pch=20, col=gray(0.2,0.2))
abline(0,1)
plot_label(lab = "(a)")
hist(residuals(mod), las=1, main=""); box(which="plot")
plot_label(lab = "(b)")
plot(residuals(mod) ~ df$water_doy, las=1, pch=20, col=gray(0.2,0.2))
abline(h=0)
plot_label(lab = "(c)")
```

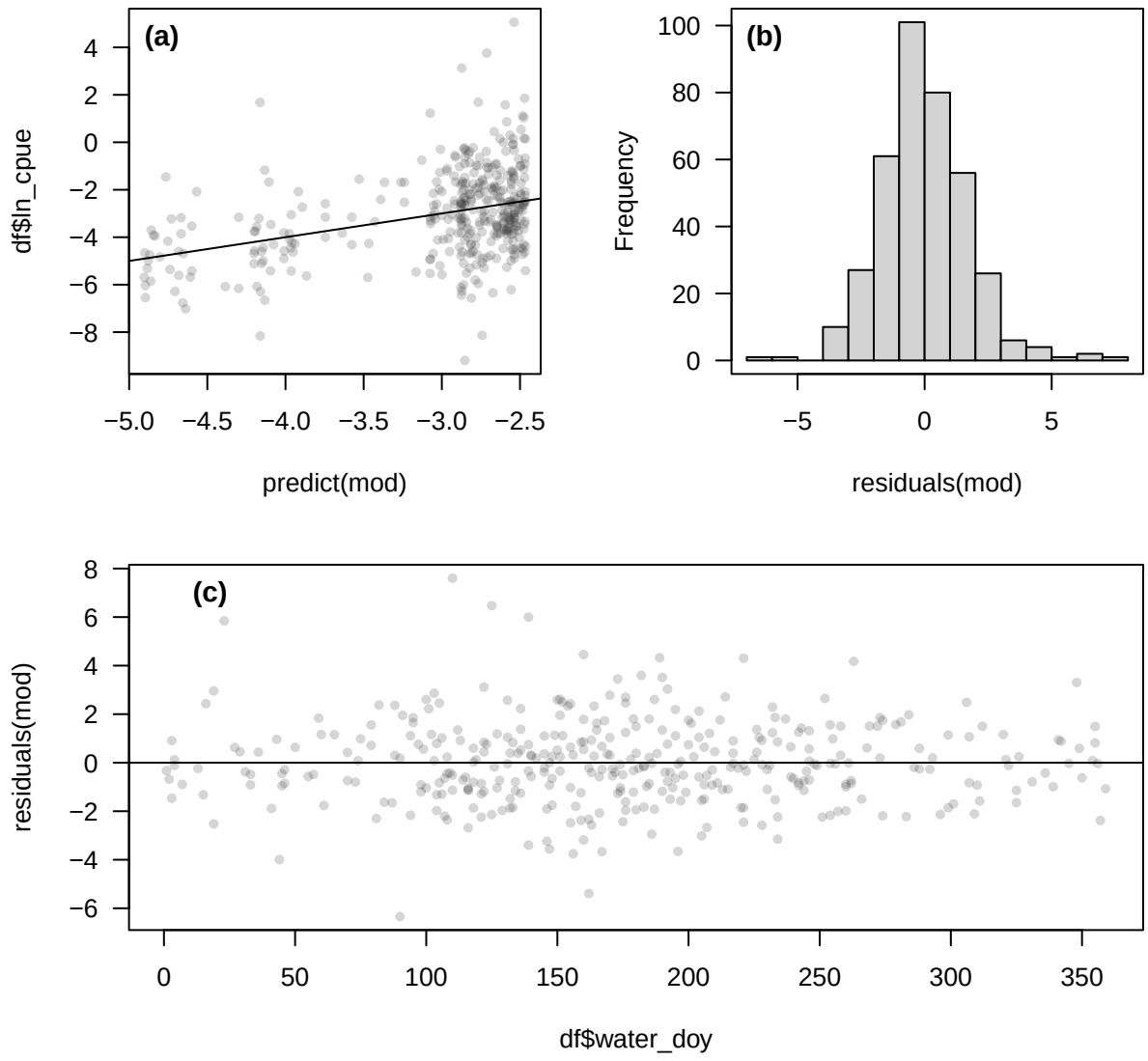

Figure 12: Model fit plot including (a) observed across predicted shown with a 1:1 line, (b) a histogram of model residuals, and (c) model residuals across the predictor variable shown with a horizontal line at a residual value of zero.

Generally speaking, I would call this a successful, albeit noisy, model fit (but noisy data result in noisy models). First, let's look at the plot of observed across predicted (Figure 12a): we see that while the data are noisy, the data appear to be fairly evenly distributed above and below the 1:1 line across the range of predictions. The histogram of residuals (Figure 12b) appears to be fairly normal. While we could apply tests to statistically assess normality (e.g., a Shapiro-Wilk test), ecologists generally accept that the eye-ball test is suitable. Finally, we can see that the model meets the assumption of homogeneity with the variability of residuals consistent across the predictor variable (Figure 12c).

3.7 Visually Presenting a GAM with a Single Smoother

Referring to Figure 3, we have validated the model and concluded the model is appropriate. Now we need to move on to model interpretation and presentation. To effectively present the model, however, we need

a better plot than the default *plot()* function (i.e., Figure 11). To develop a presentable plot, we will need to generate model predictions across the range of the predictor variable and then plot these predictions on the same plot as the underlying data. My preferred approach to generating model predictions is to use the *predict()* function. I prefer this to several black box functions as I know what is happening under the hood, so to speak. We will take the following steps to develop model predictions:

1. Determine the confidence interval to present in your figure and determine the t-distribution value of which to multiply the standard error
    i. We can use the t-distribution to convert standard error into a confidence interval
2. Generate a data frame with the sequence from the minimum to the maximum observed value of the predictor variable: day of water year
3. Use the *predict()* function to estimate model fit and standard error
4. Multiply the standard error fit by ± the t-distribution value to estimate the upper and lower confidence interval limits

```
# Determine the confidence interval to present in your figure and determine the
# t-distribution value of which to multiply the standard error
interval_value = 0.95
t_distribution_probability = 1.0 - (1-interval_value)/2
CI_factor = qt(t_distribution_probability, Inf)

#~ Generate a data frame with the sequence from the minimum to the maximum
# observed value of the predictor variable: day of water year
mod_doy_range = range(na.omit(df$water_doy))
new_dat_1 = data.frame(water_doy = seq(from = mod_doy_range[1],
                                       to = mod_doy_range[2], by=1))

#~ Use the *predict()* function to estimate model fit and standard error
preds = predict(mod, newdata = new_dat_1, se.fit = TRUE)

# Multiply the standard error fit by the t-distribution value to estimate
# the upper and lower confidence interval limits
fit = preds$fit
upper = fit-CI_factor*preds$se.fit
lower = fit+CI_factor*preds$se.fit

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#~~   Visualize the model

# Plot the raw data and make the axes and labels aesthetically pleasing
par(mfrow=c(1,1),mar=c(2.5, 4.5, 2,4.5)+0.1, oma=c(1.5,0,0.5,0))
plot(df$ln_cpue ~ df$water_doy, pch=20, col=gray(0.1,0.2),
     las=1, ylab="", xlab="")
mtext(text = "CPUE", side = 4, line = 3)
mtext(text = expression(log[e]~"(CPUE)"), side = 2, line = 2.5)
y_labs = c(0.0001,0.0003,0.001, 0.003,0.01,0.033,
           0.1,0.33, 1, 3, 8, 20, 55, 150)[c(T,F)]
axis(side = 4, at = log(y_labs), labels = y_labs, las=1)
mtext(text = "Day of Water Year", side = 1, line = 2.25)

# Use the polygon function to add the CI
polygon(x = c(new_dat_1$water_doy, max(new_dat_1$water_doy),
              rev(new_dat_1$water_doy), new_dat_1$water_doy[1]),
```

```
        y = c(lower, upper[length(upper)],
              rev(upper), lower[1]), border=NA,
        col=rgb(20,200,20,alpha=150,maxColorValue=255))

