Nonparametric Data Analysis for the Social Sciences

社会科学的非参数数据分析

JEFF GILL
Distinguished Professor
Department of Government, Department of Mathematics & Statistics
Founding Director, Center for Data Science
Member, Center for Neuroscience and Behavior
Editor-in-Chief, Political Analysis
American University

Motivation

- Sometimes we do not a priori have a specific model or parametric assumption in mind.
- Two typical uses: bivariate visualization and modeling.
- Two modeling purposes: general data exploration (a good thing), a commitment to reduce the usual number of distributional assumptions (sometimes a good thing).
- So sometimes these tools are used as a precursor to full model specification in the traditional parametric (especially Bayesian) sense.
- Also, sometimes this will suggest transformations of the data to convenient forms (sometimes referred to as the "non-linear in the parameters" approach).
- Note: nothing is truly "nonparametric," but this term is too ingrained to avoid.

Smoothing, Goals

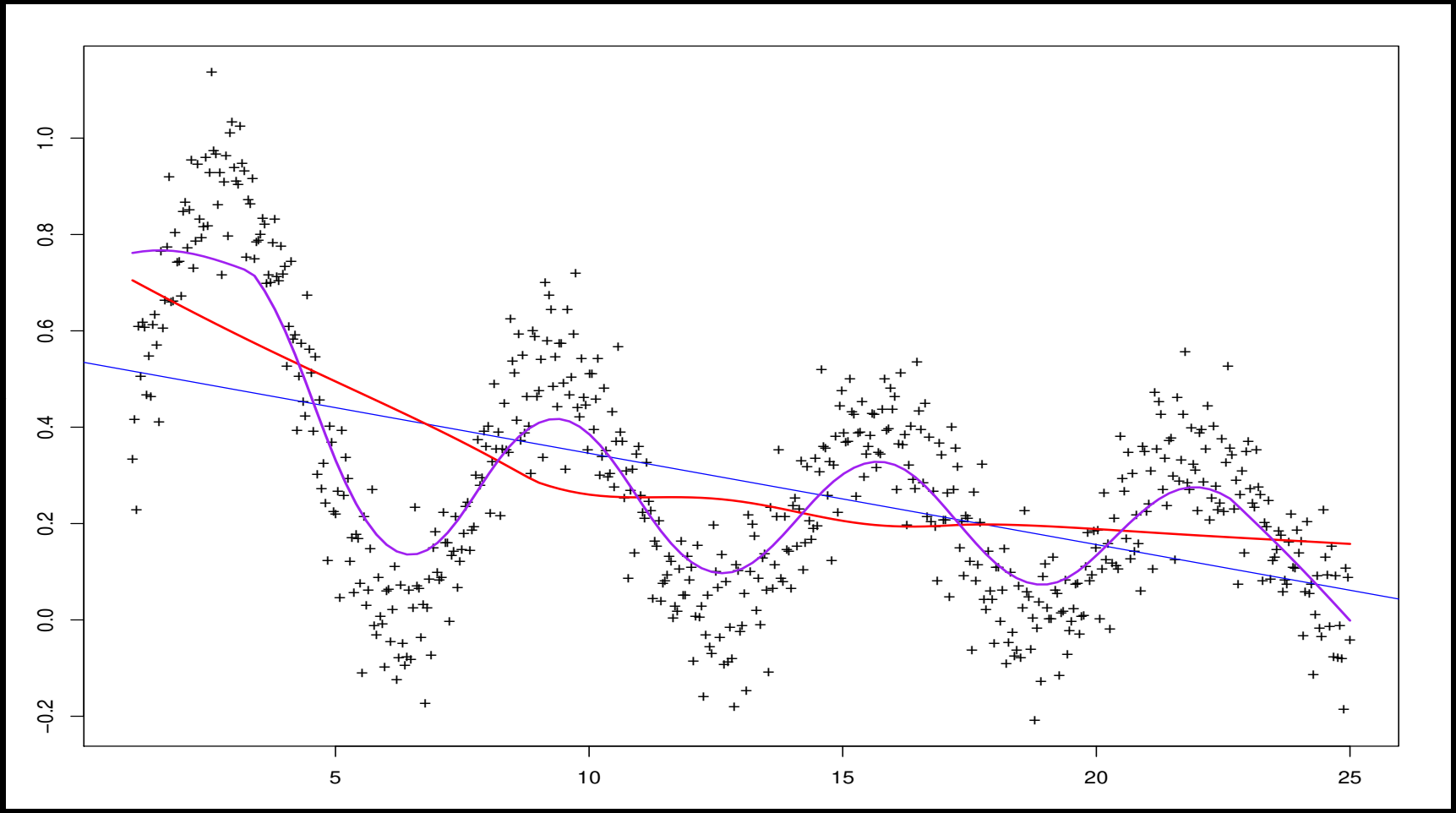
- A tool for summarizing the trend of an outcome variable as a function of explanatory variables (often only one).
- Designed to be less variable than the data itself (hence "smooth").
- How smooth do we want to be?
- For a nonlinear trend:
  - ▷ too much smoothing: variance ↓, and bias ↑,
  - ▷ too little smoothing: variance ↑, and bias ↓,

