

Nonlinear models

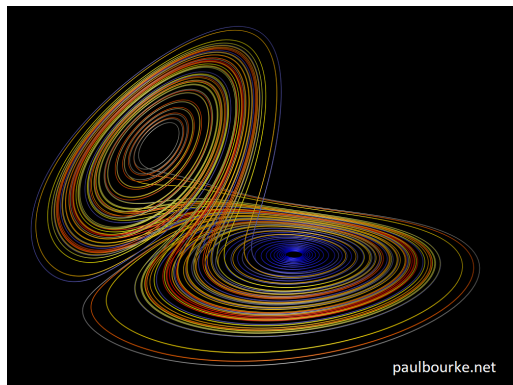
I don't think we're in Kansas anymore

Samuel Robinson, Ph.D.

Oct 13, 2023

Outline

- What are nonlinear models?
- Mechanistic models
 - Some common models
 - Strategies for fitting
- Empirical models
 - Some common models
 - GAMs



The Lorenz System: a classical 3D nonlinear system

What are nonlinear models?

- **Linear models** take the form:

$$\hat{y} = X\beta = b_0 1 + b_1 x_1 \dots + b_i x_i$$
$$y \sim \text{Normal}(\hat{y}, \sigma)$$

- **Nonlinear models** are any kind of model that can't be reduced to this linear (matrix) form:

$$\hat{y}_t = y_{t-1}^{\wedge} r \left(1 - \frac{y_{t-1}^{\wedge}}{k} \right)$$
$$y \sim \text{Normal}(\hat{y}, \sigma)$$

Two common situations

- ① “I have governing equations for this system, and I want to fit them to my data”
 - e.g. Logistic growth equation, Michaelis-Menten kinematics, Ricker model
- ② “I don't know what equations represent my system, but I need some kind of *smooth* process that describes them”
 - e.g. Changes in organism population over growing season, changes in stock prices over time

Part 1: Mechanistic models

Governing equations

Dynamics of some systems can be described by a set of equations, either in *discrete* or *continuous* time

- Exponential growth: *Discrete time*

$$n_t = n_{t-1}r$$

- Predator prey cycles: *Discrete time*

$$\text{prey}_t = \text{prey}_{t-1}(r_1 - a_1 \text{pred}_{t-1})$$

$$\text{pred}_t = \text{pred}_{t-1}(a_2 \text{prey}_{t-1} - d)$$

- Exponential growth: *Continuous time*

$$\frac{dn}{dt} = nr$$

- Predator prey cycles: *Continuous time*

$$\frac{d\text{prey}}{dt} = r - a_1 \text{pred}$$

$$\frac{d\text{pred}}{dt} = a_2 \text{prey} - d$$

Some other common dynamic models

- Logistic growth

$$n_t = n_{t-1} \left(1 + r \left(1 - \frac{n_{t-1}}{k} \right) \right)$$

- Beverton-Holt model

$$N_t = \frac{R_0 N_{t-1}}{1 + N_{t-1}/M}$$

- Michaelis-Menten

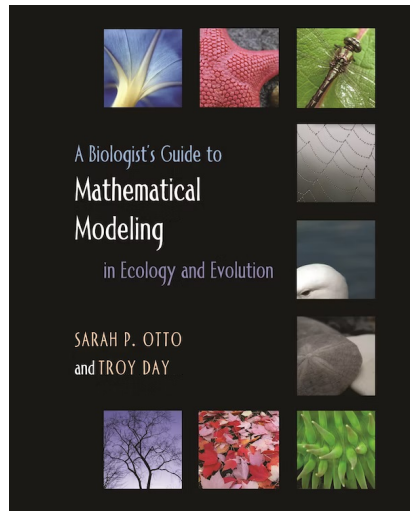
$$\frac{dp}{dt} = \frac{V_{max} a}{K_m + a}$$

- Susceptible-Infected-Recovered (SIR) model

$$\begin{aligned}\frac{dS}{dt} &= -\frac{\beta IS}{N} \\ \frac{dI}{dt} &= \frac{\beta IS}{N} - \gamma I \\ \frac{dR}{dt} &= \gamma I\end{aligned}$$

