Statistical Smoothing:

How to Analyze data like the Fonz!

by

Matthew W. Wheeler Ph.D.

Miami University 00’ and 02’

November 8, 2019

Contents

Introduction

Why Smoothing?

The problem

We are often confronted with situations where we have no idea what the predictor-response relationship is, and a simple linear regression will not cut it. Yet we need to correctly account for this relationship to:

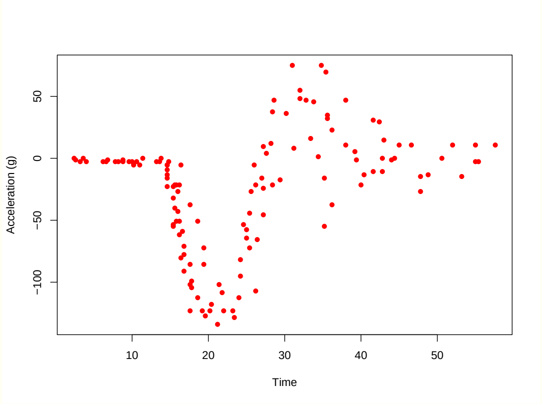
- Predict future observations.
- Properly account for one variable (possibly nuisance) as to minimize bias in a variable we are interested.

Smoothing

Smoothing is an area of statistics that seeks to fill this void. It's purpose is to create flexible model forms that will **approximate any smooth function.**

Why Smoothing?

What do you do with this data?



Acceleration data from a motorcycle accident Silverman B.W (1985). Data shows the relative acceleration of the crash test dummy in milliseconds. The crash begins at approximately 15 milliseconds.

Frame Title

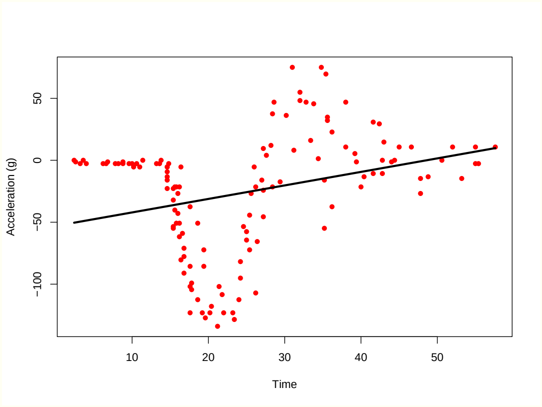
```r
library(MASS)
data(mcycle)
plot(mcycle$times,mcycle$accel,xlab="Time (ms)"
                 ,ylab = "Acceleration", pch=16,
                , col='red')

#fit a line to this
mcycle.fit.lm <- lm(accel ~ times,data=mcycle)
#plot the line
new.TIMES <- seq(1,70,1)
pred<-predict.lm(mcycle.fit.lm,
                newdata=data.frame(times=new.TIMES))
lines(new.TIMES,pred,lwd=2)

```

Why Smoothing?

A simple linear regression

$$y = \beta_0 + \beta_1 x + \epsilon$$

is a horrible idea!



Basic multivariate regression...

Standard regression review:

1. We have the relationship:

$$Y_{(N\times 1)} = X_{(N\times P)}\beta_{(P\times 1)} + \epsilon_{(N\times 1)}$$

2. We want to find the $\beta \in \mathbb{R}^P$ that is the “best” estimate of this linear relationship.
3. In most applications, we define “best” as the minimum of the sum of squares, i.e.:

$$SSE = (Y - X\beta)'(Y - X\beta)$$

4. In regression 101, we learned this estimate is

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

Basic multivariate regression..

In the above:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{bmatrix}$$

Basic multivariate regression...

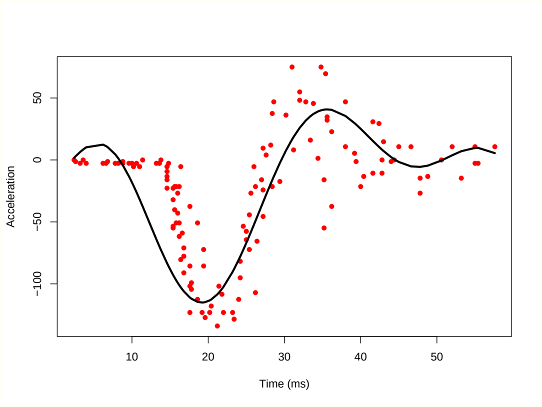
Standard regression review:

1. To properly define the function, P is usually large and x is "flexible"
2. Think x being a high degree polynomial e.g., Taylor series function.
3. Let's see what happens when $P = 10$ for a Taylor's series polynomial.

Why Smoothing?

