BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

Comments on Modeling and Machine Learning

The Trade-Off between Bias and Variance

Random error (can't do anything with this)

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \mathrm{Var}(\hat{f}(x_0)) + [\mathrm{Bias}(\hat{f}(x_0))]^2 + \mathrm{Var}(\epsilon). \qquad (2.7)$$

Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set *(ISL 2.2)*

If I create a sophisticated machine-learning model with highly accurate predictions, this sensitive model will react more (changes in error) to changes in data than an simpler, more biased model.

Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. *(ISL 2.2)*

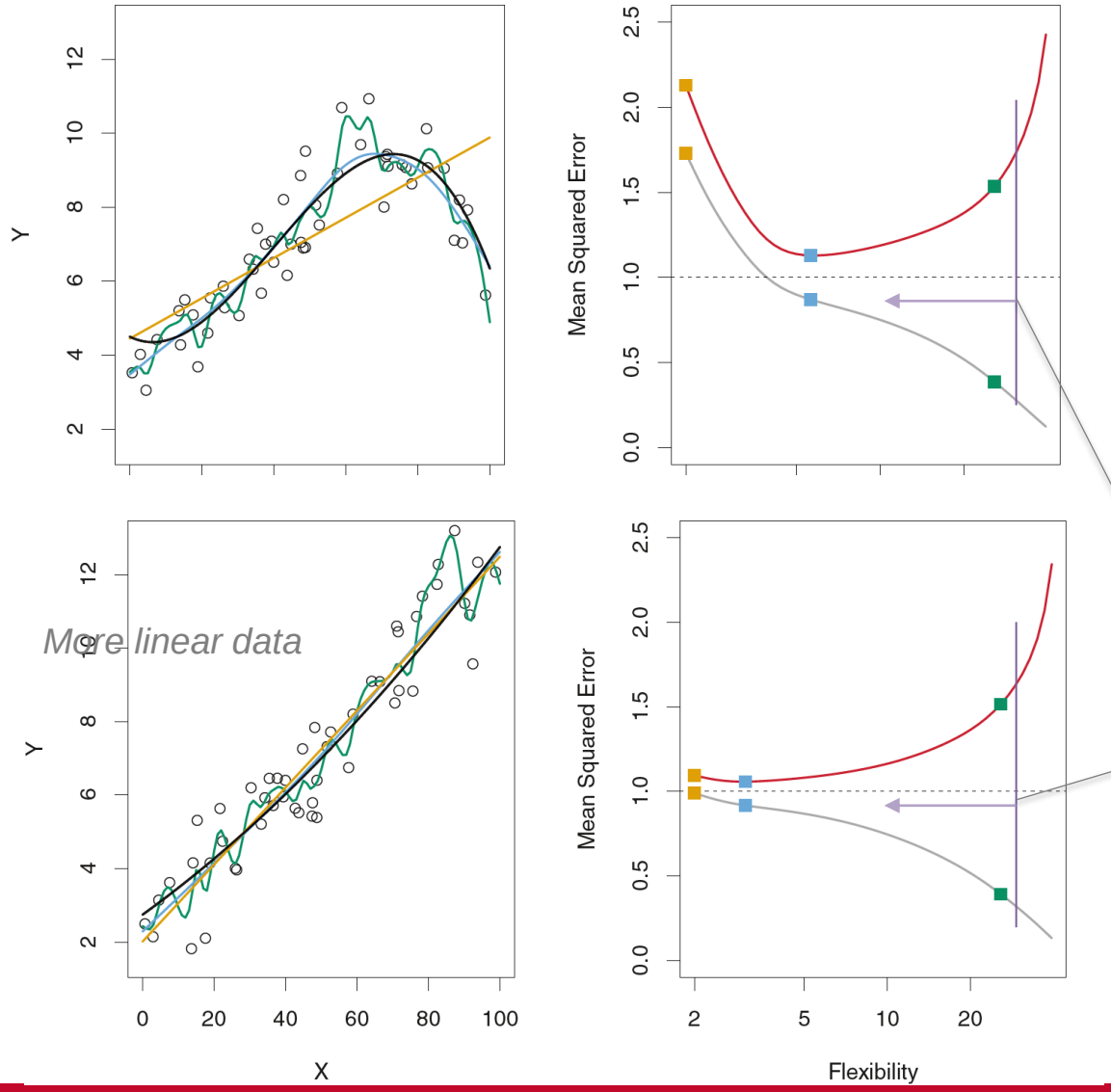
If I use a simple linear model to describe a complex problem (e.g., commodity futures pricing), I'm expecting a certain amount of error, which I accept to keep the model simple and understandable.

This is a bit like a balloon. If I squeeze the Bias error, the Var error blows up, and vice-versa.

The Trade-Off between Prediction Accuracy and Interpretability

*As ML algorithms create, transform and expand parameters – they quickly become too complex for human interpretation, which means that they do not **explain** (i.e., explanatory models) the relationships between dimensions and outcome (i.e., business drivers can't be explicitly managed).*

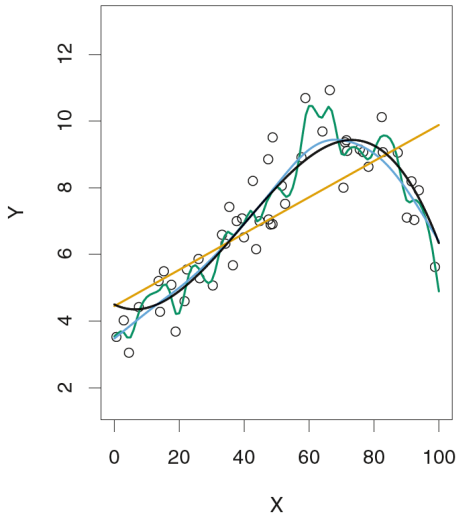
Ch 2 ISL

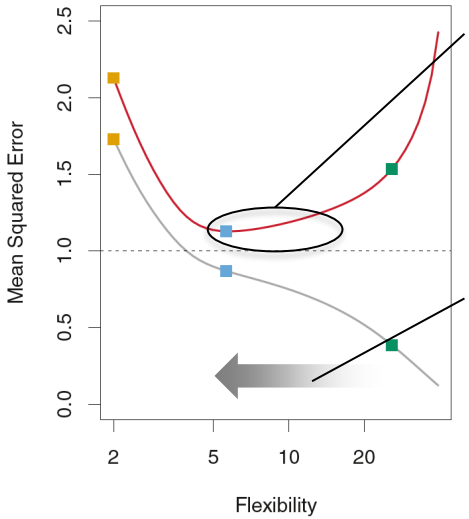


Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).

Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

So what we need is a mechanism to prevent overfitting. To "push" flexibility back.





Here's our sweet spot – good metrics *(depends on your specifications)* with real *(out-of-sample)* data is what matters. Business data are dynamic and validation is dependent on those dynamics - sometimes once a year, sometimes once a minute.

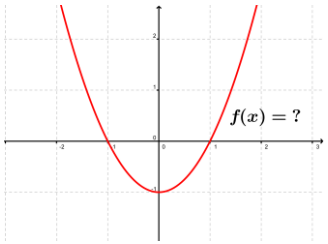
If you want your model to explain relationships causes, drivers and be able to make recommendations, then you have to consider the direction of interpretability

This is s core competency for analysts. Your judgment determines model selection, tuning and interpretation. Technology won't solve this problem for you - it's your ambiguous intuition and experience that determines success. That's why we have case methods in business school, case-based interviews and internships. Companies will test your understanding of business drivers, data relevancy and problem rationalization. Pushing buttons won't get you answers.

Note: Bayesian modeling retains interpretability with increasing flexibility, and also integrates ambiguous judgment. That's why we'll study it.

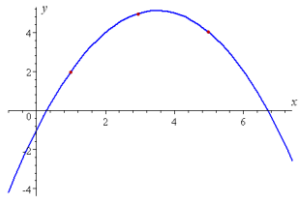
Review - Polynomial Regression

Polynomials *(i.e., quadratic, cubed, quartic, quantic…)* transform a linear model into a "non-linear" model *(caveat: linear models will often appear "curved" if there's more than 2 dimensions).*

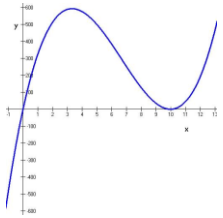


Because we're transforming the data, not the coefficients, we can still use an linear model to fit.

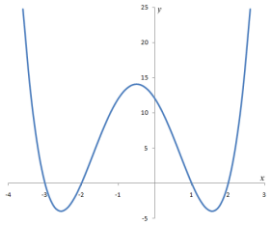
Like function fitting earlier, you need to look at the data first. You can add as many terms as you want, but to diminishing returns. Envision the curve – good place to start:



- A quadratic term creates a curve with one "hump" a U or inverted U shape. The curve does not need to contain both sides of the U. In can contain just part of it too!.



- A cubic has two humps–one facing upward and the other down. The curve goes down, back up, then back down again (or vice-versa).

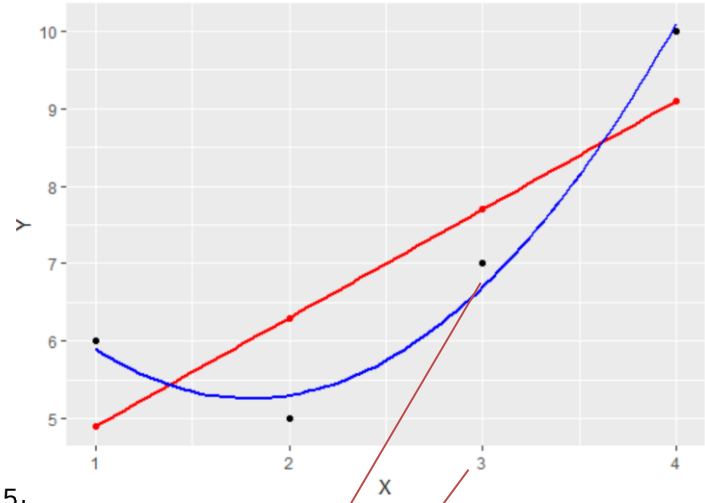


- A quartic function has multiple humps…. On and on.

ISLR Section 3.2, Multiple Regression is a good discussion (including indicator and surrogate variables) Section 3.3, Other Considerations in the Regression Model is also has good discussion (including interaction effects).
You should review if you haven't studied these topics

Polynomial Regression

```
library(ggplot2)

setwd("C:/Users/ellen/OneDrive/Documents/Spring 2017/Section III/History")
mydata <- read.csv(file="Ex1LS.csv", header=TRUE, sep=",")

model <- lm( formula = Y ~ X, mydata)
modelQ <- lm( formula = Y ~ X + I(X^2), mydata)

modData <- mydata
modData$newYQ <- predict(modelQ, mydata)
modData$newYQ <- predict(modelQ, mydata)

modData$newY <- predict(model, mydata)

p <- ggplot(modData, aes(x=X, y=Y))+geom_point()
p <- p + geom_point(data = modData, aes(x=X, y = newY), color = 'red')
p <- p + geom_smooth(data=modData, aes(X, newY), se=FALSE, color = "red", span = 1.5)
p <- p + geom_smooth(data=modData, aes(X, newYQ), se=FALSE, color = "blue", span = 1.5)
p

model$coefficients
modelQ$coefficients

x <- mydata$X
y <- mydata$Y
d <- data.frame(x=x,y=y)
#mydata <- d

p <- ggplot(d, aes(x,y)) + geom_point()
p
```




```
> model$coefficients
(Intercept)           X
        3.5         1.4
> modelQ$coefficients
(Intercept)           X      I(X^2)
        8.5        -3.6         1.0
>
> tstX <- 3
> tstY <- 8.5 -3.6*tstX + 1*(tstX^2)
> tstY
[1] 6.7
```

To set expectations, you are absolutely expected to be able to translate coefficients to an equation and create predictions. You're also expected to be able to solve for x, given a y value.