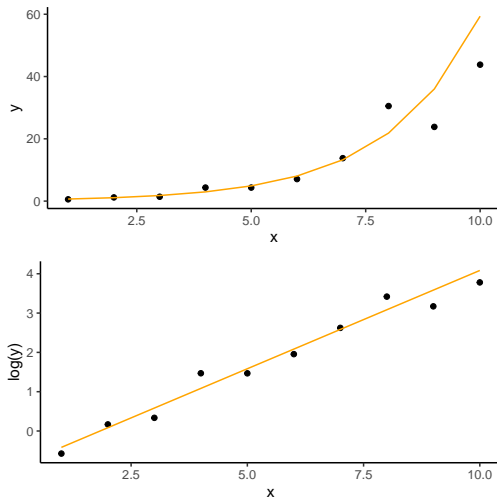
Where do these equations come from?

- Mostly from literature, sometimes from your own derivations
- Can be derived from causal models, flow diagrams, organismal life cycles
- Math-heavy topic for another class!
If you're interested, I might start with this book:



Fitting nonlinear models: transformations

- Sometimes you can transform your data to approximate nonlinear models
- e.g. $y = b_0 e^{xb_1}$ (Exponential growth)
 - Transformation:
 $\ln(y) = \ln(b_0 e^{xb_1}) =$
 $\ln(b_0) + \ln(e^{xb_1}) = \ln(b_0) + xb_1$
 - Linear model in R: `lm(log(y)~x)`
- This can cause problems because *distances* don't mean the same thing at all ranges of x-values; in general, it's better to use a NLM if you're able to

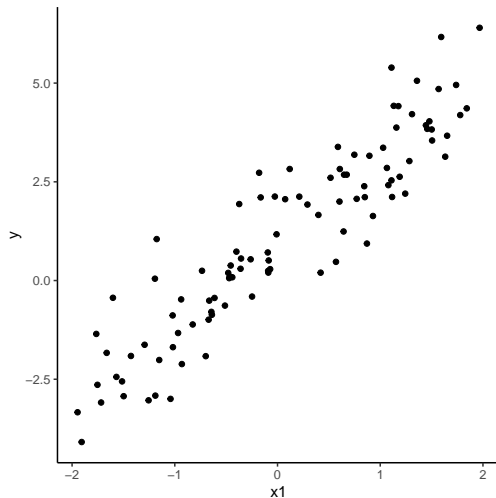


Fitting nonlinear models: simple example

- We have a *pretty good idea* what rules the system is following, and we want to figure out the parameters that it uses
- Simple example: let's start with a simple linear model, where we have 2 parameters b_0 and b_1 that we're looking for

$$\hat{y} = X\beta = b_0 + b_1x_1$$

- We're trying to find the parameters of a line that *most closely* fits our data:

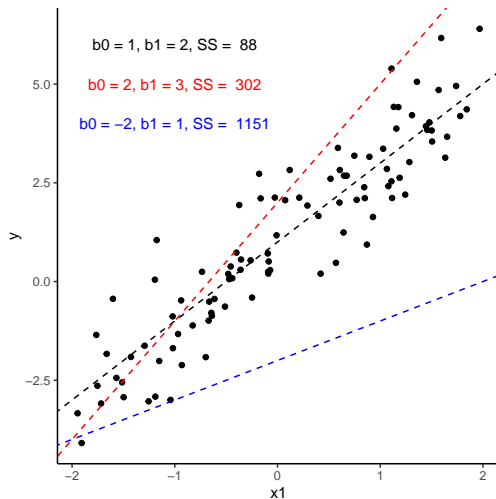


Fitting mechanistic models (cont.)