What about something that is similar to a Taylor polynomial

$$y = \beta_0 + \sum_{i=1}^{P} \beta_i x^i + \epsilon$$



```r
#polynomial
#lets do this with standard regression
X <- matrix((mcycle$times-mean(mcycle$times))/100,
            nrow=length(mcycle$times),
            ncol=10)^matrix(1:10,
            byrow=T,nrow=133,ncol=10)
X <- cbind(rep(1,133),X)
BETA <- solve(t(X)%*%X)%*%t(X)%*%mcycle$accel
pred <- X%*%BETA
plot(mcycle$times,mcycle$accel,xlab="Time (ms)"
     ,ylab = "Acceleration", pch=16,
     , col='red')
lines(mcycle$times,pred,lwd=3)

```

Problem s galore..

1. Polynomials Work, kind of...

   The last slide kinda works, but it is very difficult to get it to work because of matrix conditioning. Taylor polynomials are "global" local approximations might be better. NEED: Something like polynomials that can model an arbitrary curve (More on splines later).

2. In our setting, $P \approx N$.
3. I have modeled cases where $N < P$ and $(X'X)^{-1}$ doesn't exist.
4. NEED: A way to estimate β (MORE on penalization later).

Problems galore... (cont)

1. Model Choices..

   In general, the fit is sensitive on the degree of the polynomial, i.e., choice of m.

2. Artifactual bumps

   We tend to get these oscillations when the data are flat.

3. Consistency

   We also want to get the true relationship as $n \to \infty$.

We want a method to prevent problems associated with model choice and ill conditioned matrices. Further, the estimate should be smooth and have as few artifactual bumps as possible.

Smoothing

Smoothing is a large field

Types of Smoothing

- LOESS smoothing.
- Spline Smoothing.
- Gaussian Process Smoothing.

All have the same needs

Correct definition of the 'smoothing' parameter.

- LOESS smoothing - this is known as the 'bandwidth'.
- Smoothing splines - this is the penalty term.
- Gaussian Process - length scale parameter and the variance.

1-D Smoothing splines

Set up

We observe :

Data: $\{(x_1, y_1), \ldots, (x_n, y_n)\}$

We Want :

A smooth estimate of the underlying relationship between x and y.

We choose:

Knots: $\{t_1, t_2, \ldots, t_p\}$

Spline Basis: A linear basis function to define our linear regression X matrix.

We fit:

A linear regression based upon the data and our choices. We want this to be smooth.

What are Splines.

Splines are polynomials, except they are defined on an interval the user defines st. $[t_i, t_{i+1}] \in \mathbb{R}$), where $\{t_1, \ldots, t_P\}$ are knots defined above. Splines are:

- Are **local** polynomials (usually of degree 3).
- Come in all types.
- Have great mathematical properties.

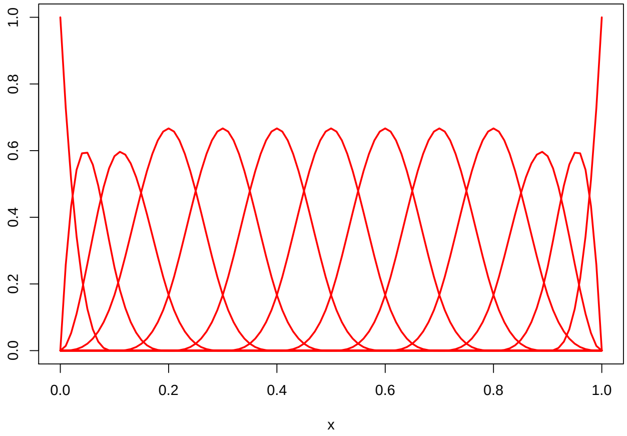
For the rest of the tutorial, we are going to look at 'B-splines' because they have great properties including numerical stability, but there are tons of other types.

Using R to understand B-splines

What do they look like?

```r
library(splines)

knots <- seq(0.1,0.9,by=0.1) #defined by the user
x     <- seq(0,1,by=0.01)    #x values to approximate
#Compute the B-Spline X matrix using the function bs()
#NOTE:
#   intercept=TRUE implies the spline has a intercept
#   degree=3 implies a local cubic polynomial
X     <- bs(x=x,knots=knots,intercept=TRUE,degree=3)
# See what they look like
plot(x,X[,1],type='l',col='red',lwd=2,ylab='')
for (ii in 2:ncol(X))
  lines(x,X[,ii],col='red',lwd=2)
```

Using R to understand B-splines



Using R to understand B-splines

Use the above code to:

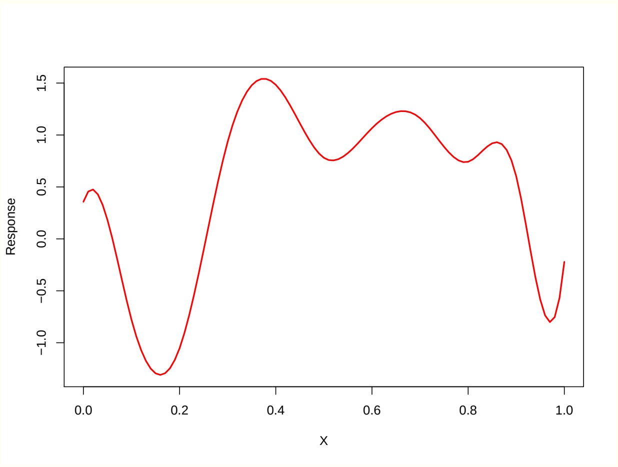
- Plot the splines when “degree = 1.”
- Plot the splines with “intercept = F.”
- Sum up the rows of the X matrix.
- Change the knot set to evenly spaced between 0.2 and 0.8, then plot.