Just as a note here: when we fit a polynomial, we're transforming the data (x^2) to accommodate a non-linear relationship within a linear model to fit the parameters. So this is still a LINEAR model

BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

Regularization

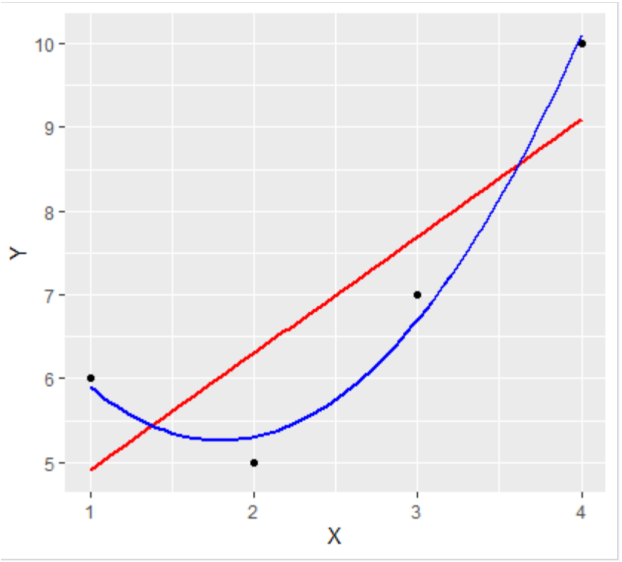
Regularization really lies at the heart of machine learning. Think of it this way:

We use algorithms to find parameters *(like we use gradient descent to find a regression coefficient).* As we'll soon see, many machine learning algorithms are free to create as many parameters as they need to *fit* the data. And recall from ISL that, as algorithms become more flexible, more complex, they also tend to be more variable, Sometimes this is needed – you might have many, many variables that affect, or even cause, the values of a dependent variable, that involve complex relationships that cannot represented by interpretable equations. So complex algorithms become our best solution *(there are many caveats to this statement!)*.

But this variability is also a problem because, as soon as real scenarios and data are applied. The algorithms will freak out *(my term for expansion of error and confidence intervals ☺)*. Simply stated, expected values will be way off. We call this "overfitting", and it's a common failure with inexperienced analysts *(who either trust technology and data science way, WAY too much, or are just lazy)*

Regularization is one approach for compensating for models that tend to overfit. In fact, the more complex the model, the more regularization impact *(it penalizes complexity and large coefficients that are often large due to complexity – so, it's a great idea in that the penalty fits the crime, and it's really quite simple conceptually)*.

```
> mydata <- read.csv(file="Ex1LS.csv", header=TRUE, sep=",")
>
>
> model <- lm( formula = Y ~ X, mydata)
> modelQ <- lm( formula = Y ~ X + I(X^2), mydata)
>
> modData <- mydata
> modData$newY <- predict(model, mydata)
> modData$newYQ <- predict(modelQ, mydata)
>
> rmse(modData$newY - modData$Y)
[1] 1.024695
> rmse(modData$newYQ - modData$Y)
[1] 0.2236068
>
> p <- ggplot(modData, aes(x=X, y=Y))+geom_point()
> p <- p + geom_smooth(data=modData, aes(X, newY), se=FALSE, color = "red", span = 1.5)
> p <- p + geom_smooth(data=modData, aes(X, newYQ), se=FALSE, color = "blue", span = 1.5)
> p
```



We studied polynomial regression in DA1, and found out that, for some datasets (esp. non-linear), it gives us a better fit – i.e., we're increasing the complexity and flexibility of the model to fit the data.

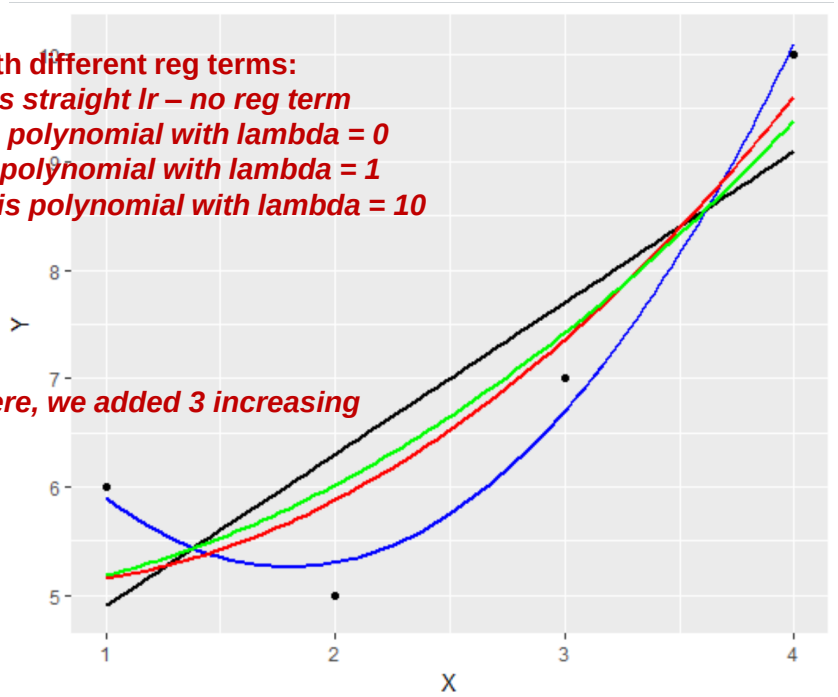
Comparing the rmse above, modelQ, the polynomial model, has lower error on TEST data.