- How might we define “closest fit” in a mathematical sense?
- One common measure is *sum of squared distances*. This is just the difference between the data and the **line**:

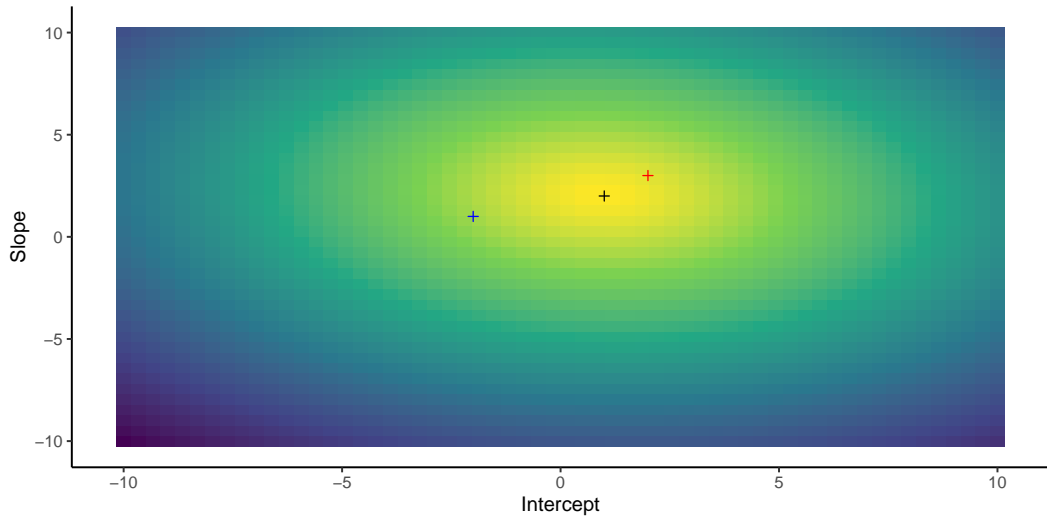
$$S = \sum_{i=1}^N (y_i - (b_0 + b_1 x_i))^2$$

- Here are three “guesses” at the slope and intercept, along with their SS scores. Which one looks to be the best?



Map of fitting surface

We can try this for a whole bunch of intercepts and slopes:



Getting R to do this

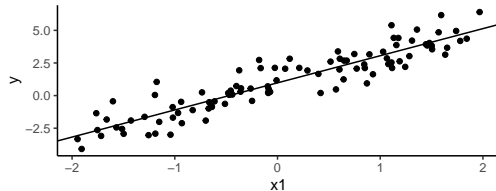
- It's pretty clear where the best intercept and slope is, but how do we get R to do this?
- First, we need a function that returns SS given a set of parameters:

```
#Function to calculate SS  
ssFun <- function(B,xdat,ydat){  
  sum((ydat - (B[1] + B[2]*xdat))^2)  
}
```

- Next, we use the `optim` function to find the intercept and slope values that return the minimum value of SS. How did it do? (Actual values: $b_0:1$, $b_1:2$)

```
#Starts at 0,0 and "looks around" from there  
optim(par = c(0,0) , fn = ssFun)
```

```
## $par  
## [1] 0.9769795 2.0779779  
##  
## $value  
## [1] 86.78819  
##  
## $counts  
## function gradient  
##      67      NA  
##  
## $convergence  
## [1] 0  
##  
## $message  
## NULL
```