Using R to understand B-splines

What do they look like?

```r
#using the splines to form a function
#Set the random number spline
set.seed(5551212)
#Create the beta vector in a standard linear
#regression.
beta <- matrix(rnorm(ncol(X),0,1),ncol=1)
#Estimate the function.
Y <- X%*%beta
plot(x,Y,type='l',xlab="X",
    ylab="Response",lwd=2,col=2)

```

Using R to understand B-splines



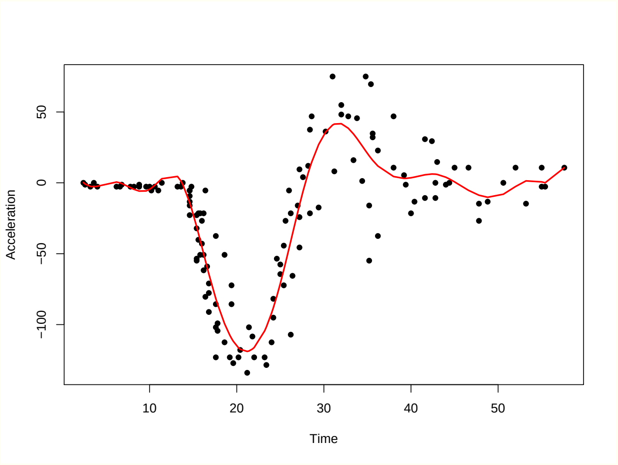
Change the seed in 'set.seed(..)' to

Fitting with our data.

Let's use B-splines with our data example.

```
library(MASS)
library(splines)
data(mcycle)
knots <- seq(min(mcycle$times),max(mcycle$times),
           by = (max(mcycle$times)-min(mcycle$times))/12)
Y     <- matrix(mcycle$accel,ncol=1)
#note I don't use the first and last not defined above
#bs ‘‘adds’’ end knots that the [min(x),max(x)]
X      <- bs(x=mcycle$times,knots=knots[2:12],
             intercept = T)
#compute betas (X'X)^{-1}X'Y
b      <- solve(t(X)%*%X)%*%t(X)%*%Y
#we can use lm() as well
#-1 removes the intercept it is already there
mcycle.lm <- lm(Y ~ X -1)
```

B-Spline Fit



Using splines we still have artifactual squiggles.

Coefficients....

```
    Coefficients:
    Estimate Std. Error t value Pr(>|t|)
X1     -3.987     17.308  -0.230  0.81821
X2     12.136     30.502   0.398  0.69145
X3    -26.390     24.007  -1.099  0.27389
X4     25.621     18.960   1.351  0.17918
X5    -38.198     11.441  -3.339  0.00113 **
X6   -159.140     13.910 -11.440  < 2e-16 ***
X7    -65.794     12.630  -5.209 8.15e-07 ***
X8     62.651     15.743   3.980  0.00012 ***
X9     22.491     14.854   1.514  0.13267
X10    -6.787     18.701  -0.363  0.71733
X11    17.069     18.794   0.908  0.36562
X12   -25.116     24.330  -1.032  0.30402
X13    13.287     36.157   0.367  0.71392
X14    -7.607     31.718  -0.240  0.81088
X15    11.011     22.924   0.480  0.63190
```

B-Spline Fit

If we look at the coefficients some observations can be made:

1. Only four coefficients are significant from zero.
2. Everything else is bouncing back and forth around zero.
3. This bouncing is related to the observed bumps.
4. If we could remove these bumps, the curve would be smoother.

We need to define a way to make these estimates close to zero. Such an estimate would be smoother than the standard linear model estimate.

Intuition of smooth?

If you don't know who these two are, you still can probably guess which one is "smooth."

What is smooth?

Technical Point

The point of the previous slide is that we have an intuitive and formal definitions we can appeal to.

- Intuitively we don’t want the slope to be “big” without enough information.
- Formally we want the derivative to be small over the function $f(x)$ (i.e. $\int_{-\infty}^{\infty}[f''(t)]^2dt.$)
- In fact, this can be used to penalize our SSE term.

What is smooth?

More formally we want to penalize the sum of square error across the entire function.
That is:

$$(Y - X\beta)'(Y - X\beta) + \lambda \int_{-\infty}^{\infty} [f''(t)]^2 dt.\lambda$$

The above looks daunting, but there are a lot of fun theorems about B-splines that can reduce it to finding the minimum of the form:

$$(Y - X\beta)'(Y - X\beta) + \lambda\beta'\Omega\beta$$

Our solution given λ and *Omega* is:

$$\beta = (X'X + \lambda\Omega)^{-1}X'Y$$

Special Quantities

Here λ and Ω are special quantities that we talk about more and are used behind the scenes. Typically:

Penalty

λ is the penalty term it has an optimal value that minimize the penalized SSE. This value is found using Cross Validation.