  where bias in this context means missing curvilinear features.
- Linear regression is then infinitely smooth.
- Pointwise interpolation is then infinitely unsmooth (rough).

Running the Lowess Smoother

```
x <- seq(1,25,length=600)
y <- (2/(pi*x))^(0.5)*(1-cos(x)) + rnorm(100,0,1/10)

par(mar=c(3,3,2,2), bg="white")
plot(x,y,pch="+")

ols.object <- lm(y~x)
abline(ols.object,col="blue")

lo.object <- lowess(y~x,f=2/3)
lines(lo.object$x,lo.object$y,lwd=2,col="red")

lo.object <- lowess(y~x,f=1/5)
lines(lo.object$x,lo.object$y,lwd=2,col="purple")
```

Running the Lowess Smoother



Smoothing, Starting Vocabulary

- We smooth by adjusting data points vertically through weighting to be more "harmonious" with their neighbors.
- The bivariate case is usually called scatterplot smoothing.
- The key smoothing decision is the determination of the size of the neighborhood around each point.
- Larger neighborhoods lead to more smoothness since points further out are included in the weighting.
- We then slide this neighborhood from left to right adjusting the point in the middle.
- The span is defined as the proportion of the total points included in the neighborhood:

$$\omega = \frac{2K+1}{n}$$

  so there are K points on either side of the point to be smoothed.
- One complication: the ends of the data.

Illustrative Beginning Example

- To test memory retrieval Kail and Nippold ("Unconstrained Retrieval From Semantic Memory", 1984, *Child Development*, 944-951) asked 8, 12, and 21 year olds to name as many animals and pieces of furniture as possible in separate seven minute intervals.
- They find that this number increases across the tested age range but that the rate of retrieval slows down as the period continues.
- In fact, the responses often came in "clusters" of related responses ("lion," "tiger," "cheetah," etc.), where the relation of time in seconds to cluster size is fitted to be

$$cs(t) = at^3 + bt^2 + ct + d,$$

  where time is t, and the others are estimated parameters (which differ by topic, age group and subject).
- There are strong theoretical reasons that $b = -18a$ from the literature.
- The researchers were very interested in the inflection point of this function since it suggests a change of cognitive process.

Illustrative Beginning Example

dog . . . (1 sec) . . . *cat* . . . (3 sec) . . . *bird* . . . (8 sec) . . . *lion* . . . (2 sec) . . . *tiger*. If a pause time of 1 sec or less is taken to reflect items retrieved from the same cluster (i.e., $t \leq 1$), then *dog*/*cat* would be from the same cluster; the remaining words would represent different clusters. In this case, $cf(1) = 1$ and $N = 5$, so the mean cluster size is $5/(5 - 1)$, or 1.25, reflecting three one-word clusters and one two-word cluster. Continuing the analysis, $cf(2) = 2$, so the mean cluster size is $5/(5 - 2) = 1.67$. Again verifying this result, with $t \leq 2$ sec as a criterion, clusters consist of *dog*/*cat*, *bird*, and *lion*/*tiger*. $cf(3) = cf(4) = cf(5) = cf(6) = cf(7) = 3$, hence the mean cluster size for $t = 3$–7 is $5/(5 - 3) = 2.5$. Finally, $cf(8) = 4$, so the mean cluster size is $5/(5 - 4) = 5$.