Regularization.R

… polynomial and regularization

```
m = length(mydata$X)
x = matrix(c(rep(1,m), mydata$X, mydata$X^2), ncol=3)
n = ncol(x)
y = matrix(mydata$Y, ncol=1)
lambda = c(0,1,10)
d = diag(1,n,n)
d[1,1] = 0
th = array(0,c(n,length(lambda)))

for (i in 1:length(lambda)) {
  th[,i] = solve(t(x) %*% x + (lambda[i] * d)) %*% (t(x) %*% y)
}

nwx = seq(1, 4, len=50)
x = matrix(c(rep(1,length(nwx)), nwx, nwx^2), ncol=3)
newData <- as.data.frame(nwx)
newData$th1 <- (x %*% th[,1])
newData$th2 <- (x %*% th[,2])
newData$th3 <- (x %*% th[,3])

X <- newData$nwx
newData$lm <- predict(model, newData = X)


p <- ggplot(mydata, aes(x=X, y=Y))+geom_point() + geom_smooth(method =
'lm', se=FALSE, color = 'black')
p <- p+ geom_smooth(data = newData, aes(x=nwx, y = th1), color = 'blue')
p <- p+ geom_smooth(data = newData, aes(x=nwx, y = th2), color = 'red')
p <- p+ geom_smooth(data = newData, aes(x=nwx, y = th3), color = 'green')
p
```

Plot out with different reg terms:
Black line is straight lr – no reg term
Blue line is polynomial with lambda = 0
Red line is polynomial with lambda = 1
Green line is polynomial with lambda = 10



Add a regularization term *(here, we added 3 increasing values to see the effects)*

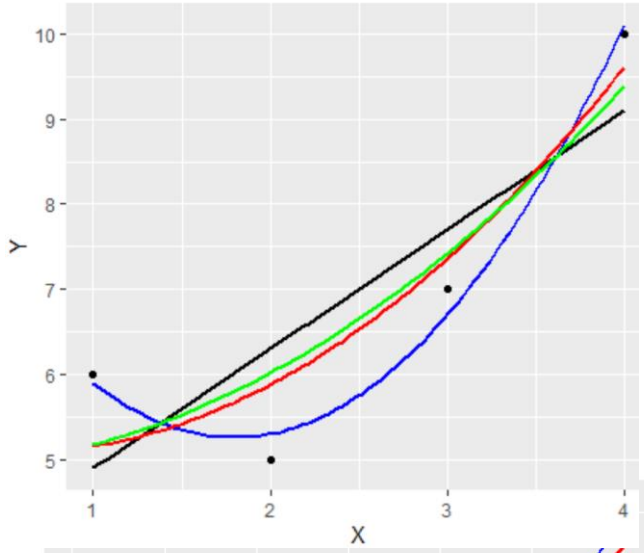
Take the normal equation from Regression 1 *(which is a least cost minimization function)*

$$\beta = (X^T X)^{-1} (X^T Y)$$

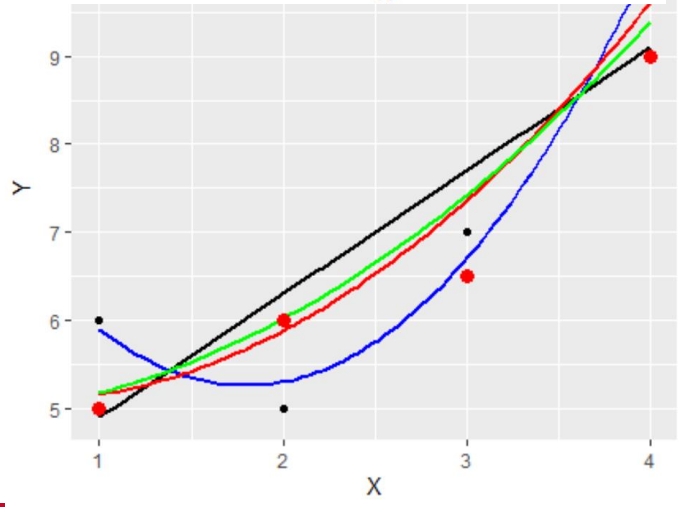
betaHat <- solve(t(X)%*%X) %*% t(X) %*%y

But what happens when new data is applied? In business, we might use last years data to build a pricing model, and this year, new products are introduced, old ones retired, new locations open up, new competitors enter the market… and all the relationships change. So, the more flexible, variable models often miss the mark. Note that the new data yields a lower rmse with the linear model here.

```
> # and compute errors
> # --- blue
> sqrt(sum(((xq %*% betaHatQ)- mydata$Y)^2)/(nrow(betaHat)))
[1] 0.3162278
> # --- red
> sqrt(sum(((xq %*% th[,2])- mydata$Y)^2)/(nrow(betaHat)))
[1] 0.940631
> # --- green
> sqrt(sum(((xq %*% th[,3])- mydata$Y)^2)/(nrow(betaHat)))
[1] 1.065107
>
> # so the blue has the lowest error
```




```
> # --- blue
> sqrt(sum(((xq2 %*% betaHatQ)- newData$Y)^2)/(nrow(betaHat)))
[1] 1.129159
> # --- red
> sqrt(sum(((xq2 %*% th[,2])- newData$Y)^2)/(nrow(betaHat)))
[1] 0.7578277
> # --- green
> sqrt(sum(((xq2 %*% th[,3])- newData$Y)^2)/(nrow(betaHat)))
[1] 0.7166328
>
> # and now the lowest error is green. WHY?
```



Again, this is a central concept in machine learning. Remember, the bias variance tradeoff. *(Chpt 6 in the book)*.

Regularization adds a term to a cost function to penalize complexity (which prevents overfitting – remember the bias / variance tradeoff)

The reg term works in opposition to the minimization function

$$\min_{b_0,b_1,b_2,\ldots}\left[\sum_{i=1}^{n}(y_i - (b_0 + b_1x_{i,1} + b_2x_{i,2} + \ldots))^2 + C(b_0^2 + b_1^2 + b_2^2 + \ldots)\right]$$

We'll use Lambda for the Constant

- It favors smaller coefficients when there's uncertainty.
- ***Overfitting*** is controlled by parameter *C.*
- *C* is set in practice using nested cross-validation (later in the course).