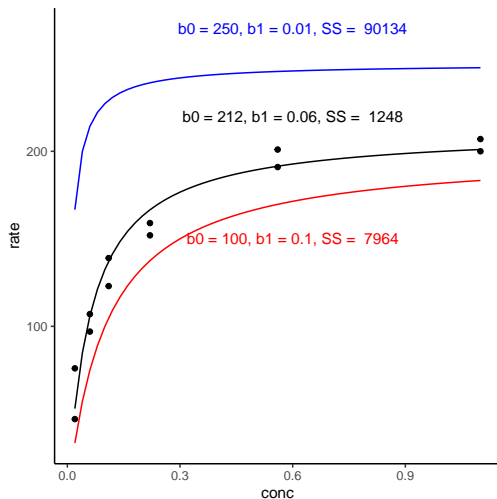


Fitting mechanistic models: nonlinear example

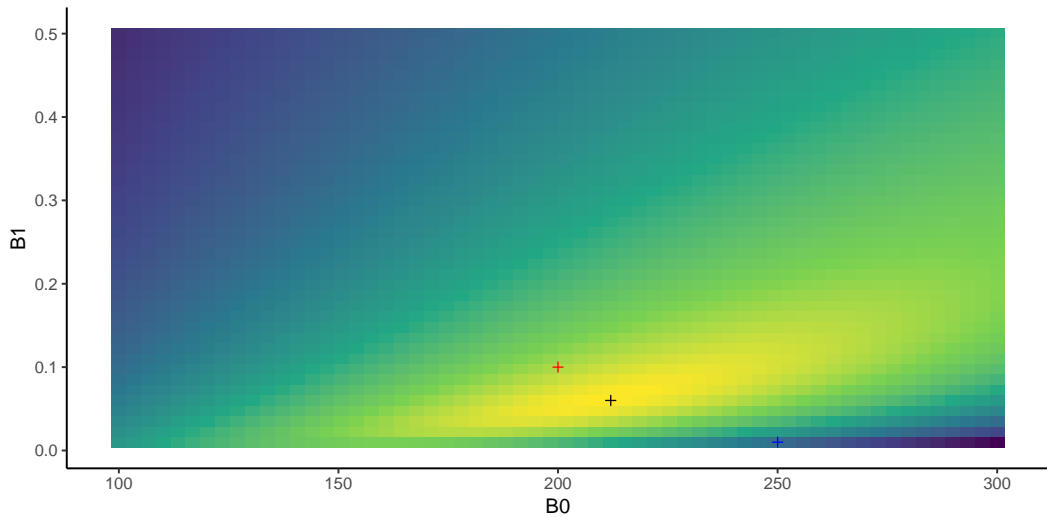
- Let's move on to a nonlinear model (Michaelis-Menten), where we also have 2 parameters b_0 and b_1 that we're looking for

$$\hat{y} = \frac{b_0 x_1}{b_1 + x_1}$$

- Again, we're trying to find the parameters of a nonlinear curve that *most closely* fits our data:



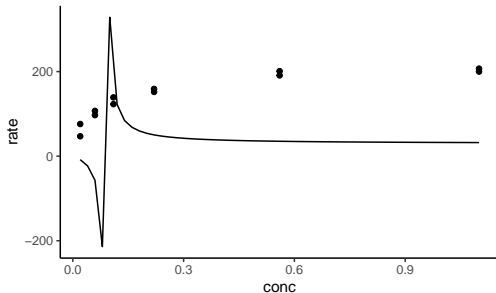
Nonlinear example (cont.)



Get R to do it

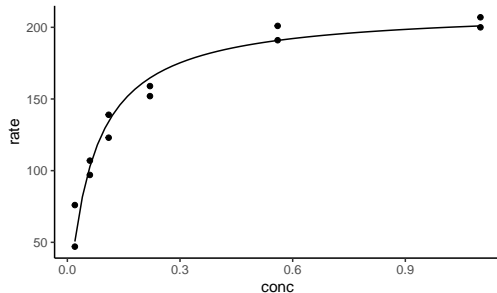
Mind your starting parameters!

```
## $par
## [1] 29.4800715 -0.0910235
##
## $value
## [1] 196776.4
##
## $counts
## function gradient
##      195      NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```



Better to start with more “realistic” values

```
## $par
## [1] 212.68974107  0.06412427
##
## $value
## [1] 1195.449
##
## $counts
## function gradient
##      81      NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```



General framework

Here are some simple rules for fitting models:

- ① Think about how your system works. What rules do you think your system follows?
- ② Write down these rules as equations, with **parameters** that control the system at time t
 - Some differential ($\frac{dx}{dt}$) equations can sometimes be solved by hand
 - Otherwise you need to use an ODE solver (fme in R)
- ③ Come up with an *objective function* that describes the differences between predictions and actual data
- ④ Get R to find parameters that *minimize* the objective function
- ⑤ See how well your model predicted your data:
 - Are all of your parameters *identifiable* from your data?
 - Do you need to go back to step 1?