Cluster sizes computed in this manner are depicted in the right panel of Figure 2 as a function of t for the cumulative frequency data depicted in the left-hand panel of that figure. The cluster size function, like the cumulative frequency distribution, has a plateau between 5 and 7 sec. As before, this plateau corresponds to the break between the two distributions of pause times.

The final issue to be considered is how to identify the precise point at which the initial decelerating curve begins to accelerate, for this value differentiates the longer pause times associated with retrieval of clusters from the briefer pause times associated with rapid emission of items. In fact, functions like those depicted in Figure 2 are well described by a third-order polynomial of the type

$$cs(t) = at^3 + bt^2 + ct + d, \qquad (2)$$

where cs refers to cluster size and t is time in seconds. Further, the second derivative of this polynomial, $-b/3a$, corresponds to the inflection point at which the function stops decelerating and starts accelerating. Once this inflection point is known, pauses in the retrieval protocol can be identified unambiguously as reflecting either search for additional clusters or emission of items from within a cluster. Then one can derive the number of clusters as well as the average size of clusters in the retrieval protocol.

Cluster sizes were calculated for each individual's retrieval protocol for t ranging from 2 to 10 sec. These cluster values were then fit to equation (2) with STEPIT. The estimated values of a and b were used to calculate the inflection point of the cluster size function. Of the 39 individuals, six had at least one protocol that included either a negative number or an extraordinarily large

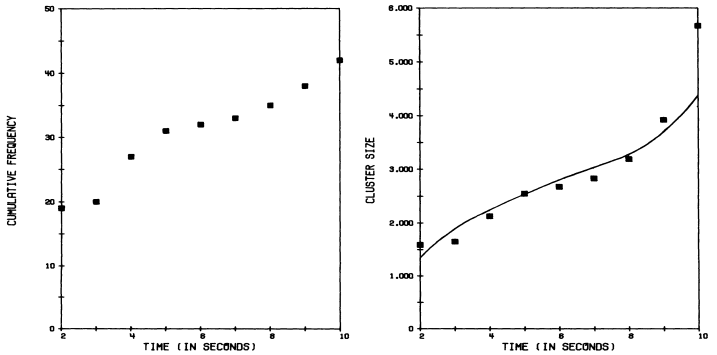


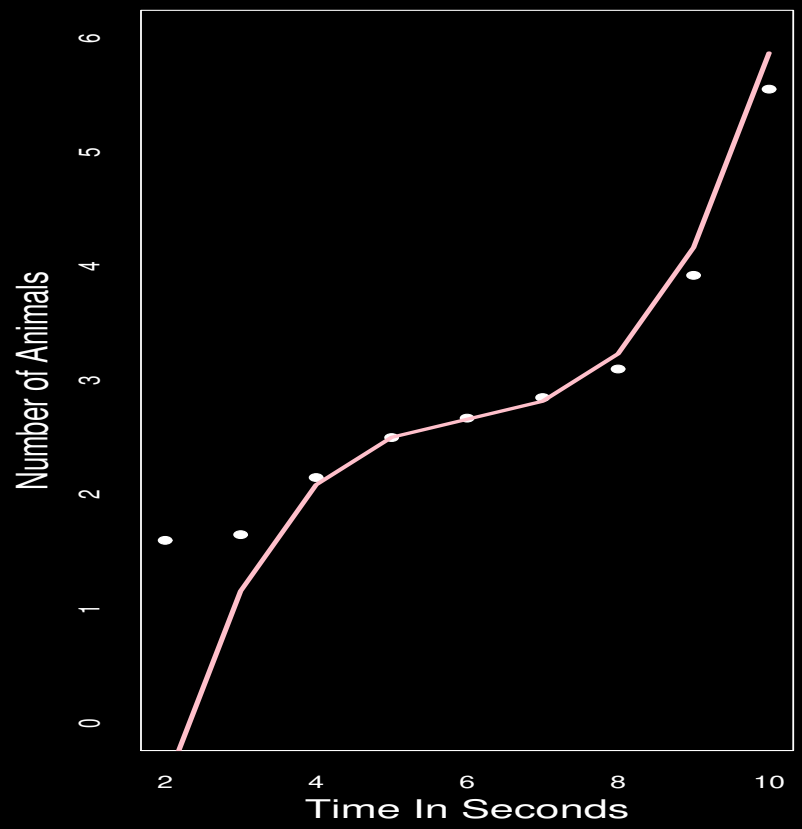
FIG. 2.—Cumulative frequency of pause times (left panel) and cluster size (right panel) as a function of time for one 8-year-old. The function in the right panel is derived from the best-fitting values of the a and b parameters from equation (2).