# Use the lines() function to add the mean model prediction
lines(x = new_dat_1$water_doy, y = fit, lwd=2,
      col=rgb(20,150,20,alpha=255,maxColorValue=255))
```

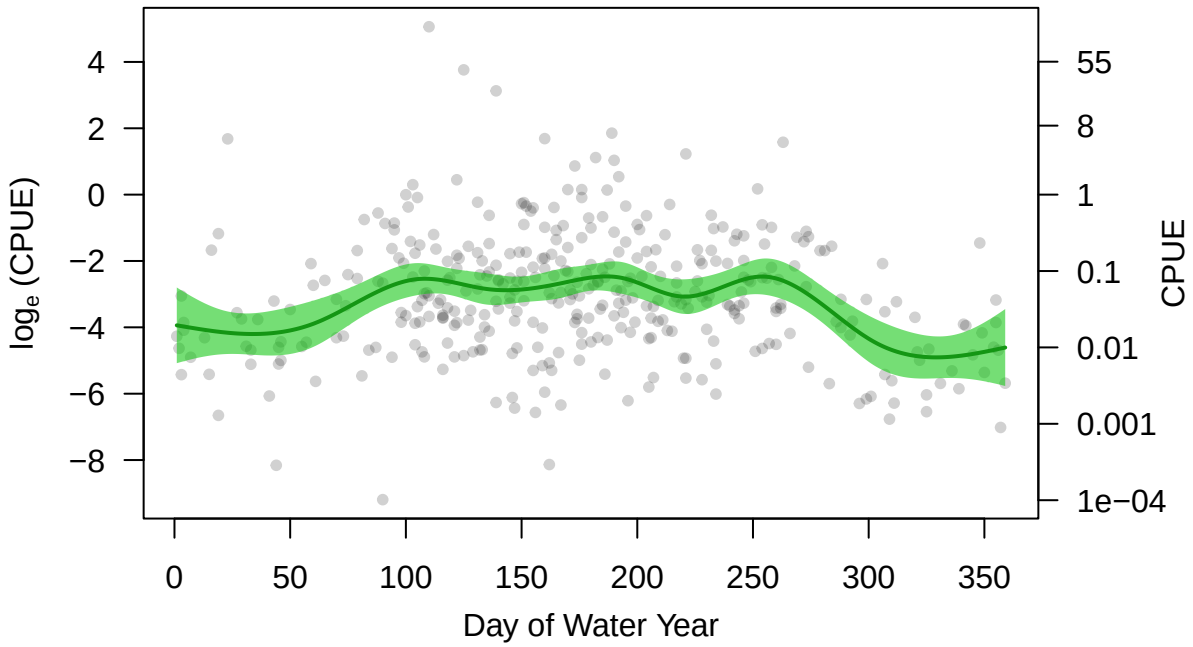

Figure 13: Model predicted $\log_e$-transformed aquatic macroinvertebrate CPUE (green line) across day of water year; shown with point-wise 95%-CI (green polygon) and raw data (points).

Is it time to declare victory? Well, let's go back to the initial analytical goal: *to test whether aquatic macroinvertebrate CPUE varies throughout the water year and, if so, whether the relationship differs if the floodplain is inundated.*

Well, the answer to the first part of our goal is clearly, ***yes***, but we have not addressed the second part of our goal. Let's start by looking at the data:

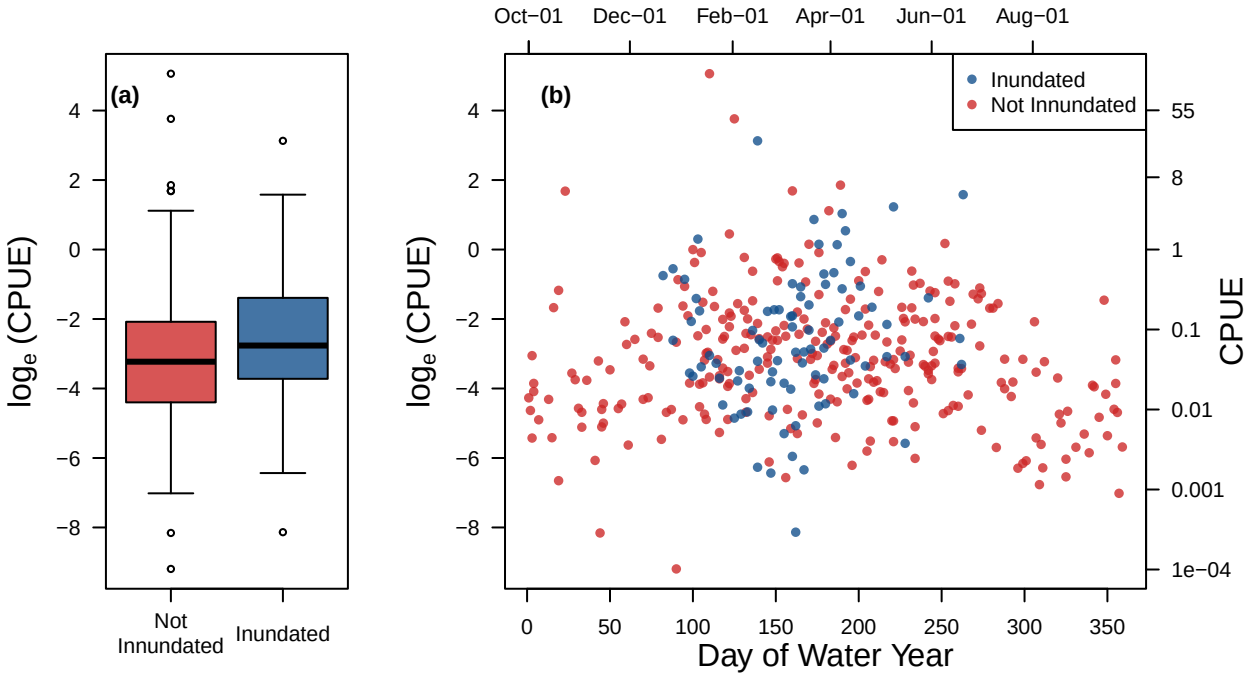

Figure 14: Log$_e$-transformed aquatic macroinvertebrate CPUE when the floodplain is and is not inundated shown (a) as a boxplot and (b) across day of water year.

The box-plot (Figure 14a) strongly suggests that CPUE is greater during period of inundation and the scatter plot (Figure 14b) indicates the floodplain is only ever inundated during winter, spring, and early summer (i.e., late-December through June).

3.8 Building a GAM with Multiple Smoothers

Our analytical goal (i.e., to test whether aquatic macroinvertebrate CPUE varies throughout the water year and, if so, whether the relationship differs if the floodplain is inundated) informs our model building process. Specifically, we are assessing CPUE across the day of water year (a continuous variable) when the floodplain is or is not inundated (a categorical variable that can be converted into a 2-level factor; "0" or "1" in our case). We will be leveraging the *by=* argument in the smoother function *s()* to build a model that meets our analytical goal. The *by=* argument allows us to "tell" the model that we want to allow for two smoothers: one for observations made during periods of inundation and one for observations made during periods without inundation. The model includes the following two components:

1. The factor-level predictor
2. A new smoother function that includes the *by=* argument

Let's look at the syntax:

```
# make sure fInun is, in fact, a factor
df$fInun = as.factor(df$fInun)

#build the model
```

```
mod2 = gam(ln_cpue ~ fInun + s(water_doy, bs = "cr", by = fInun),
           data = df)
```

Now let's compare the two model summaries:

```
summary(mod)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## ln_cpue ~ s(water_doy, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.03509    0.09033   -33.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df     F p-value
## s(water_doy) 8.141  8.775 6.346  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.122   Deviance explained = 14.1%
## GCV = 3.1522  Scale est. = 3.0758    n = 377
```