First challenge: segmented regression

A classic nonlinear model is *segmented regression*. It's basically just two linear models smushed together:

$$\hat{y} = \begin{cases} b_0 + b_1x & \text{if } x \leq a \\ b_2 + b_3x & \text{if } x > a \end{cases}$$

$$b_0 + b_1a = b_2 + b_3a$$

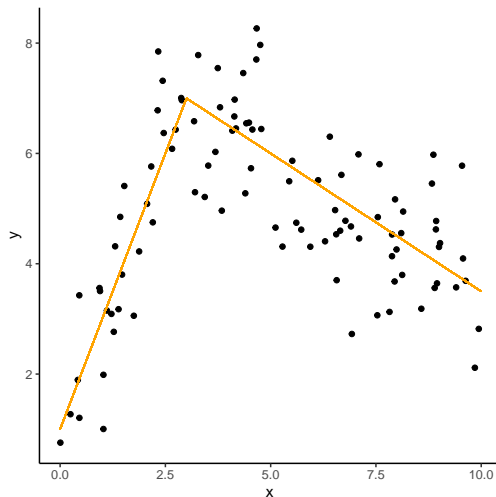
$$y \sim \text{Normal}(\hat{y}, \sigma)$$

This looks like a lot of *parameters*!

However, because of the equality

$b_0 + b_1a = b_2 + b_3a$ you get 1 for free

- 1 Rewrite the model in terms of 4 parameters instead of 5
- 2 Create a objective function to minimize across



First challenge results

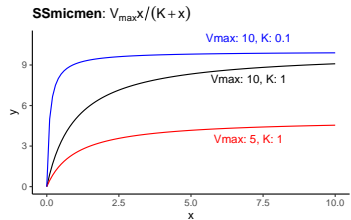
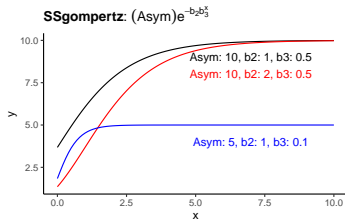
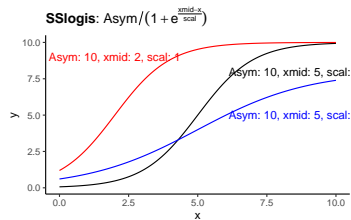
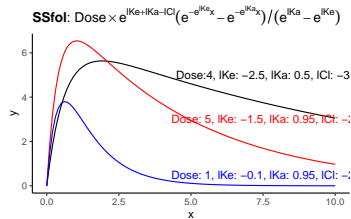
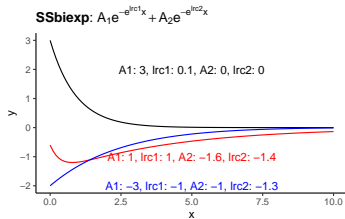
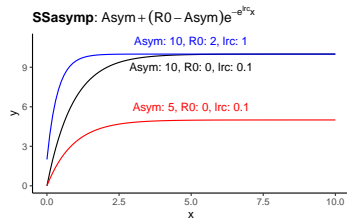
- Broken stick model

How do you get SEs on parameters?

- Easy way: bootstrapping
- Medium way: MCMC sampling (Bayesian estimation)
- Hard way: calculate Hessian of the objective function (serious math)

“There’s got to be a better way!”

Good news: someone already did the scary math for you!



Second challenge

- Try fitting Michaelis-Menten model using:
 - Your own custom function
 - `SSmicmen`