Illustrative Beginning Example

- We can specify hard-coded values of the parameters (below) by trial and error.

```
cs <- c(1.6,1.65,2.15,2.5,2.67,2.85,3.1,3.92,5.55)
seconds <- 2:10

cog <- function(a,c,d,t) a*t^3 + (-18*a)*t^2 + c*t + d

postscript("Class.Stat.Comp/cognitive2a.ps")
par(mfrow=c(1,1),mar=c(5,5,3,3),oma=c(6,6,6,6),col.axis="white",col.lab="white",
    col.sub="white",col="white",bg="black")
plot(seconds,cs,pch=19,ylim=c(0,6),xlab="",ylab="")
cs.vals <- cog(a=0.04291667,c=4.75,d=-7.3,t=seconds)
# try a=0.0405,c=5.03,d=-6.36
lines(seconds,cs.vals,col="pink",lwd=3)
mtext(side=1,line=2.5,cex=1.5,"Time In Seconds")
mtext(side=2,line=2.5,cex=1.5,"Number of Animals")
dev.off()
```

Nonlinear (Weighted) Least-Squares



Illustrative Beginning Example

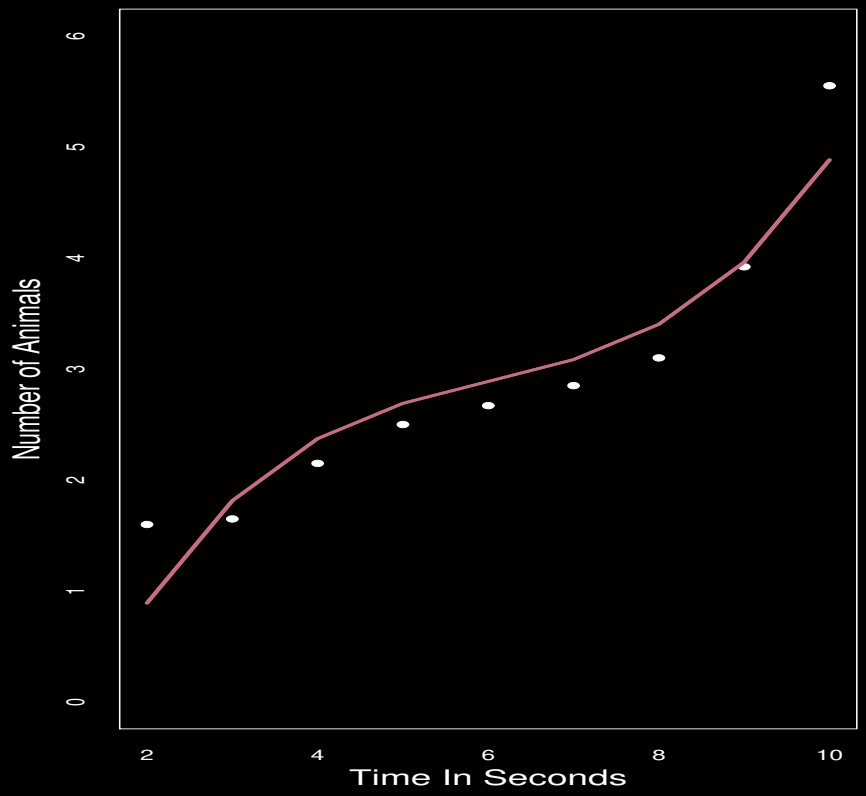
- We can also use the R function nls to estimate these by minimizing residuals:

```
cog.df <- data.frame(seconds=seconds,cs=cs)
cog.nls <- nls(cs ~ a*seconds^3 + (-18*a)*seconds^2 + c*seconds + d,
               start=c(a=10,c=10,d=-10),trace=TRUE); summary(cog.nls)

  Estimate Std. Error t value Pr(>|t|)
a  0.02013    0.01211   1.662   0.1476
c  2.35048    1.16637   2.015   0.0905
d -2.52056    1.79998  -1.400   0.2109
Residual standard error: 0.4572 on 6 degrees of freedom
```