Penalty Matrix

Ω helps define the “distance” used in the penalty term. This value is typically fixed or known before hand.

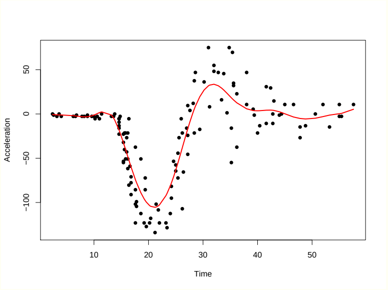
What is smooth?

Suppose we let $\lambda = 1$ and $\Omega = I_P$

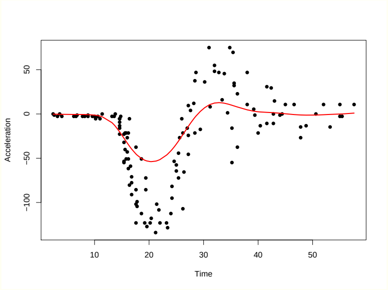
```
#####################################
lambda <- 1 #penalty smaller
I      <- diag(ncol(X))
b1    <- solve(t(X)%*%X+*lambda*I)%*%t(X)%*%Y
#####################################
lambda <- 10 #penalty larger
I      <- diag(ncol(X))
b2     <- solve(t(X)%*%X+lambda*I)%*%t(X)%*%Y
#####################################
```

Note: For this example we **ARE USING** ridge regression! and as $\lambda \to 0$, we get no penalty. As $\lambda \to \infty$ we get a flat horizontal line.

What is smooth?



$\lambda = 1$



$\lambda = 10$

They both remove bumps, but what is the correct value of λ?
NOTE: This is where the term shrinkage comes from. Values are being shrunk back to zero.

What is smooth?

With the above code:

- Play with the value of λ. What happens when it gets large or small?
- Try to do the same thing by adding knots.
- Try to take knots away.
- How are the figures the same/different?

What is smooth?

What I did was naive. Ridge regression is nice, but it is not the answer most people look for. It over penalizes when there is a signal.

There are tons of different approaches people use. We are going to focus on a simple methodology called P-splines, or penalized splines. It can be looked at as something similar to ridge regression?

P-splines

One could argue that the β values are related. If one is large, then there should be an increased probability the one right after it is large. One can formalize this idea using a random walk:

$$\beta_i = \beta_{i-1} + \epsilon_i \tag{1}$$

Here, ϵ is usually “small.” Such a set up would assume that the $\beta's$ are correlated, and if the function is changing rapidly in one region, then it should change almost as rapidly in the next region.
This can be formalized by specifying Ω

P-splines

First order random walk:

$$\Omega = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

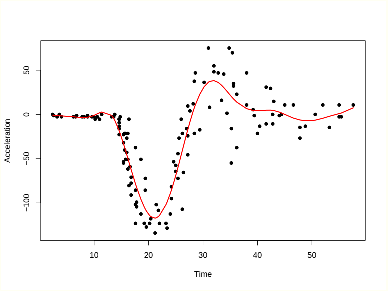
Second order random walk:

$$\Omega \times \Omega = \begin{bmatrix} 5 & -4 & 1 & 0 & 0 \\ -4 & 6 & -4 & 1 & 0 \\ 1 & -4 & 6 & -4 & 1 \\ 0 & 1 & -4 & 6 & -4 \\ 0 & 0 & 1 & -4 & 5 \end{bmatrix}$$

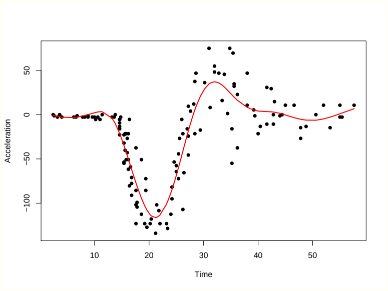
Code for P-splines

```r
Omega <- diag(ncol(X))*2
for (i in 2:ncol(X)){
  Omega[i,i-1] = -1; Omega[i-1,i] = -1;
}
#first order
lambda <- 0.25
b1     <- solve(t(X)%*%X+lambda*Omega)%*%t(X)%*%Y
#second order
lambda <- 0.25
b2      <- solve(t(X)%*%X+lambda*Omega%*%Omega)%*%t(X)%*%Y
```

What is smooth?



1st order random walk



2nd order random walk

Cross Validation

Now we still need to find an optimal value of λ, so how do we do it?

Cross Validation: Main Idea

Imagine that we don't observe n observations in our data set. Instead we $n-1$ of the observations, The one we don't observe y_n (i.e., the one we hold out from the fit) can then be predicted from the $n-1$ observations. For different λ, we can calculate the predicted squared error

$$\frac{\sum_{i=1}^{n}[Y_i - \hat{f}_{-i,\lambda}(x_i)]^2}{n}$$

where $\hat{f}_{-i,\lambda}(x_i)$ is the estimate of $f(x)$ without observation i. The minimum of this value across all λ is our winner!.