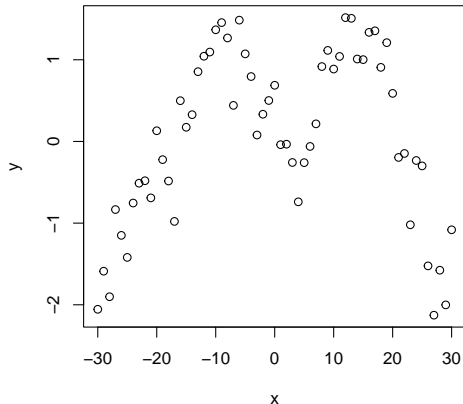
Second challenge results

- Second challenge

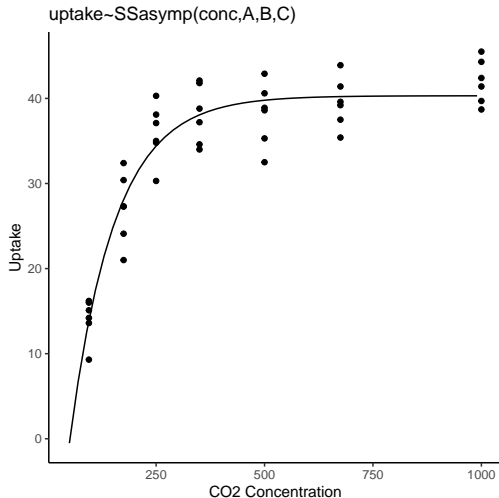
Part 2: Empirical models

Empirical smoothing

- Sometimes we don't know the specific rules that govern your system, but we want to know the *general shape*
 - e.g. population changes across time or space, temperature across seasons
- We want something that can give us *general predictions* across the range of your data without actually dealing with the underlying process
- Solution: “empirical” smoothing



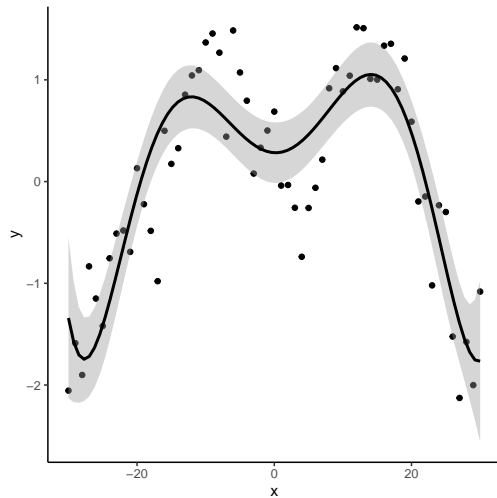
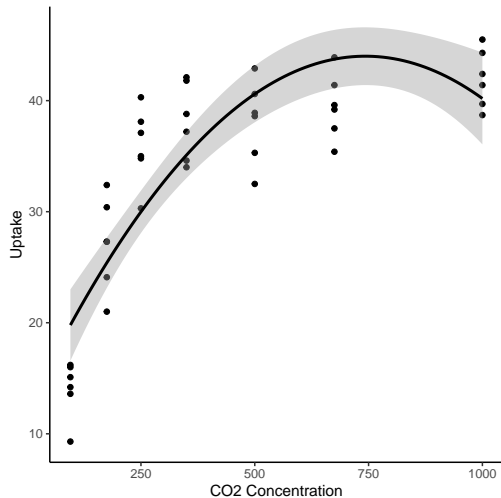
“Guess the family”



- Sometimes you can use a preset nonlinear family that looks “similar enough” to your data
- e.g. SSlogis, SSweibull
- See also: “Transformations” slide from first section

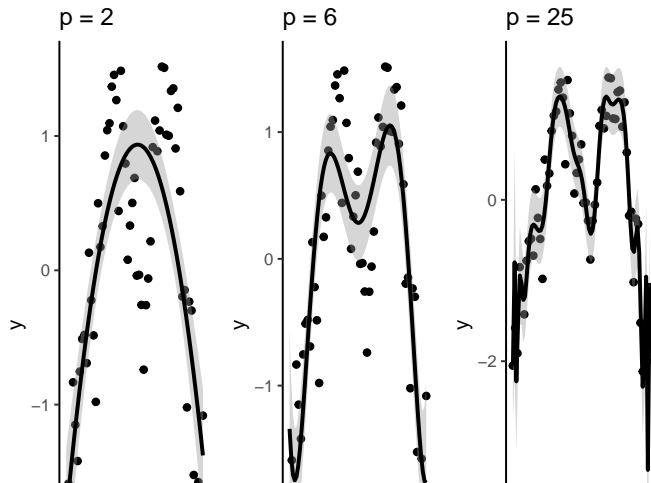
Polynomial smoothing

If the pattern is “wiggly”, you can use polynomials:



Problems with polynomials

- How many orders of polynomials do you use? Limited to discrete values
- Polynomial models don't do well outside of the range of prediction, especially at the edges of your data