L1 and L2 Norms

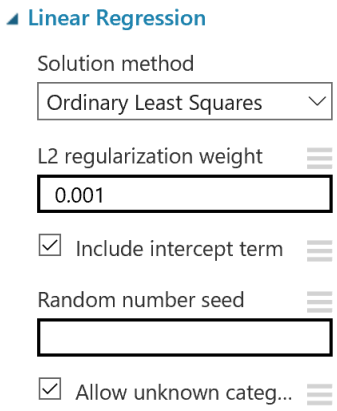
Note: the book refers to L1 as Lasso and L2 as Ridge Regression

L1 Loss Function

$$S = \sum_{i=1}^{n} |y_i - f(x_i)|$$

L2 Loss Function

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$



L2 loss function	L1 loss function
Not very robust	Robust
Stable solution	Unstable solution
Always one solution	Possibly multiple solutions

*The compares with a **scaled** lambda term in your code sample*

L1 Regularization

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

L2 Regularization *(what we just did)*

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

L2 regularization	L1 regularization
Computational efficient due to having analytical solutions	Computational inefficient on non-sparse cases
Non-sparse outputs	Sparse outputs
No feature selection	Built-in feature selection

Read about comparison between L1 and L2 re: dimension reduction in book – Chptr 6!!

BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

L1 and L2 norms

OLS assumes that the "best" predictions for y minimize the squared residuals. Known as the **L2 norm**:

$$\frac{1}{m} \sum_{i=1}^{m} (y_i - (\beta_0 + \beta_1 X_i) = 0$$

An **L1 norm** considers the absolute error *(instead of the squared error which can over-emphasize outliers)*.

$$\frac{1}{m} \sum_{i=1}^{m} (y_i - |\beta_0 + \beta_1 X_i| = 0$$

Comparing again:

Least Squares Regression	Least Absolute Deviations Regression
Not very robust	Robust
Stable solution	Unstable solution
Always one solution	Possibly multiple solutions
No feature selection	Built-in feature selection
Non-sparse outputs	Sparse outputs
Computational efficient due to having analytical solutions	Computational inefficient on non-sparse cases

True if you consider 'Inf' and 'NA' solutions

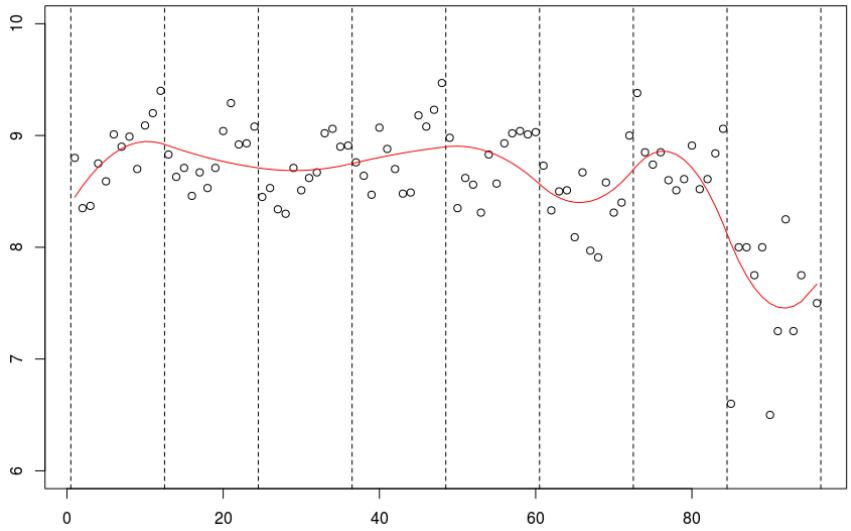
BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

splines

A ‘spline’ is a function that is constructed piece-wise from polynomial functions. The term comes from the tool used by shipbuilders and drafters to construct smooth shapes having desired properties. Drafters have long made use of a bendable strip fixed in position at a number of points that relaxes to form a smooth curve passing through those points

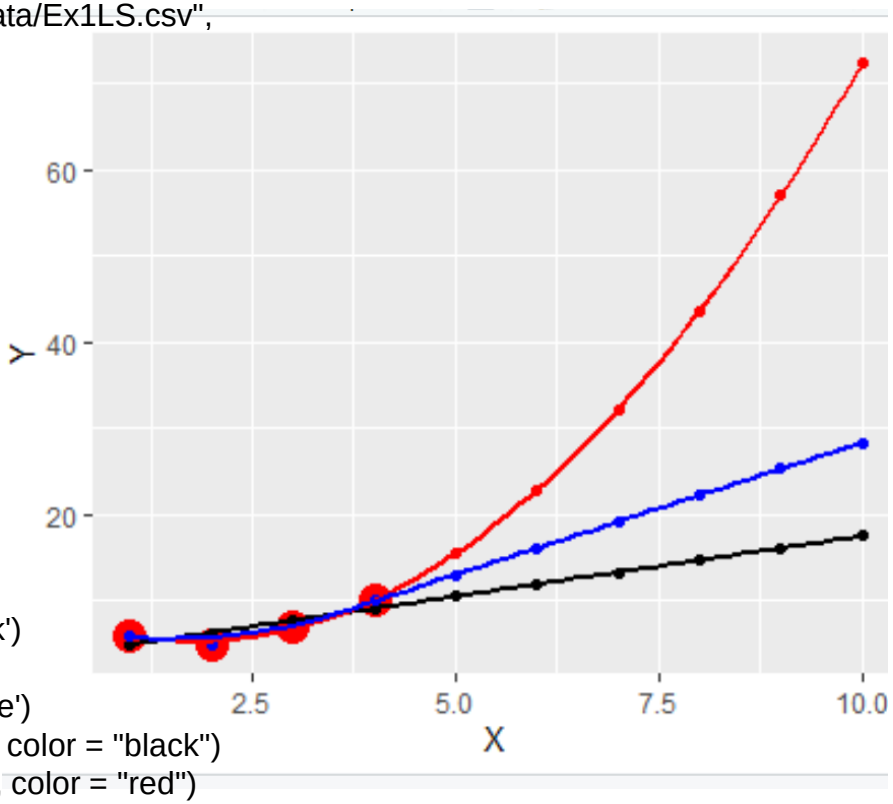
B-spline knots. B-spline curves are composed from many polynomial pieces.