Labor Intensive Cross Validation

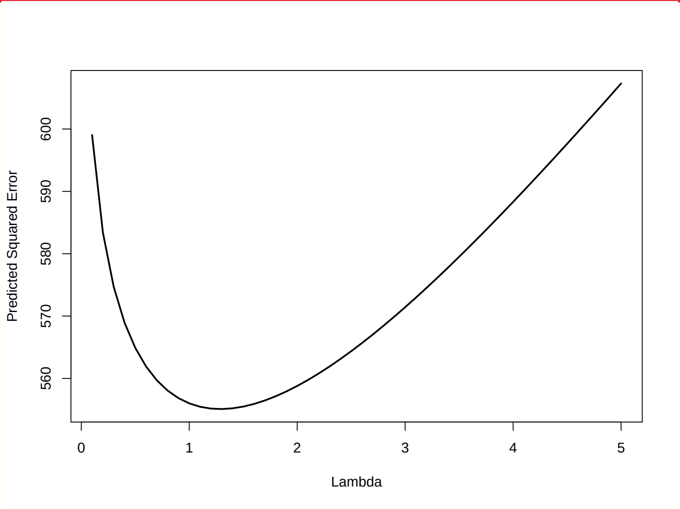
Set it all up:

```
library(splines)
library(MASS)
data(mcycle); attach(mcycle)
P = 31 #number of splines + 1
knots <- seq(min(times),max(times),by =
                (max(times)-min(times))/P)
#remove the knot associated with the min
#and max time
knots = knots[-1]; knots = knots[-31];
X = bs(times,knots=knots)
#compute Omega
Omega <- diag(ncol(X))*2
for (i in 2:ncol(X)){
  Omega[i,i-1] = -1; Omega[i-1,i] = -1;
}
```

Labor Intensive Cross Validation

```r
#our possible lambdas
lambda = seq(0.1,5.0,by=0.1)
#our Predicted Squared Errors
PSE = rep(0,length(lambda))
for (j in 1:length(lambda)){
  for(i in 1:length(accel)){
    tX = X[-i,]
    tB = solve(t(tX)%*%tX+lambda[j]*Omega)
                    %*%t(tX)%*%accel[-i]
    Y_pred = X[i,,drop=F]%*%tB
    PSE[j] = PSE[j]+1/length(accel)*(accel[i]-Y_pred)^2
  }
}
plot(lambda,PSE,type='l',xlab="Lambda",lwd=2
        ylab="Predicted Squared Error")
```

Best λ

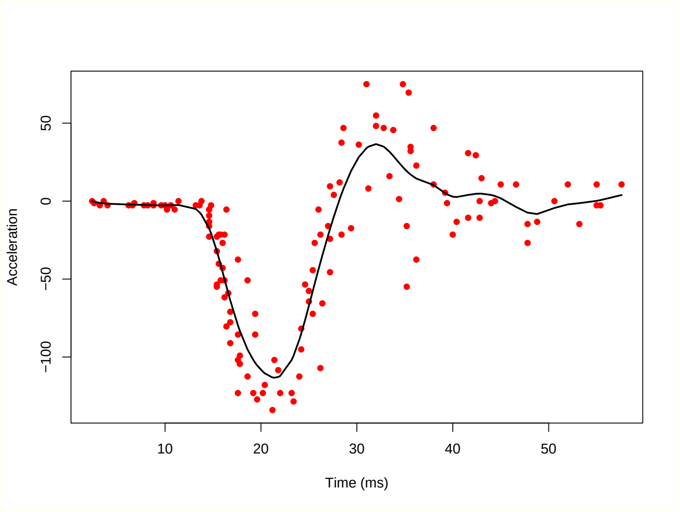


So the best estimate is around $\lambda = 1.3$

Labor Intensive Cross Validation

```r
#estimate using 1.3 for lambda
B = solve(t(X)%*%X+1.3*Omega)%*%t(X)%*%accel
nY = X%*%B
#plot the answer
plot(times,accel,xlab="Time (ms)"
     ,ylab = "Acceleration", pch=16,
     , col='red')
lines(times,nY,lwd=2)
```

Prediction with LOO



What about 2nd order random walk

Your turn:

To Try:

- Do this with the second order random walk by squaring Omega.
- Try increasing to a larger number, what happens?

NOTE: You can do this with other types of analyses as well. For example, in LOESS you can adjust bandwidth parameter.

Cross Validation

Fortunately there are some nice formulas that don't make this too computationally expensive. I am not going to give them here.

Notes:

- One can approximate the degrees of freedom in our exercise.
- $df(\lambda) \approx trace(X[X'X + \lambda\Omega]^{-1}X)$
- This is about 13.8 in our example, most other smoothing estimates will be similar.

P-splines ...