```
summary(mod2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## ln_cpue ~ fInun + s(water_doy, bs = "cr", by = fInun)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.0540     0.1032 -29.607   <2e-16 ***
## fInun1        1.2835     0.6989   1.837   0.0671 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                         edf Ref.df     F p-value
## s(water_doy):fInun0 6.948  7.946 6.668  <2e-16 ***
## s(water_doy):fInun1 3.987  4.766 2.834  0.0214 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## R-sq.(adj) =  0.153   Deviance explained =   18%
## GCV = 3.0735  Scale est. = 2.968     n = 377
```

Pay attention to two important takeaways from the model summaries:

1. The explained deviance (i.e., a GAM's R^2) increases from 14.13 to 17.99
    i. You could report this as a 27.35% increase in R^2 relative to a model not taking floodplain inundation into account
2. The residual variance decreases from 3.0758 to 2.968
    i. You could report this as a 3.63% decrease in model uncertainty (as measured by residual variance) relative to a model not taking floodplain inundation into account

3.9 Evaluating Results of a GAM with Multiple Smoothers

Returning to our general modeling framework for guidance (Figure 3), we need to evaluate model results. Similar to the previous model, we need to plot our observations across predictions, a histogram of residuals, and the residuals across the predictor variable. However, we now have two predictor variables: day of water year (*water_doy*) and whether the floodplain is inundated (*fInun*). Let's take a look (Figure 15):

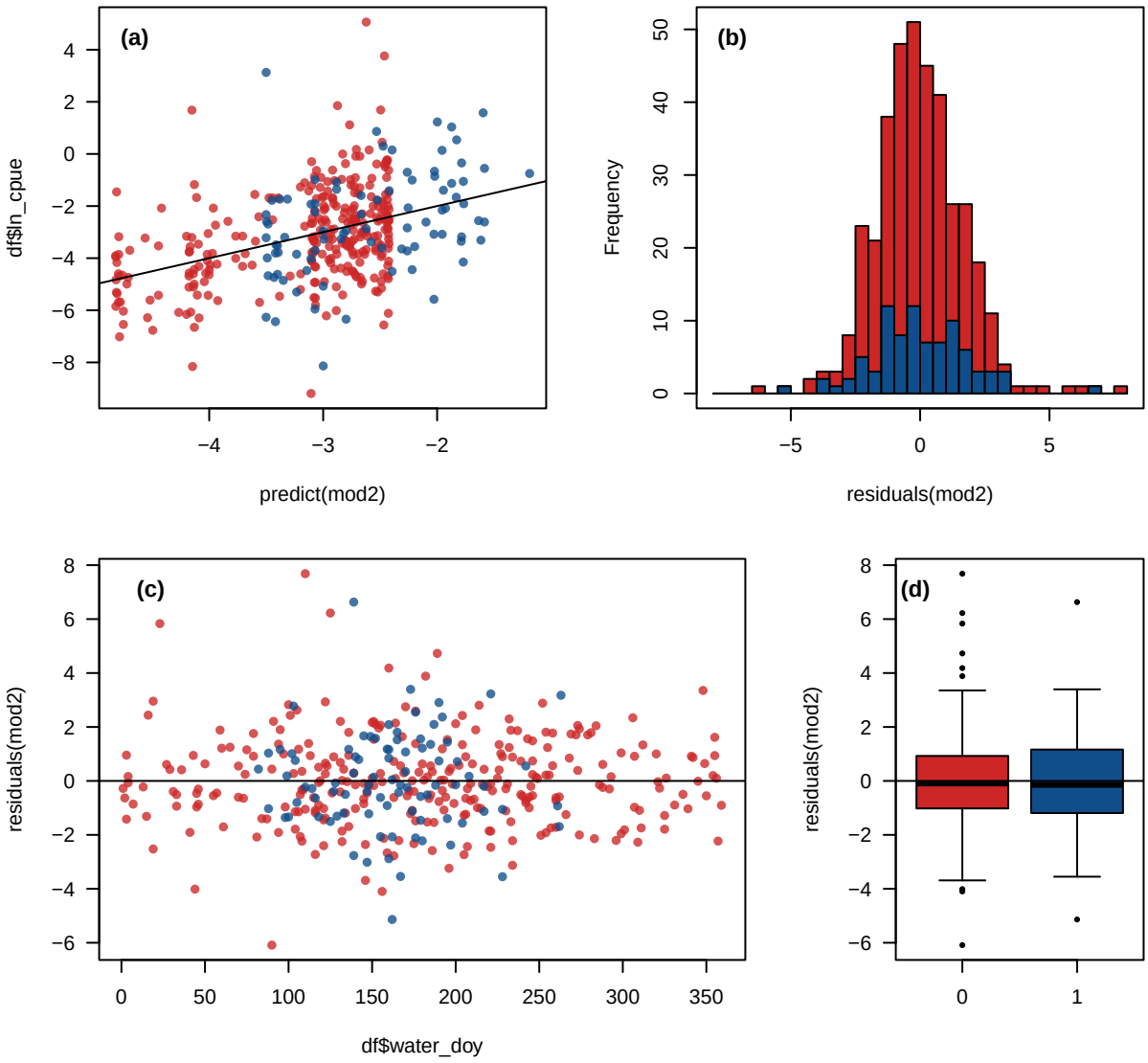

Figure 15: Model fit plot including (a) observed across predicted shown with a 1:1 line, (b) a stacked histogram of model residuals, (c) model residuals across the first predictor variable (day of water year) shown with a horizontal line at a residual value of zero, and (d) model residuals across the second predictor variable (floodplain inundation) shown with a horizontal line at a residual value of zero. Red denotes non-inundation and blue denotes inundation.

As with the previous model (Figure 12), our evaluation of fit does not raise any red flags. That said, your eye might be draw toward the spread of the residuals in Figure 15b & d. Specifically, the fact that the range of residuals is larger for the non-inundation group than the inundation group I am not worried because I believe this is simply a product of sample size. Specifically, the groups have sample sizes of 291 and 86, respectively.

Let's perform an experiment using the *rnorm()* function to test my hypothesis. The *rnorm()* function allows you to draw from a normal distribution with a pre-determined mean and standard deviation. Let's simulate the residual variance from our model using the information from the model summary (i.e., the *summary(mod2)* output above). Looking back on equation 3, we know the residual variance (that is, ε) follows a normal distribution with a mean of zero and a variance of σ^2. Therefore, let's simulate drawing

from the following a distribution using the *rnorm()* function and sample sizes of 291 and 86:

$$\varepsilon \sim N(0, 2.968) \tag{7}$$

```
# Setting the seed makes sure the random draws are the same regardless
# of how many times I print this document
set.seed(84323)

# randomly draw from the distribution defined above
group1 = rnorm(n = length(which(df$fInun=="0")),
               mean = 0, sd = summary(mod2)$ dispersion)
group2 = rnorm(n = length(which(df$fInun=="1")),
               mean = 0, sd = summary(mod2)$ dispersion)

# combine the draws into a dataframe for ease of plotting
group1_df = data.frame(values = group1,
                       group = rep("group1", times = length(group1)))
group2_df = data.frame(values = group2,
                       group = rep("group2", times = length(group2)))
group_df = rbind(group1_df, group2_df)