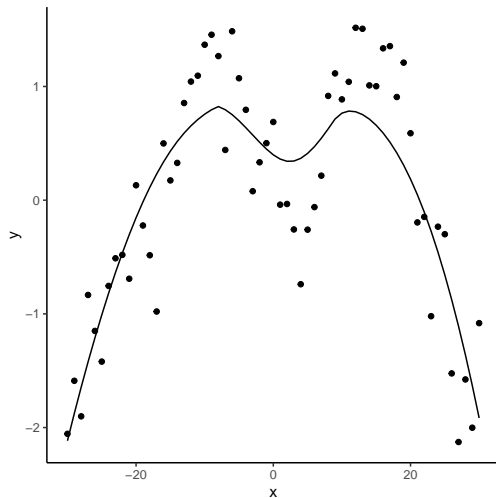


LOESS smoothers

- LOESS (LOcal regrESSion) fits simple polynomial model at *each* data point,
- Similar to a moving-average smoother (“window” of nearest N data points)

Problems:

- Computation-heavy: fits a weighted model for every data point
- Require a fair bit of data to get good predictions, sensitive to outliers
- Similar to polynomials, doesn't do well outside the range of the data



GAM: Generalized Additive Models

- Additive models are a hybrid linear model that use *basis functions* to approximate “wiggly” data
- Uses random effects to penalize curves in order to avoid overfitting (i.e. “just wiggly enough”)
- The `mgcv` package can deal with a large range of additive models, from a large range of distributions (count data, presence/absence, survival, categorical, and more)
- This package is useful for a wide variety of things, and it’s definitely worth learning

How do GAMs work?

GAMs take the form:

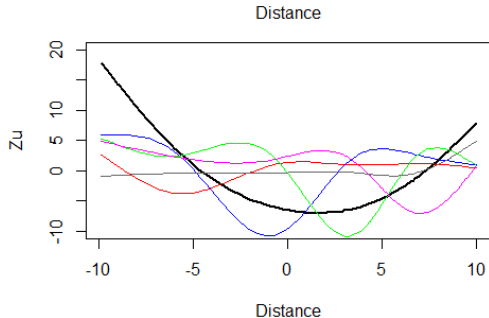
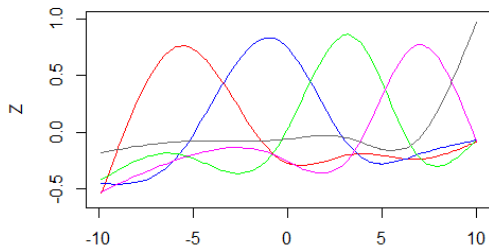
Prediction = Fixed Effect + Random Effect

$$\mu = X\beta + Zu$$

$$\text{Yield} \sim \text{Normal}(\mu, \sigma)$$

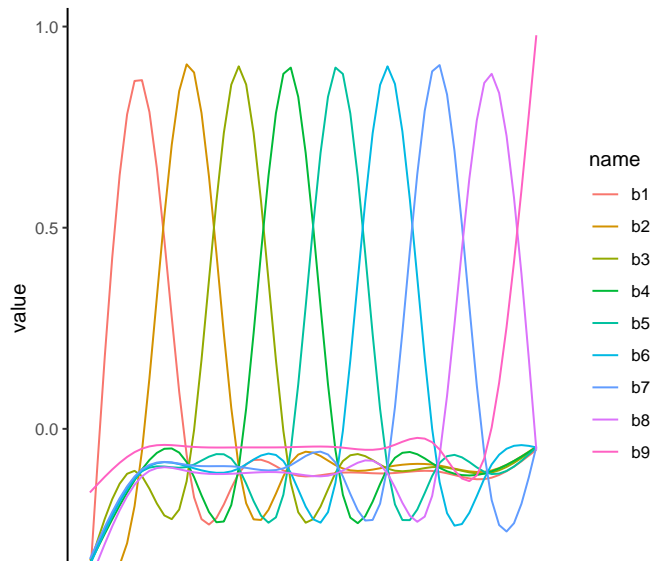
$$u \sim \text{Normal}(0, \lambda S)$$

- Creates *basis functions* across the range of data stored in columns of Z
- Finds values u
- λS penalty term: selects for optimal “wiggleness”



GAM example

Let's see how this works on a dataset:



More GAM things

- More tips and tricks about GAMs

2-column slide

a

b