Illustrative Beginning Example

```
postscript("Class.Stat.Comp/cognitive2b.ps")
par(mfrow=c(1,1),mar=c(4,4,4,4),oma=c(3,3,3,3),col.axis="white",col.lab="white",
    col.sub="white",col="white",bg="black")
plot(seconds,cs,pch=19,ylim=c(0,6),xlab="",ylab="")
lines(seconds,cs.vals,col="lightsteelblue4",lwd=3)
mtext(side=1,line=2.5,cex=1.5,"Time In Seconds")
mtext(side=2,line=2.5,cex=1.5,"Number of Animals")
cs.vals <- cog(a=summary(cog.nls)$parameters[1,1],
               c=summary(cog.nls)$parameters[2,1],
               d=summary(cog.nls)$parameters[3,1],
               t=seconds)
lines(seconds,cs.vals,col="palevioletred3",lwd=3)
dev.off()
```

Illustrative Beginning Example



General Expression For Smoothers

- Now consider the general model:

$$y_i = f(x_i) + \epsilon_i$$

where $f()$ is an unspecified (for now) smooth, nonlinear function, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

- One choice of the function: **Scatterplot Smoother:**

$$\hat{f}(x_i) = \sum_{j=1}^{n} s_{ij} y_j$$

$s_{ij} = s(x_i, x_j)$, some weighting function,