# plot the simulation results
par(mar=c(4,4,1,1)+0.1, oma=rep(0,4))
boxplot(group_df$value ~ group_df$group, whisklty=1,
        pch=21, las=1, ylab="Simulated Value", xlab="")
abline(h=0)
```

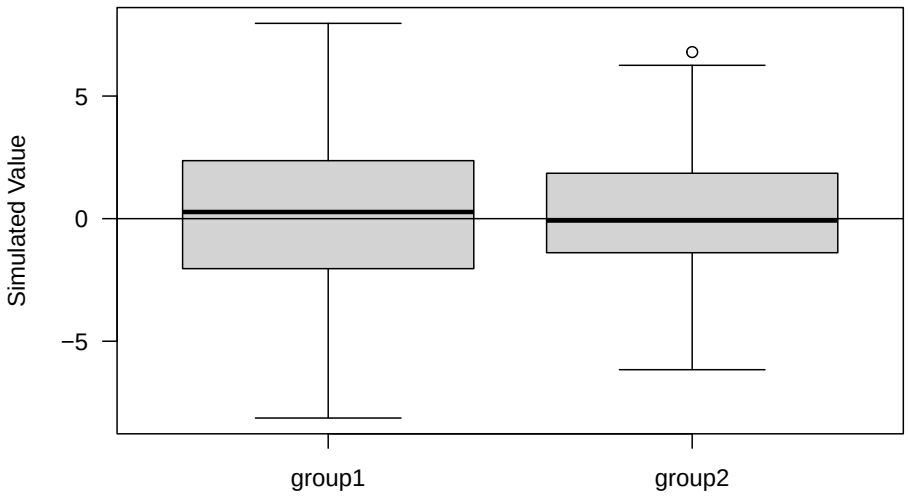

Figure 16: Simulated draws from the same normal distribution with varying sample sizes.

Hopefully, this little experiment proves that the spread of the residuals in Figure 15b & d are a product of sample size and illustrates that we should not be worried in our assessment of model fit.

3.10 Visually Presenting a GAM with Multiple Smoothers

As per usual, let's referring to Figure 3, we have validated the model and concluded the model is appropriate. Now we need to move on to model interpretation and presentation. We will follow the same procedures as discussed above to make Figure 13, but with a few differences. Because we have two predictor variables, we need to consider how to present the findings. Given that the second predictor (*fInun*) is categorical, our process is fairly easy in that we repeat the same steps as before while accounting for the data ranges of each category:

1. Determine the range of day of water year for each inundation category
2. Predict $\log_e$(CPUE) for each inundation category across the category-specific observed range of day of water year
3. Add these predictions with uncertainty to a plot of data

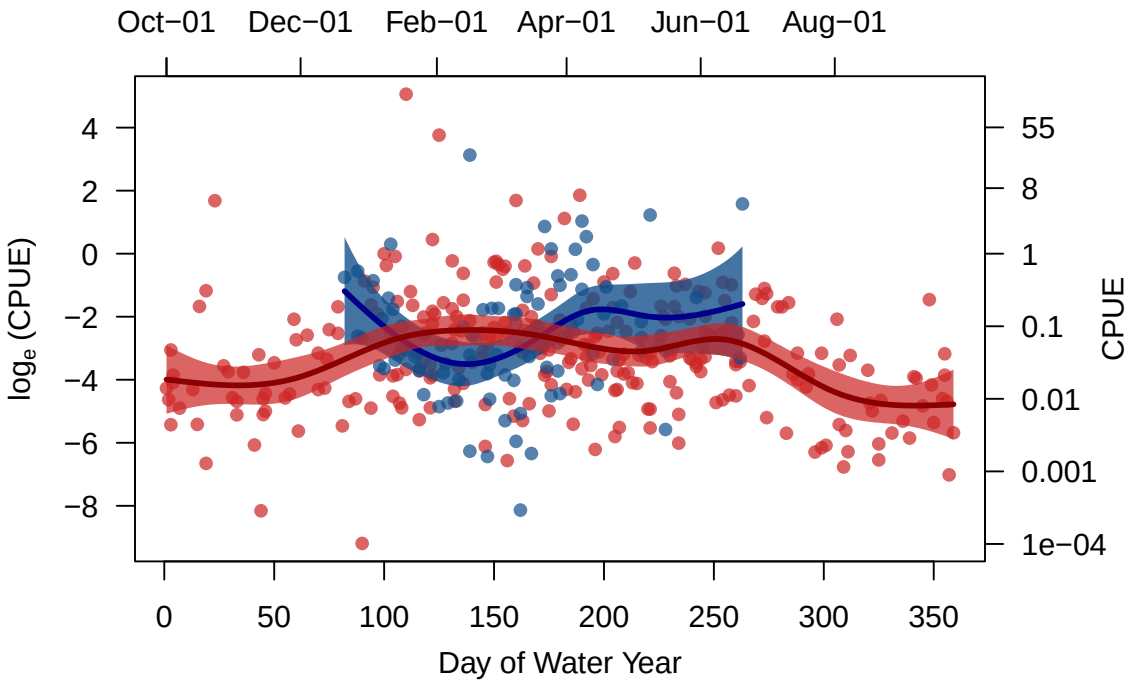

Figure 17: Model predicted (lines) $\log_e$-transformed aquatic macroinvertebrate CPUE across day of water year when the floodplain is not inundated (red) and during inundation (blue). Shown with point-wise 95%-CI (polygons) and observations (points).

Let's take a look at Figure 17 while reflecting back on the ***Take Note*** section above. Remember: GAM estimates are less stable at the data extremes due to the the underlying algorithm. So, I strongly recommend presenting GAM results that exclude the data extremes. I recommend constraining the predictions from the 5^{th}- to the 95^{th}-percentile of the data range or even from the 10^{th}- to the 90^{th}-percentile of the data range. While this is not a requirement for publication, I believe it is good practice to keep the reader/down-stream user from applying too much value at the data extremes. Let's go ahead and revise the Figure 17 by constraining the model predictions lower and upper limits to the 10^{th}- and 90^{th}-percentile of the data range, respectively (Figure 18a). Furthermore, I generally present the final model results back-transformed if I used transformed data during analysis (Figure 18b). Please note, however, that I added color-coded tick marks to Figure 18b because I removed the raw data from the plot. I recommend this addition whenever constraining the range of the model prediction in order to be fully transparent with the audience/reader.