Most people select knots by trail and error. Some R packages will select knots for you *(Semi-Parametric)*.



Local Regression - splines

```
library(ggplot2)
library(splines)

mydata <- read.csv(file="C:/Users/ellen/Documents/UH/Fall 2020/Data/Ex1LS.csv",
header=TRUE, sep=",")

p  = ggplot(mydata, aes(X, Y)) + geom_point(color = "red", size = 5)
p



model <- lm( formula = Y ~ X, mydata)
modelQ <- lm( formula = Y ~ X + I(X^2), mydata)
modelNS <- lm(data = mydata, Y ~ ns(X, 3))
predData <- data.frame(X = seq(1, 10, 1))

predData$Y <- predict(model, predData)
predData$Q <- predict(modelQ, predData)
predData$NS <- predict(modelNS, predData)

p <- p + geom_point(data = predData, aes(x=X, y = Y), color = 'black')
p <- p + geom_point(data = predData, aes(x=X, y = Q), color = 'red')
p <- p + geom_point(data = predData, aes(x=X, y = NS), color = 'blue')
p <- p + geom_smooth(data=predData, aes(x=X, y = Y), se=FALSE, color = "black")
p <- p + geom_smooth(data=predData, aes(x=X, y = Q), se=FALSE, color = "red")
p <- p + geom_smooth(data=predData, aes(x=X, y = NS), se=FALSE, color = "blue")
p

summary(model)
summary(modelQ)
summary(modelNS)
```



BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

splines – comparing models

```
> summary(model)

Call:
lm(formula = Y ~ X, data = mydata)

Residuals:
   1    2    3    4
 1.1 -1.3 -0.7  0.9

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.5000     1.7748   1.972    0.187
X             1.4000     0.6481   2.160    0.163

Residual standard error: 1.449 on 2 degrees of freedom
Multiple R-squared:    0.7,     Adjusted R-squared:   0.5
F-statistic: 4.667 on 1 and 2 DF,  p-value: 0.1633
```

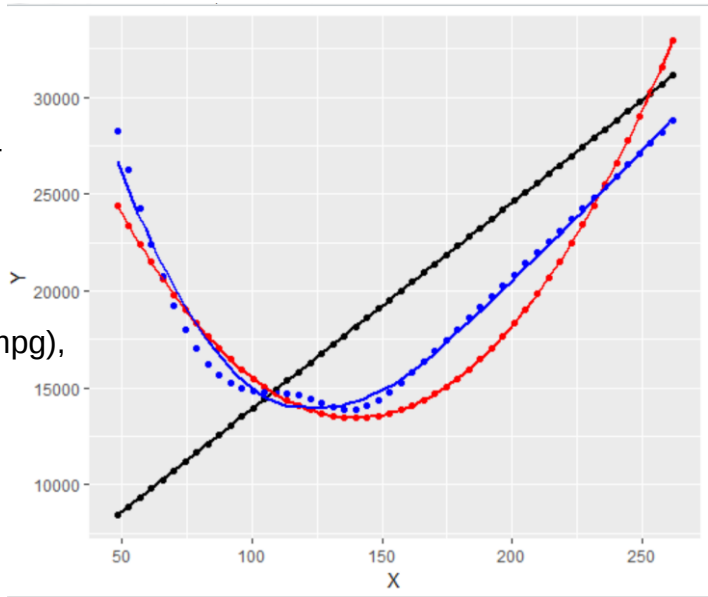
```
> summary(modelQ)

Call:
lm(formula = Y ~ X + I(X^2), data = mydata)

Residuals:
   1    2    3    4
 0.1 -0.3  0.3 -0.1

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   8.5000     1.2450   6.827   0.0926 .
X            -3.6000     1.1358  -3.170   0.1946
I(X^2)        1.0000     0.2236   4.472   0.1400
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4472 on 1 degrees of freedom
Multiple R-squared:  0.9857,    Adjusted R-squared:  0.9571
F-statistic:  34.5 on 2 and 1 DF,  p-value: 0.1195
```

```
> summary(modelNS)

Call:
lm(formula = Y ~ ns(X, 2), data = mydata)

Residuals:
   1    2    3    4
 0.1 -0.3  0.3 -0.1

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.9000     0.4359  13.536   0.0469 *
ns(X, 2)1     1.3648     1.0611   1.286   0.4207
ns(X, 2)2     4.8409     0.5848   8.278   0.0765 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4472 on 1 degrees of freedom
Multiple R-squared:  0.9857,    Adjusted R-squared:  0.9571
F-statistic:  34.5 on 2 and 1 DF,  p-value: 0.1195
```

BAUER
COLLEGE OF BUSINESS
UNIVERSITY of HOUSTON

Comparing Splines to Polynomial

```
model <- lm(price ~ horsepower + highway.mpg, data = Auto)
modelQ <- lm( formula = price ~ horsepower + I(horsepower^2) + highway.mpg + I(highway.mpg^2), Auto)
modelNS <- lm(data = Auto, price ~  ns(horsepower, 5) + ns(highway.mpg, 5))

predData <- data.frame(horsepower = seq(min(Auto$horsepower), max(Auto$horsepower), length.out = 50), highway.mpg = seq(min(Auto$highway.mpg), max(Auto$highway.mpg), length.out = 50))

predData$Y <- predict(model, predData)
predData$Q <- predict(modelQ, predData)
predData$NS <- predict(modelNS, predData)

p <- ggplot(predData, aes(x=X, y=Y))
p <- p + geom_point(data = predData, aes(x=horsepower, y = Y), color = 'black')
p <- p + geom_point(data = predData, aes(x=horsepower, y = Q), color = 'red')
p <- p + geom_point(data = predData, aes(x=horsepower, y = NS), color = 'blue')
p <- p + geom_smooth(data=predData, aes(x=horsepower, y = Y), se=FALSE, color = "black")
p <- p + geom_smooth(data=predData, aes(x=horsepower, y = Q), se=FALSE, color = "red")
p <- p + geom_smooth(data=predData, aes(x=horsepower, y = NS), se=FALSE, color = "blue")
p

summary(model)
summary(modelQ)
summary(modelNS)
```