x_i, point to be smoothed (moved),

x_j, all other points: $1, \ldots, n$

y_j, all outcome variable values: $1, \ldots n$

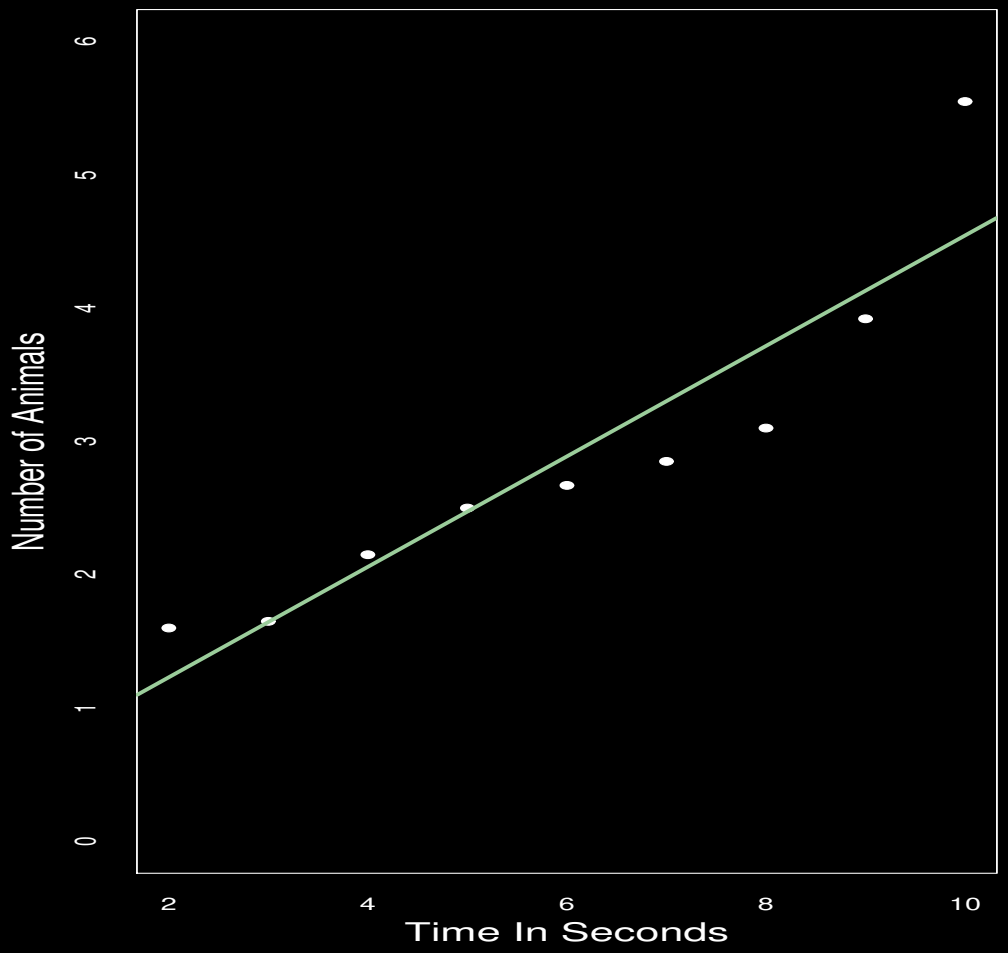
- The key decision (as we'll see) is the choice of s_{ij} through neighborhood treatment: large neighborhoods or diffuse functions produce less variable and more smooth fits with greater bias, and small neighborhoods or narrow functions produce more variable and less smooth fits with less bias.

Linear Regression as a Smoother

- Smoother #1: linear regression.
  - $\hat{f}(x_i) = \alpha + x_i\beta$.
  - Called "infinitely smooth" since the second derivative is zero everywhere on the line.

```
postscript("Class.Stat.Comp/cognitive2c.ps")
par(mfrow=c(1,1),mar=c(5,5,3,3),oma=c(3,3,3,3),col.axis="white",col.lab="white",
    col.sub="white",col="white",bg="black")
plot(seconds,cs,pch=19,ylim=c(0,6),xlab="",ylab="")
abline(lm(cs.vals~seconds),col="darkseagreen3",lwd=3)
mtext(side=1,line=2.5,cex=1.5,"Time In Seconds")
mtext(side=2,line=2.5,cex=1.5,"Number of Animals")
dev.off()
```