Observations:

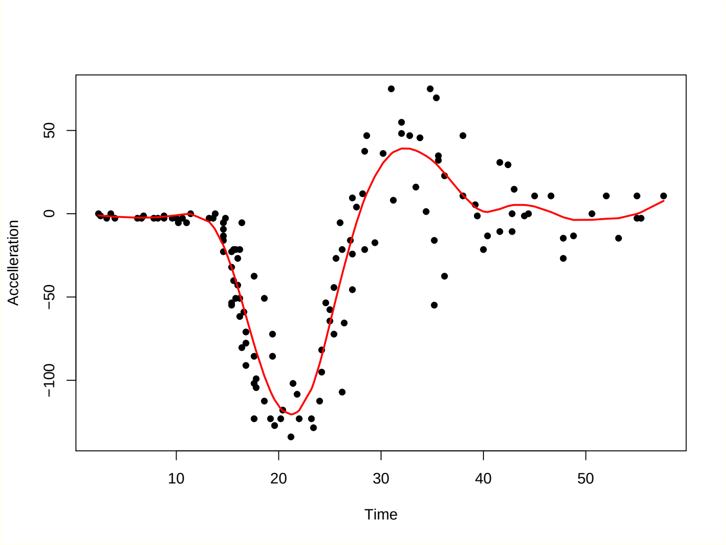
- The λ term always needs an “optimal” value. This is computed with Generalized Cross Validation or some other technique. I like the Bayesian approach.
- There are a bunch of ways to compute the optimal values of λ and they don’t always agree on the exact same number, but the curve is qualitatively the same.
- The Bayesian method requires a little more knowledge than I can impart in two hours.
- For simplicity, we focus on this using the R package “pspline” and “mgcv”.
- The general approach is to put in a large number of evenly spaced splines (e.g. 40+).

Code for P-splines

P-spline code using R library “pspline.”

```r
library(MASS)
library(pspline)
attach(mcycle)
nt = sort(times,index.return=T)
#Fit a pspline model
fit.pspline<- sm.spline(times,accel)
plot(mcycle$times,mcycle$accel,ylab="Acceleration",
     xlab="Time",pch=16)
lines(predict(fit.pspline)$x,predict(fit.pspline)$ysmth,
      col=2,lwd=2)
```

P-splines Estimate



Cross Validation

```

Call:
smooth.Pspline(x = ux, y = tmp[, 1],
               w = tmp[, 2], method = method)

Smoothing Parameter (Spar): 8.375531
Equivalent Degrees of Freedom (Df): 13.34478
GCV Criterion: 312.5231
CV  Criterion: 310.8485
```

sm.spline

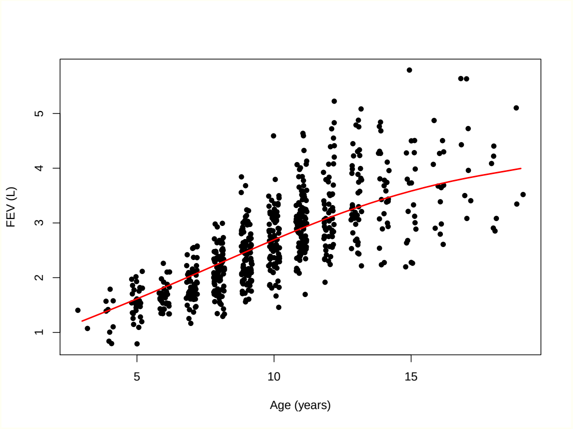
- sm.spline( ) produces a smoothed estimate of the data.
- It automatically estimates λ using generalized cross validation (GCV) or cross validation (CV).
- Generalized Cross Validation is CV = FALSE
- Cross Validation is CV = TRUE
- You can also fix the degrees of freedom and do many other things with this function.

FEV Data

```
# A tibble: 6 x 5
  age.yrs fev.L ht.in ind.Male ind.Smoke
    <dbl> <dbl> <dbl>    <dbl>     <dbl>
1       9  1.71  57          0         0
2       8  1.72  67.5        0         0
3       7  1.72  54.5        0         0
4       9  1.56  53          1         0
5       9  1.90  57          1         0
6       8  2.34  61          0         0
```

Your Turn

Make this figure with ’sm.spline’.



Code for P-splines

Some starting code to fill out.

```r
    library(readr)
    fev_data <- read_table2("./fev_data.txt")
    #####################################
    plot(jitter(fev_data$age.yrs),
         fev_data$fev.L,xlab = "Age (years)",
         ylab="FEV (L)",pch=16,col=1)
    #####################################
    library(pspline)
    x <- as.matrix(fev_data$age.yrs)
    y <- as.matrix(fev_data$fev.L)
```

Code for P-splines

The solution:

```
#fit p-spline smooth
fit.ps <- sm.spline(x,y)
#fit new ages
x.new = seq(3,19,0.5)
#predict new observations
pred.gf <-predict(fit.ps,x.new)
lines(x.new,pred.gf,col=2,lwd=2)
```

Generalize Additive Models

Let’s revisit our FEV-1 data

```
  age.yrs fev.L ht.in ind.Male ind.Smoke
    <dbl> <dbl> <dbl>    <dbl>     <dbl>
1       9  1.71  57          0         0
2       8  1.72  67.5        0         0
3       7  1.72  54.5        0         0
4       9  1.56  53          1         0
5       9  1.90  57          1         0
6       8  2.34  61          0         0
```

We don’t just have one variable in age, we have three others that may influence our result. But how do we include them given what we did above?