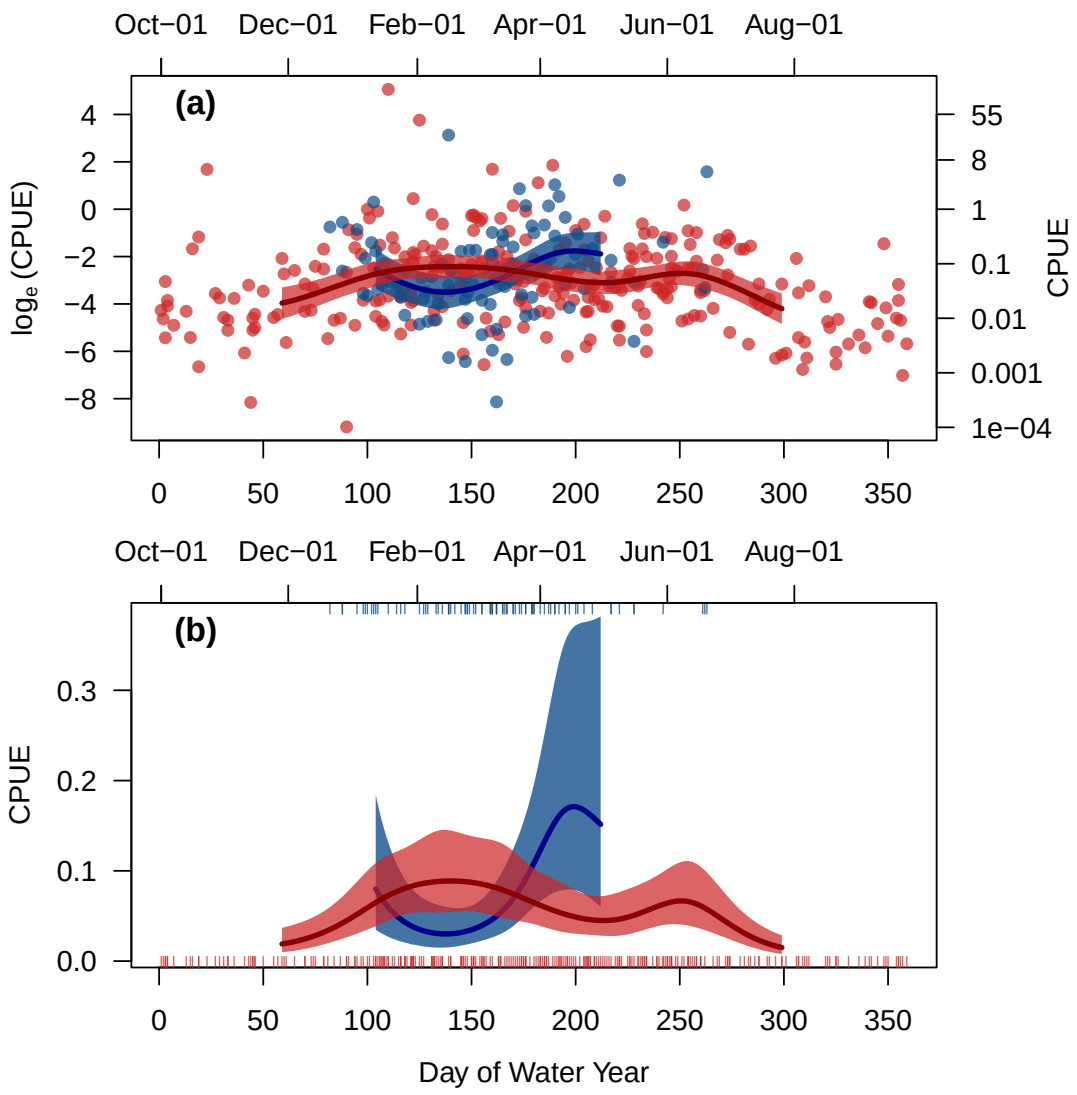

Figure 18: Model predicted (lines) (a) $\log_e$-transformed and (b) back-transformed aquatic macroinvertebrate CPUE across day of water year when the floodplain is not inundated (red) and during inundation (blue) with model predictions constrained to the 10^{th}- through the 90^{th}-percentile of the data range. Shown with point-wise 95%-CI (polygons) and observations as points in (a) and the x-axis location as color-coded rugs (i.e., vertical lines) in (b).

4 GAMs and Statistical Inference

At the end of the day (or analysis), we are often required to say whether an observed pattern is "significant." While I do not prescribe to p-values *per se* (this is a longer discussion for a different day), I do believe that whatever metric of significance you use, you must be explicit about how you are defining *significance.* To that end, I will present two methods of identifying significant differences in GAM predictions. Let's start with the most intuitive approach first: Goldstein and Healy (1995) state that if your regression meets the assumption of normality, non-overlapping error estimates derived using at a Student t distribution value of 1.39 indicates a 95% chance that the two observations not co-occurring (i.e., significantly different). In other words, you can plot 83.5% point-wise confidence intervals around the model predictions and consider

non-overlapping intervals as significantly different (Figure 19a).

The second approach is to simulate the posterior estimated from the GAM in a Bayesian framework. I have modified code from Dr. Gavin Simpson's great blog post here that I will include below. I highly recommend you read through the entire post. In short, you calculate the differences between the posterior draws when the floodplain is inundated and when it is not inundated, then determine whether the differences are greater than or less than zero given a pre-determined α value. For this analysis, we will use an α of 0.05 and consider any point estimates in which the confidence interval does not overlap with zero as a significant difference between the inundated and non-inundated categories (Figure 19b).

Interestingly, the two approaches yield the exact same results.

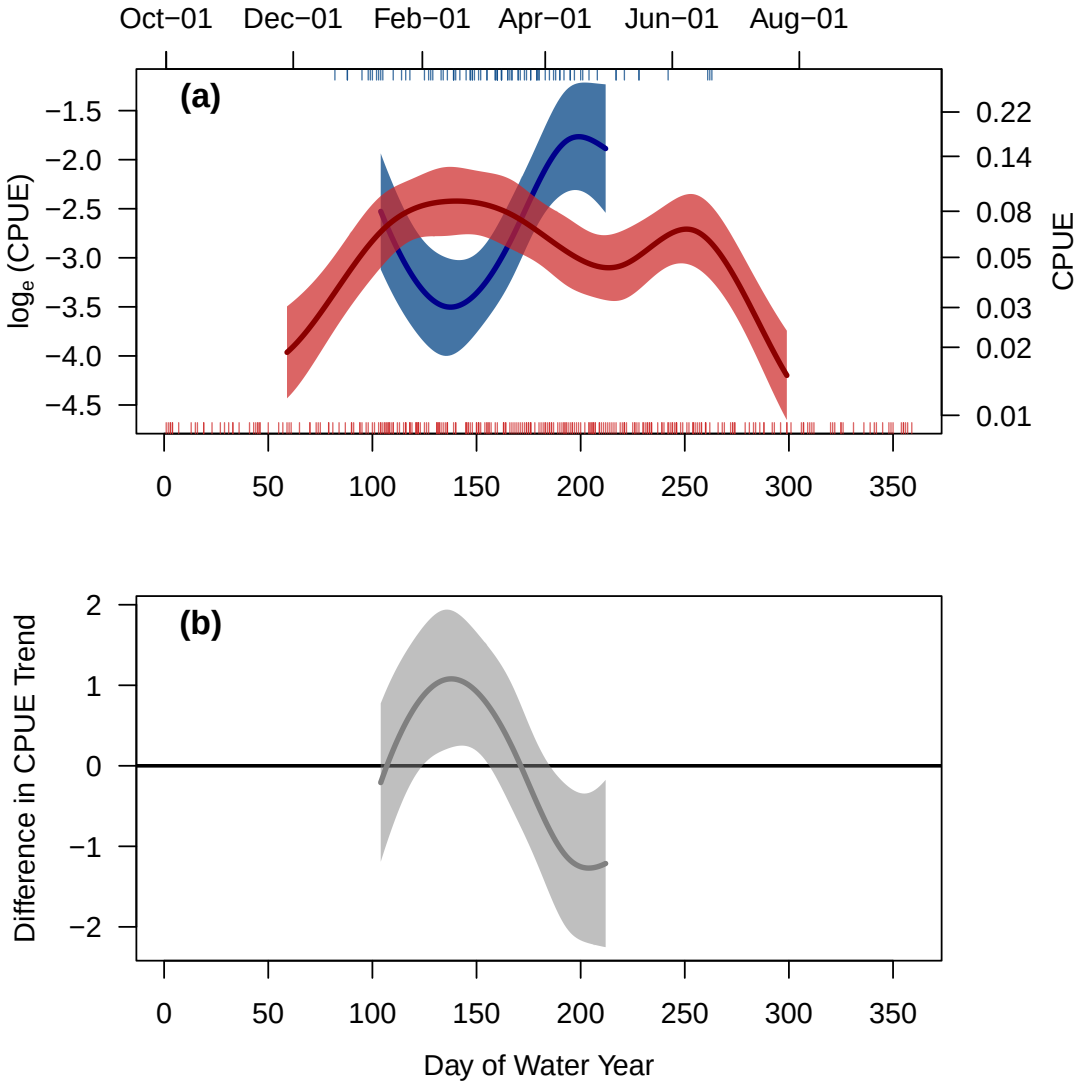

Figure 19: (a) Model predicted (lines) $\log_e$-transformed aquatic macroinvertebrate CPUE across day of water year when the floodplain is not inundated (red) and during inundation (blue) with model predictions constrained to the 10^{th}- through the 90^{th}-percentile of the data range shown with point-wise 83.5%- confidence interval (polygons) and the x-axis location as color-coded rugs (i.e., vertical lines). (b) The difference between model predictions across day of water year given floodplain inundation and no floodplain inundation with 95% confidence interval.