BAUER
COLLEGE OF BUSINESS
UNIVERSITY of HOUSTON

```
> summary(model)

Call:
lm(formula = price ~ horsepower + highway.mpg, data = Auto)

Residuals:
    Min      1Q  Median      3Q     Max
-8994.5 -2433.2  -343.2  1673.5 17544.6

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4865.99    4006.57   1.215   0.2261
horsepower    142.94      15.21   9.399   <2e-16 ***
highway.mpg  -207.00      84.69  -2.444   0.0154 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4668 on 190 degrees of freedom
Multiple R-squared:  0.6704,    Adjusted R-squared:  0.667
F-statistic: 193.3 on 2 and 190 DF,  p-value: < 2.2e-16
```

```
> summary(modelQ)

Call:
lm(formula = price ~ horsepower + I(horsepower^2) + highway.mpg +
    I(highway.mpg^2), data = Auto)

Residuals:
    Min      1Q  Median      3Q     Max
-9793.8 -1925.5  -233.9  1304.9 14635.7

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      55693.8744  8516.7293   6.539 5.68e-10 ***
horsepower          -3.9421    55.6674  -0.071   0.9436
I(horsepower^2)      0.3498     0.1897   1.844   0.0667 .
highway.mpg      -2477.2185   365.0246  -6.786 1.46e-10 ***
I(highway.mpg^2)    30.1977     5.0524   5.977 1.12e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4212 on 188 degrees of freedom
Multiple R-squared:  0.7345,    Adjusted R-squared:  0.7289
F-statistic:   130 on 4 and 188 DF,  p-value: < 2.2e-16
```

```
> summary(modelNS)

Call:
lm(formula = price ~ ns(horsepower, 5) + ns(highway.mpg, 5),
    data = Auto)

Residuals:
     Min       1Q   Median       3Q      Max
-12513.7  -1938.4   -462.3   1401.7  15279.0

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)              28253       4330   6.525 6.54e-10 ***
ns(horsepower, 5)1        1755       3464   0.507  0.61302
ns(horsepower, 5)2        5208       3816   1.365  0.17403
ns(horsepower, 5)3        7332       3170   2.313  0.02187 *
ns(horsepower, 5)4       14672       7236   2.028  0.04406 *
ns(horsepower, 5)5       23015       4291   5.364 2.45e-07 ***
ns(highway.mpg, 5)1     -19027       2578  -7.380 5.46e-12 ***
ns(highway.mpg, 5)2     -21647       2970  -7.288 9.27e-12 ***
ns(highway.mpg, 5)3     -13014       3253  -4.001 9.16e-05 ***
ns(highway.mpg, 5)4     -36228       5491  -6.598 4.40e-10 ***
ns(highway.mpg, 5)5     -13509       4043  -3.341  0.00101 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4110 on 182 degrees of freedom
Multiple R-squared:  0.7553,    Adjusted R-squared:  0.7418
F-statistic: 56.17 on 10 and 182 DF,  p-value: < 2.2e-16
```

We asked lm to use a natural spline and create 5 knots. We get a different coefficient for each section, and to construct a formula, we would have to segment the data and run each through the coefficient values.

BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

GAMs

Generalized additive models (GAMs), invented by Trevor Hastie and Robert Tibshirani in 1986, have a lot of appeal:

- Relationships between the individual predictors and the dependent variable follow smooth patterns that can be linear or ***nonlinear***.
- We can estimate these smooth relationships simultaneously and then predict $\mathbf{g(E(Y))}$ by simply adding them up.

Mathematically speaking, a GAM is an additive modeling technique where the impact of the predictive variables is captured through smooth functions which, depending on the underlying patterns in the data, can be nonlinear The structure of these models generally follows:

$$g\big(E(y)\big) = \alpha + f_1(x_1) + \ldots + f_p(x_p)$$

link function *similar to splines –* ***different functions (not just different coefficients) for knots***

Where g() is the link function *(links the expected value to the predictor variables)*. The terms denote smooth, ***nonparametric*** functions *(the shape of predictor functions are fully determined by the data).*

GAMs are a type of hybrid – containing parametric and non-parametric terms. Moreover, like generalized linear models *(GLMs will be introduced in Classification)*, GAM supports multiple link functions.

For more details on how to create these smooth functions, see the section called "Splines 101"
https://multithreaded.stitchfix.com/assets/files/gam.pdf

GAMs strike a balance between the interpretable, yet biased, linear model, and the extremely flexible, “black box” learning algorithms.

Interpretability
When a regression model is additive, the interpretation of the marginal impact of a single variable (the partial derivative) does not depend on the values of the other variables in the model. Hence, by simply looking at the output of the model, ***we can estimate the effects of the predictive variables***.