What about other variables?

So we have a few problems:

- Smoothing is about accurate prediction not necessarily inference. Often what we are smoothing is a nuisance parameter.
- What I have shown you does **nothing** to help us with additional variables.
- You can generalize this whole thing and have some smoothing components, and some non-smoothing components.

GAM

We now want to consider models of the following form:

$$Y = X\beta + f_1(c^1) + \ldots f_K(c^K) + \epsilon$$

Where:

X is a matrix of covariates of **interest**.
β is the vector of coeffiecients we estimate.
$c^1, \ldots, c^K$ are covariates that have a smooth relationship to Y.
$f_1, (\cdot) \ldots, f_K(\cdot)$ are smooth functions estimated as above.

GAM(cont.)

P-spline

Now the same p-spline game above applies here. There are more options and I can't really explain everything, but you can include P-spline components to your model **additively**. This is where the 'Generalized Additive Model' comes in. It is a additive model that essentially accepts smoothing components.

Covariates

Now with $c^1, \ldots, c^K$ properly accounted for, I can look at the effects of X. Here, standard statistical methods still apply (e.g., p-values and hypothesis testing).

In R...

A GAM model can be fit using the “mgcv” package using the “gam( )” function

```
gam(formula,family=gaussian(),data=list(),
weights=NULL,subset=NULL,
na.action,offset=NULL,method="GCV.Cp",
optimizer=c("outer","newton"),
control=list(),scale=0,
select=FALSE,knots=NULL,sp=NULL,min.sp=NULL,
H=NULL,gamma=1,
fit=TRUE,paraPen=NULL,G=NULL,
in.out,drop.unused.levels=TRUE,
drop.intercept=NULL,...)
```

Using ”gam( )”

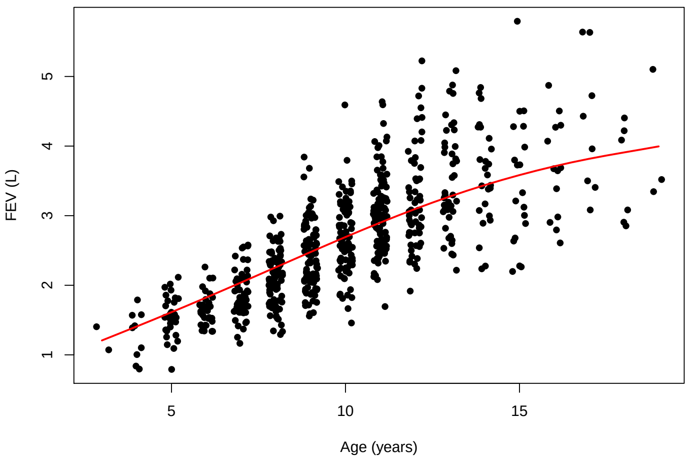
Now there is way too much this function does, but if you know how to use lm(), you will be fine.

```r
#load the mgcv package for the
#gam function
library(mgcv)
fit.model <- gam(fev.L~ ind.Male +
                        s(fev_data$age.yrs,bs='ps'),
                 data=fev_data)
summary(fit.model)
```

Output...

```
Formula:
fev.L ~ ind.Male + s(fev_data$age.yrs, bs = "ps")

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.46872    0.02967  83.207  < 2e-16 ***
ind.Male     0.32712    0.04141   7.899 1.21e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
                      edf Ref.df     F p-value
s(fev_data$age.yrs) 3.782  4.539 226.3  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.628   Deviance explained = 63.1%
GCV = 0.28212  Scale est. = 0.27962   n = 654
```

GAM Estimate



s( ) is for SMOOTH

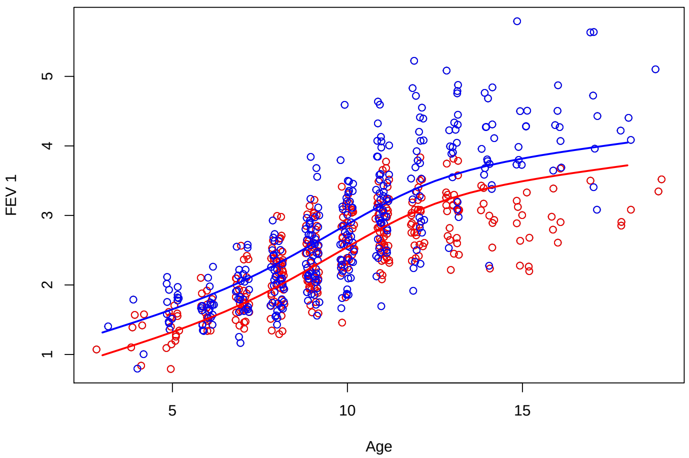
The function “s ( )” tells the gam ( ) function that we want a smooth function over that particular covariate.