```
smooth_diff <- function(model, newdata, f1, f2, var, alpha = 0.05,
                        unconditional = FALSE) {
  xp <- predict(model, newdata = newdata, type = 'lpmatrix')
  c1 <- grepl(f1, colnames(xp))
  c2 <- grepl(f2, colnames(xp))
  r1 <- newdata[[var]] == f1
  r2 <- newdata[[var]] == f2
  ## difference rows of xp for data from comparison
  X <- xp[r1, ] - xp[r2, ]
  dif <- X %*% coef(model)
  se <- sqrt(rowSums((X %*% vcov(model, unconditional = unconditional)) * X))
  crit <- qt(alpha/2, df.residual(model), lower.tail = TRUE)
  upr <- dif + (crit * se)
  lwr <- dif - (crit * se)
  data.frame(pair = paste(f1, f2, sep = '-'),
             diff = dif,
             se = se,
             upper = upr,
             lower = lwr)
}
pdat <- expand.grid(water_doy = 104:212,
                    fInun=c("0", "1"))
comp1 <- smooth_diff(model = mod2, newdata = pdat, f1 = '0',
                     f2 = '1', var = 'fInun')
comp <- cbind(water_doy = 104:212,
              comp1)

plot(range(comp$upper, comp$lower) ~ range(df$water_doy),
     type = "n", las=1, ylab="", xlab="")
mtext(text = "Difference in CPUE Trend", side = 2, line = 2.75)
mtext(text = "Day of Water Year", side = 1, line = 2.5)
plot_label(lab = "(b)")
abline(h=0, lty=1, lwd=2)
polygon(x = c(comp$water_doy, max(comp$water_doy), rev(comp$water_doy),
              comp$water_doy[1]),
        y = c(comp$lower, comp$upper[dim(comp)[1]], rev(comp$upper),
              comp$lower[1]), col=gray(0.5,0.5), border=NA)
lines(comp$diff ~ comp$water_doy, lwd=3, col = gray(0.5,1))
```

5 Additional Considerations

5.1 Autocorrelation and Partial Autocorrelation

Phew, we made it. . . Or did we?

Let's go back to the assumptions of regression:

1. ***Normality:*** Our assessment of model residuals in Figure 15b is proof we met the assumption of normality
2. ***Homogeneity:*** Our assessment of model residuals across the predictors in Figure 15c & d is proof we met the assumption of homogeneity
3. ***Fixed X:*** See the initial discussion of this assumption, but we are as good as ecologists can get

4. ***Independence:*** *well...* while our data are not on a daily time step, our data are a time series and we ignored this aspect of our data throughout the entire analysis so far, **meaning we did not pass this assumption**

I selected this dataset for instructional purposes even though I *knew* we would violate the assumption of independence. Given that a lot of ecological data are time series based, I do not want to ignore this aspect of GAM analyses. The time series nature of our data means we need to test for autocorrelation and, if detected, we need to correct for it. Autocorrelation in the context of time series data is the idea that data from a specific time step is related to some previous time step(s). An easy example of this is water temperature. On a daily time step, you can very comfortably assume that water temperature on day x is going to be the same as water temperature on day *x-1* $\pm$ a degree or three. Similarly, we can assume day x is going to be the same as water temperature on day *x-2* $\pm$ a little more. This continues for some amount of days (e.g., *x-10* or *x-20*?), but at some point (maybe *x-30*, *x-60*, *x-90* days) this relationship breaks down. The relationship breaks down even more quickly when your time series on on a weekly or fortnightly basis.

Testing for temporal autocorrelation is fairly easy using the autocorrelation function *acf()* and the partial autocorrelation function *pacf()* from the ***stats*** package. To over simplify, the autocorrelation is the relationship between an observation at time t and all previous observations between time t and time *t-k*, where k is a lag of $\geqslant 0$. The partial autocorrelation is the relationship between an observation at time t and the single observation at time *t-k*, removing the effect of all observations between time t and time *t-k*. Let's take a look at the autocorrelation of our response variable:

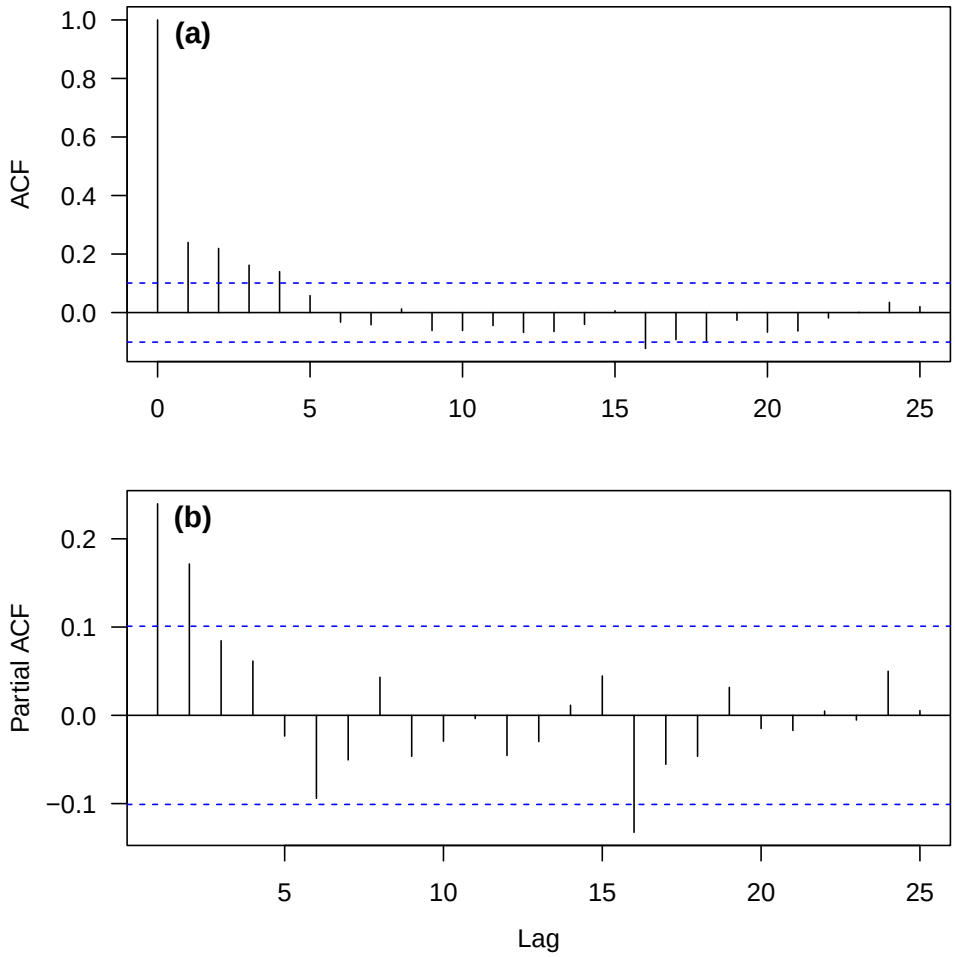

Figure 20: (a) Autocorrelation and (b) partial autocorrelation of the response variable: $\log_e$-transformed Catch per Unit Effort (CPUE).

We can see that lags (i.e., k) of 1, 2, 3, 4, and 16 have significant autocorrelation (Figure 20a) and lags 1,2, and 16 have significant partial autocorrelation (Figure 20b). These plots should raise some concerns. The next step is to determine whether the autocorrelation and partial autocorrelation is evident in residuals of the final model:

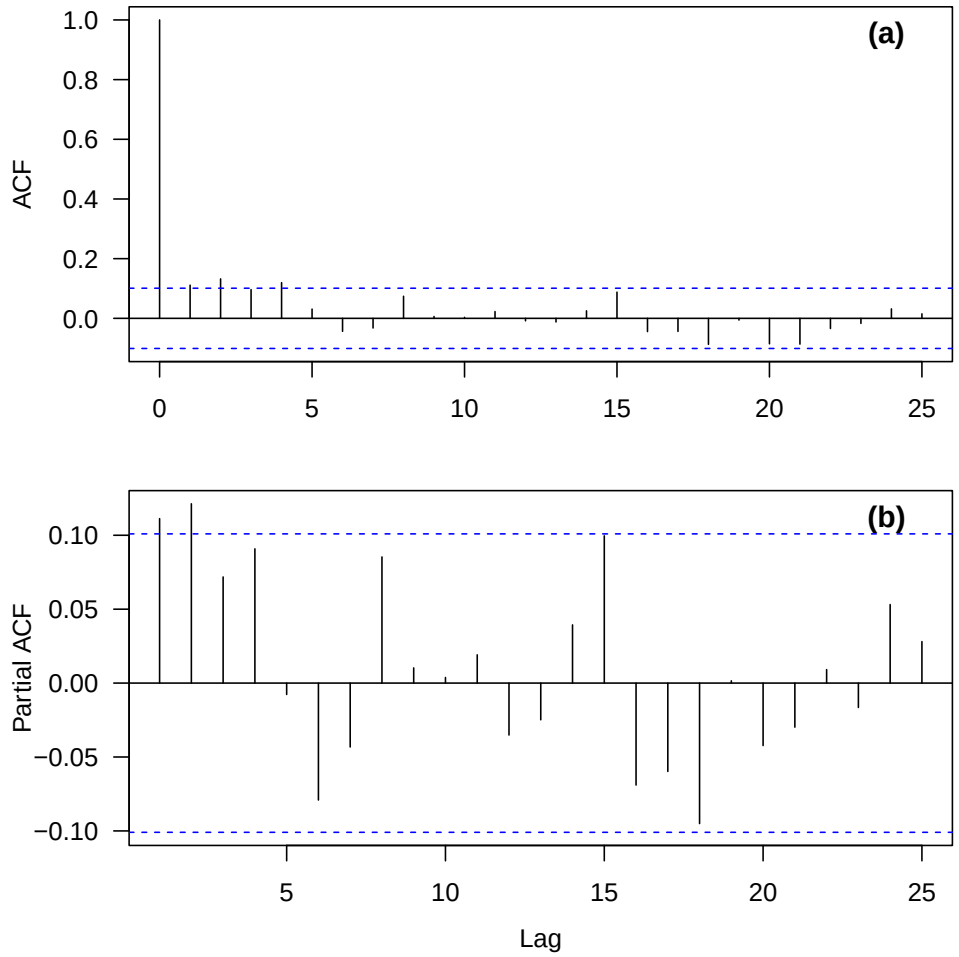

Figure 21: (a) Autocorrelation and (b) partial autocorrelation of the model residuals.

We can see the autocorrelation and partial autocorrelation is far weaker, but, technically speaking, are still significant. Specifically, the autocorrelation and partial autocorrelation are both now only significant for lags of 1 and 2.

5.2 Accounting for Autocorrelation and Cyclic Data

I highly recommend readers dig deeper into time series data and autocorrelation on there own as autocorrelation could very easily span several micro-training sessions. That is, this is *way* beyond what should be covered for GAMs. Nevertheless, I will share a little code and information to get you started. First, I recommend reading Chapter 6 in Zuur et al. (2009) titled, "*Violation of Independence - Part I*" and, if you are interested in spatial autocorrelation, Chapter 7 in Zuur et al. (2009) titled, "*Violation of Independence - Part II*." Chapter 16 in Zuur et al. (2007) titled, "*Time series analysis - Introduction*" will also be very useful to begin understanding the topic.

A very common, albeit conceptually challenging, method to account for autocorrelation in a GAM model is to "*tell*" the model that the model residuals are correlated and, essentially, that the model should account for the autocorrelation while fitting. In short, here is what we have to do:

1. We need to switch from the *gam()* function to the *gamm()* function (Generalized Additive Mixed

Models), which has the option to include the *correlation=* argument
2. Tell the model we will be using an autoregressive moving average model by setting *correlation=corARMA()*
3. Tell the model that the correlation is per water year by setting *corARMA(form = ~ 1|water_year)*
4. The model will use an optimization process to fit the two parameters for an autoregressive moving average model: ϕ (p) and θ (q); we need to give the model starting values: *corARMA(form = ~ 1|water_year, p=1, q=1)*

An additional consideration is that our data are cyclical in that day of water year 1 and day of water year 366 *should* be identical (i.e., it does not make sense for the model to fit a CPUE for the beginning of the year that is substantially different than the value at the end of the year). We can tell the model to match the CPUE the beginning and end of the water year by fitting the model using a special cubic regression spline known as a cyclic cubic spline. We can achieve this by setting the argument *by="cc"* within the *s()* function. For more information of the types of smoothers available, please run *?mgcv::smooth.terms* in your console. Let's give it a run:

```
ar_mod = gamm(ln_cpue ~ fInun +  s(water_doy, by=fInun, bs="cc"), data = df,
              correlation = corARMA(form = ~ 1|water_year, p=1, q=1))

# ~ Look at the fitted parameters: phi and theta
summary(ar_mod$lme)$ modelStruct$ corStruct
```

```
## Correlation structure of class corARMA representing
##       Phi1     Theta1
##  0.8275504 -0.6904603
```

```
par(mfrow=c(2,1), mar=c(3,4.5,1,1)+0.1, oma=c(1,0,0,0))
acf(residuals(ar_mod$lme, type = "normalized"), main="", las=1)
plot_label(lab = "(a)", x_prop = 0.92)
pacf(residuals(ar_mod$lme, type = "normalized"), main="", las=1)
plot_label(lab = "(b)", x_prop = 0.92)
mtext(text = "Lag", side = 1, line = 2.5)
```

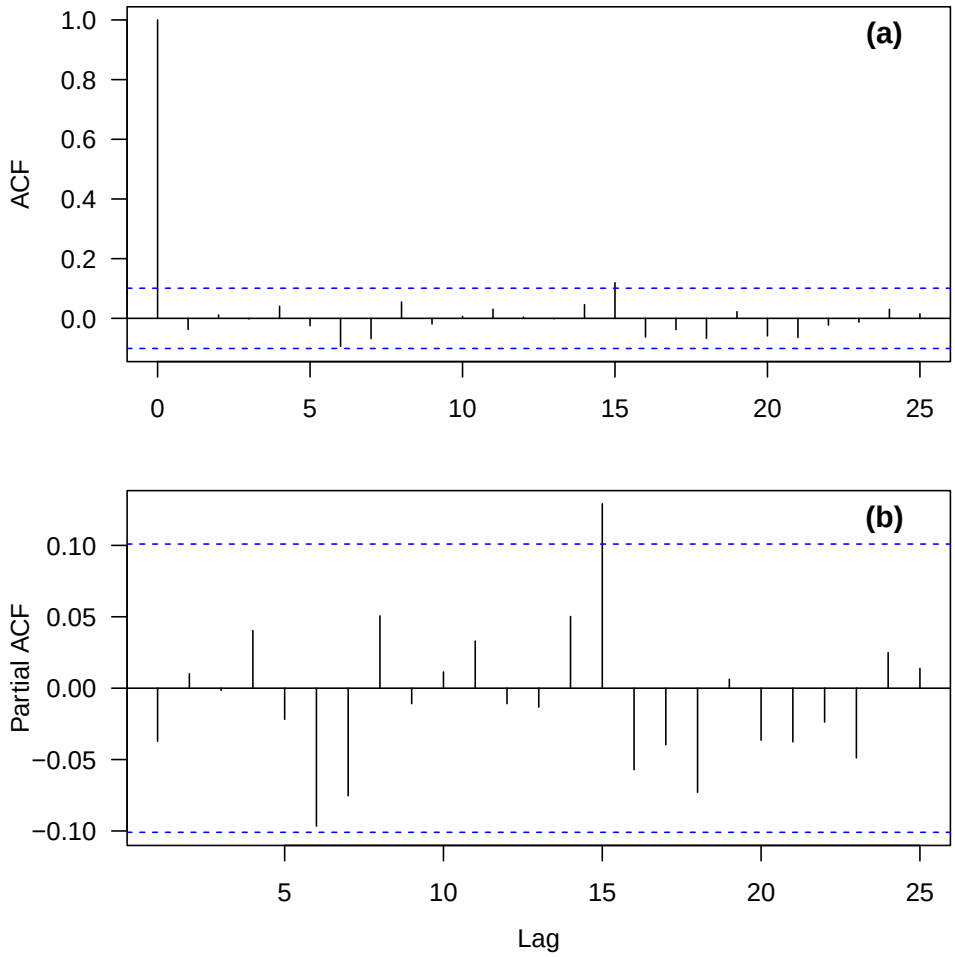

Figure 22: (a) Autocorrelation and (b) partial autocorrelation of the autoregressive moving average model residuals.

We see an ever so slightly significant autocorrelation and partial autocorrelation at a lag of 15; however, I am not very concerned as the significant lags in the previous model are now gone (Figure 22). Looking at the autoregressive moving average model fit and the final model, we see the model fits well (Figure 23) but, while the pattern remains the same, the difference is now only significant at the upper limit of day of water year (Figure 24).

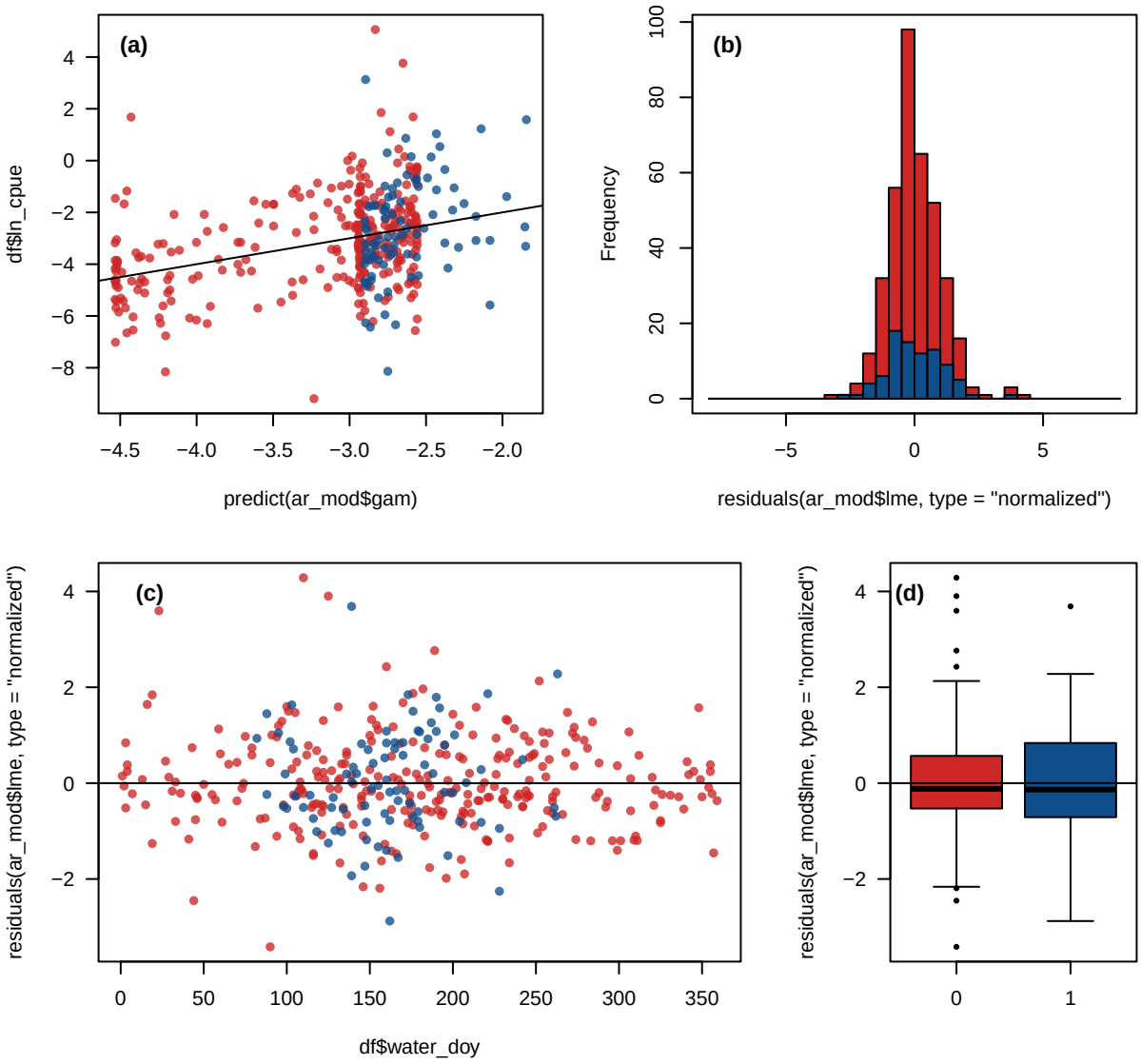

Figure 23: Autoregressive moving average model fit plot including (a) observed across predicted shown with a 1:1 line, (b) a stacked histogram of model residuals, (c) model residuals across the first predictor variable (day of water year) shown with a horizontal line at a residual value of zero, and (d) model residuals across the second predictor variable (floodplain inundation) shown with a horizontal line at a residual value of zero. Red denotes non-inundation and blue denotes inundation.