Flexibility and Automation
GAMs can ***capture common nonlinear patterns*** that a classic linear model would miss. When fitting fully parametric regression models, these types of nonlinear effects are typically captured through binning or polynomials. This leads to clumsy model formulations with many correlated terms and counterintuitive results. Moreover, selecting the best model involves constructing a multitude of transformations. We don’t have this problem with GAM. Predictor functions are automatically derived during model estimation. ***We don’t have to know up front what type of functions we will need.***

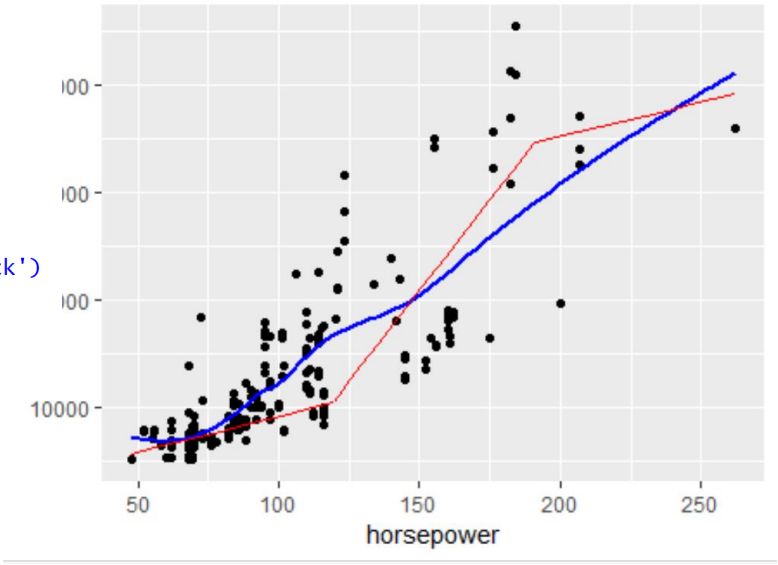
Regularization
As mentioned above, the GAM framework allows us to control smoothness of the predictor functions to prevent overfitting.

Application Note: GAMs can be very helpful in Parametric Exploratory and Diagnostic Modeling

Comparing splines to GAMs *(super simple example)*

```
library(tidyverse)
library(splines)
library(mgcv)
library(rms)

setwd("C:/Users/ellen/Documents/Fall 2018/DA2/Section1/Regression/Data")
Auto <- read.csv(file="Automobile Price Prediction.csv")

p <- ggplot(Auto, aes(x=horsepower, y=price))+geom_point()
p
modelBS <- lm(data = Auto, price ~  bs(horsepower, 3))
Auto$BSPrice <- predict(modelBS, data = Auto)
p <- ggplot(data=Auto) + geom_point(aes(x=horsepower, y = price), color = 'black')
p <- p + geom_smooth(aes(x=horsepower, y = price), se=FALSE, color = "blue")
p
```

```
Intercept <- modelBS$coefficients[1]
mb1 <- modelBS$coefficients[2]
mb2 <- modelBS$coefficients[3]
mb3 <- modelBS$coefficients[4]
rng <- range(Auto$horsepower)
div <- ((rng[2]-rng[1])/3)
## coefficients of the first model
a1 <- seq(rng[1], rng[1]+div, length.out = 10)
b1 <- seq(Intercept, Intercept+mb1, length.out = 10)
a2 <- seq(rng[1]+div, rng[1]+div+div, length.out = 10)
b2 <- seq(Intercept+mb1, Intercept+mb2, length.out = 10)
a3 <- seq(rng[1]+div+div, rng[2], length.out = 10)
b3 <- seq(Intercept+mb2, Intercept+mb3, length.out = 10)
tst <- data.frame(a1, b1, a2, b2, a3, b3)


p <- p + geom_line(data = tst, aes(x=a1, y = b1), color  = "red")
p <- p + geom_line(data = tst, aes(x=a2, y = b2), color  = "red")
p <- p + geom_line(data = tst, aes(x=a3, y = b3), color  = "red")
p
```



Splines will take these the functions for each knot, and smooth them with another function (going from red line => blue line)

```
modelGAM <- gam(price ~ s(horsepower), data = Auto, family = gaussian(link = identity)

Auto$GAMPrice <- predict(modelGAM, data = Auto)
p <- p + geom_smooth(data = Auto, aes(x=horsepower, y = GAMPrice), se=FALSE, color = "
ed")
p
```

```
> summary(modelGAM)

Family: gaussian
Link function: identity

Formula:
price ~ s(horsepower)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  13285.0      298.3   44.54   <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(horsepower) 8.717  8.976 60.45  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.738   Deviance explained =   75%
GCV = 1.8081e+07  Scale est. = 1.717e+07  n = 193
```

```
> summary(modelBS)

Call:
lm(formula = price ~ bs(horsepower, 3), data = Auto)

Residuals:
     Min       1Q   Median       3Q      Max
-11528.7  -1982.4   -743.7   1713.3  17123.8

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)             5728       1303   4.395 1.84e-05 ***
bs(horsepower, 3)1      4754       4350   1.093    0.276
bs(horsepower, 3)2     28917       3954   7.314 7.17e-12 ***
bs(horsepower, 3)3     33404       4864   6.868 9.13e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4714 on 189 degrees of freedom
Multiple R-squared:  0.6657,    Adjusted R-squared:  0.6604
F-statistic: 125.4 on 3 and 189 DF,  p-value: < 2.2e-16
```

A few points here:

1. GAMs are far more powerful for modeling regression problems than polys or splines, but with more loss of interpretability *(you're dealing with a different function for each knot)*.
2. They are very useful for non-parametric apps, and should be considered vs. SVM or Tree-Based Models, and also for prototyping and testing parametric models *(like the Bayesian models we'll see in Section II of this course)*
3. *GAM's use a link function like Generalized Linear Models (which we'll introduce in the Classification Section), but they're NOT GLM's which has broad application in parametric application.*
4. *The LOESS you see out of ggplot is a type of GAM (it uses a bandwidth parameter for df control vs knots)*

Looking Forward

Bayesian modeling is a very different approach - parameters are regularized by priors, pooling and levels which gives fine control over each parameter.

Additionally, priors gives us the ability to apply judgment to parameter values. How much does judgment influence parameter values? Depends on the evidence - in the visual to the right, we see how batting averages shrink towards a range from 200-400, but those at the extremes *(> 300)*, require more evidence *(ABs)*.

Extraordinary claims require extraordinary evidence. In the real world, we often don't have data and we must apply judgment, letting evidence weigh effect.