- We use bs = "ps"to get P-splines.
- We can have as many s( ) functions as covariates we need smoothed.
- We can also use the predict statement in much the same way. Everything is handled behind the scenes.

Predicting Males and Females..

```r
#New Data
nage = seq(3,18,by=0.5)
nsex = rep(1,length(nage)) #female
fsex = rep(0,length(nage))
#Predict  males and females
predict.m <- predict(fit.model,
                     newdata=data.frame(age=nage,sex=nsex,
                     length=length(nsex)),
                     type="response")
predict.f <- predict(fit.model,
                     newdata=data.frame(age=nage,sex=fsex,
                     length=length(fsex)),
                     type="response")
```

GAM Estimate by sex



More Smooths?

```r
#GAM on more than one parameter
age = fev_data$age.yrs
sex = fev_data$ind.Male
height = fev_data$ht.in
smoke = fev_data$ind.Smoke
fev = fev_data$fev.L
fit.model <- gam(fev ~ sex + smoke
                           + s(age,bs='ps')
                           + s(height,bs='ps'))
summary(fit.model)
```

Results

```
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.60271    0.02421 107.508  < 2e-16 ***
sex          0.09278    0.03356   2.764  0.00587 **
smoke       -0.13681    0.05767  -2.372  0.01798 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘

Approximate significance of smooth terms:
            edf Ref.df     F  p-value
s(age)    2.271  2.869 20.23 6.73e-12 ***
s(height) 6.324  6.971 79.97  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘

R-sq.(adj) =  0.796   Deviance explained = 79.9%
GCV = 0.15638  Scale est. = 0.1536    n = 654
```

Thoughts

Analysis

Now we have a model that incorporates all of the information. Age and height are included as smooth covariates, both are highly significant. We also have an R-squared of 0.80. Most interesting is the smoke variable effect. If we look at a standard regression that has an R-squared of 0.77, the p-value is 0.14 for this variable. Using the GAM it is 0.02.

Note: There clearly is a linkage between smoking and lung capacity, but we don't have a lot of smokers in this dataset and thus little power. The data sets is of children under 20. There are not a lot of 3 year olds smoking a pack a day.

To Do:

Now try:

- Plotting the smooth across ages given smoking status.
- Plot the effect of height on FEV1 in the model.

Hint: You need to select the other covariates.

Bayesian Analogue?

What we have been doing

We have been finding the minimum of:

$$(Y - X\beta)'(Y - X\beta) + \lambda\beta'\Omega\beta$$

Where λ is estimated using cross validation. Now the minimum is the mode of a specific Bayesian posterior.

Bayes Rule

For those who don't know/remember, Bayes theorem expresses the uncertainty of the parameter β given data as a probability distribution.

$$P(\beta, \Theta|Y) \propto \ell(Y|\beta, \Theta)Pr(\beta, \Theta),$$

where $\ell(\cdot)$ is the likelihood of the data given β and some nuisance parameters Θ (e.g. σ^2 and λ).

The Connection

Now assume that one has

$$Pr(\beta) \sim N(0, \lambda\Omega^-1)$$

Ignoring the prior on the variance and smoothing term λ then $log[P(\beta, \Theta \mid Y)]$ is

$$\approx \frac{-1}{2\sigma^2}(Y - X\beta)'(Y - X\beta) - \frac{\lambda}{2}\beta'\Omega\beta$$

Which looks mighty similar to the above!

In fact, the main difference is that a Bayesian does put a prior over the variance, usually inverse-gamma, and does put a prior over λ, usually a gamma random variable. In this way, we don't have to

worry about CV we estimate the variance and smoothing parameter as part of the model.

Note

You can estimate these two parameters using optimization techniques. In this case, it is called empirical Bayes, which is very much related to cross validation.

Conclusions

Smoothing

Topic

As mentioned before smoothing is a large topic that I just touched on here. There are many ways to smooth. Splines are just one variety, some others are:

- LOESS regression
- Kernel regression
- Gaussian process regression

In the background, you can show they are all related.

Smoothing

With these slides you should be able to smooth your own data, and have some intuition about what is going on when you run the smooth.spline() function. Also, you can use the GAM approach to do actual data analyses. Your typical statistics 101 knowledge applies to the output.

Further Reading

Smoothing

Stasinopoulos, MD, Rigby RA, Heller GZ, Voudouris V, De Bastiani, F (2017) Flexible Regression and Smoothing. CRC Press

Wang, Y (2011) Smoothing Splines: Methods and Applications CRC Press

Generalized Additive Models

Wood, SN. (2017) Generalized Additive Models: An Introduction with R, 2nd ED. CRC Press

Hastie, TJ., Tibshirani RJ., (1990) Generalized Additive Models CRC Press