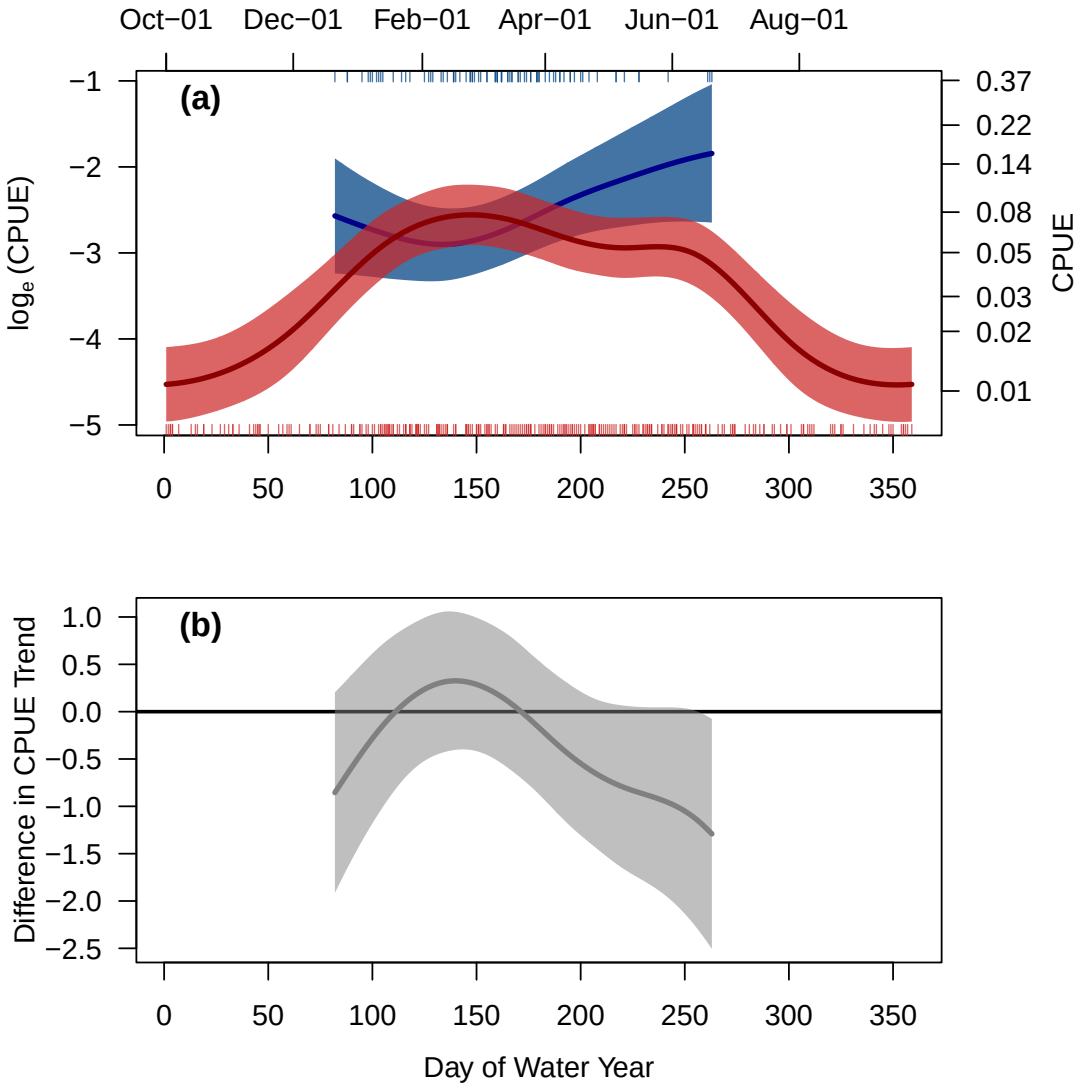

Figure 24: (a) Autoregressive moving average model predicted (lines) $\log_e$-transformed aquatic macroinvertebrate CPUE across day of water year when the floodplain is not inundated (red) and during inundation (blue); shown with point-wise 83.5%- confidence interval (polygons) and the x-axis location as color-coded rugs (i.e., vertical lines). (b) The difference between model predictions across day of water year given floodplain inundation and no floodplain inundation with 95% confidence interval.

Literature Cited

Gelman, A., and J. Hill. 2006. Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press, Cambridge.

Goldstein, H., and M. J. R. Healy. 1995. The Graphical Presentation of a Collection of Means. Journal of the Royal Statistical Society. Series A (Statistics in Society) 158:175–177.

IEP, C. Pien, B. M. Schreier, and J. Adams. 2021, November. Interagency Ecological Program: Drift invertebrate catch and water quality from the Sacramento River channel, and Sacramento River floodplain and tidal slough, collected by the Yolo Bypass Fish Monitoring Program, 1998-2019. Environmental Data Initiative.

Zuur, A. F. 2012. A Beginner's guide to generalized additive models with R. Highland Statistics, Newburgh.

Zuur, A. F., E. N. Ieno, and G. M. Smith. 2007. Analysing Ecological Data. Springer, New York.

Zuur, A., E. N. Ieno, N. Walker, A. A. Saveliev, and G. M. Smith. 2009. Mixed Effects Models and Extensions in Ecology with R. Springer Science & Business Media.