Linear Regression as a Smoother

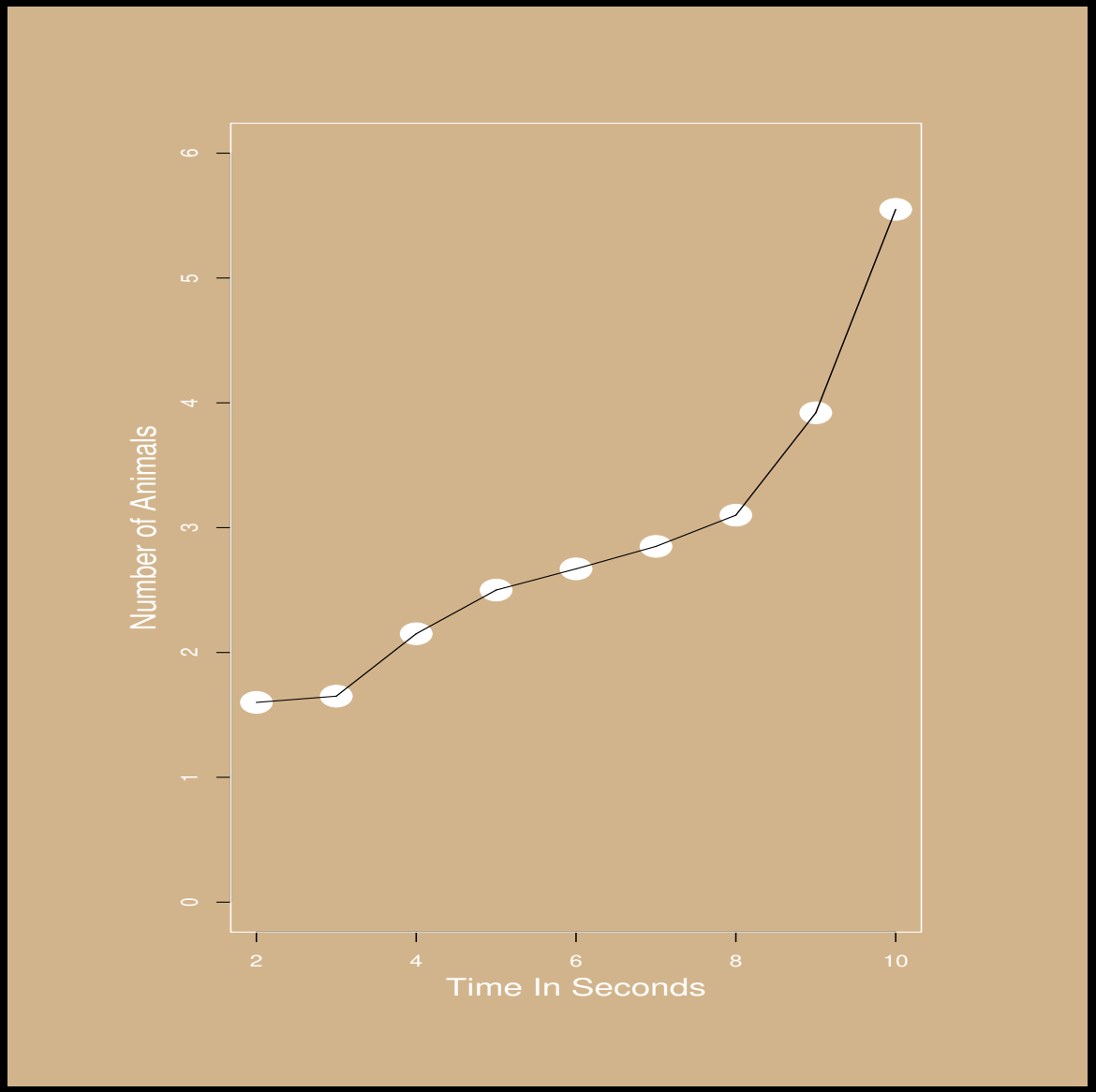


Interpolation as a Smoother

- Smoother #2: interpolation.
  - $\hat{f}(x_i) = x_i$.
  - Called "infinitely rough" since it just connects the points.

```
postscript("Class.Stat.Comp/cognitive2d.ps")
par(mfrow=c(1,1),mar=c(5,5,3,3),oma=c(3,3,3,3),col.axis="white",col.lab="black",
    col.sub="white",col="white",bg="tan")
plot(seconds,cs,pch=19,ylim=c(0,6),xlab="",ylab="",cex=3)
for(i in 1:(length(seconds)-1))
    segments(seconds[i],cs[i],seconds[i+1],cs[i+1],cex=2)
mtext(side=1,line=2.5,cex=1.5,"Time In Seconds")
mtext(side=2,line=2.5,cex=1.5,"Number of Animals")
dev.off()
```

Interpolation as a Smoother



Bin Smoother

- Smoother #3: bin smoother (regressogram). Looks very "notchy" like a city skyline. Steps:

1. Choose $J+1$ thresholds (cutpoints) along the x-axis:

$$-\infty < c_0 < c_1 < \ldots < c_{J-1} < c_J < \infty$$

2. Now "bin" the data by collecting values between the thresholds. The j th bin contains points such that:

$$R_j = \{i : c_j \leq x_i < c_{j+1}\}$$

(i.e. collect the points in R_j that are greater than c_j and less than c_{j+1}).

3. Within each bin calculate an average AVE (mean, median, or mode), and assign it to each of the bin values:

$$s_{ij} = AVE(y_i | i \in R_j)$$

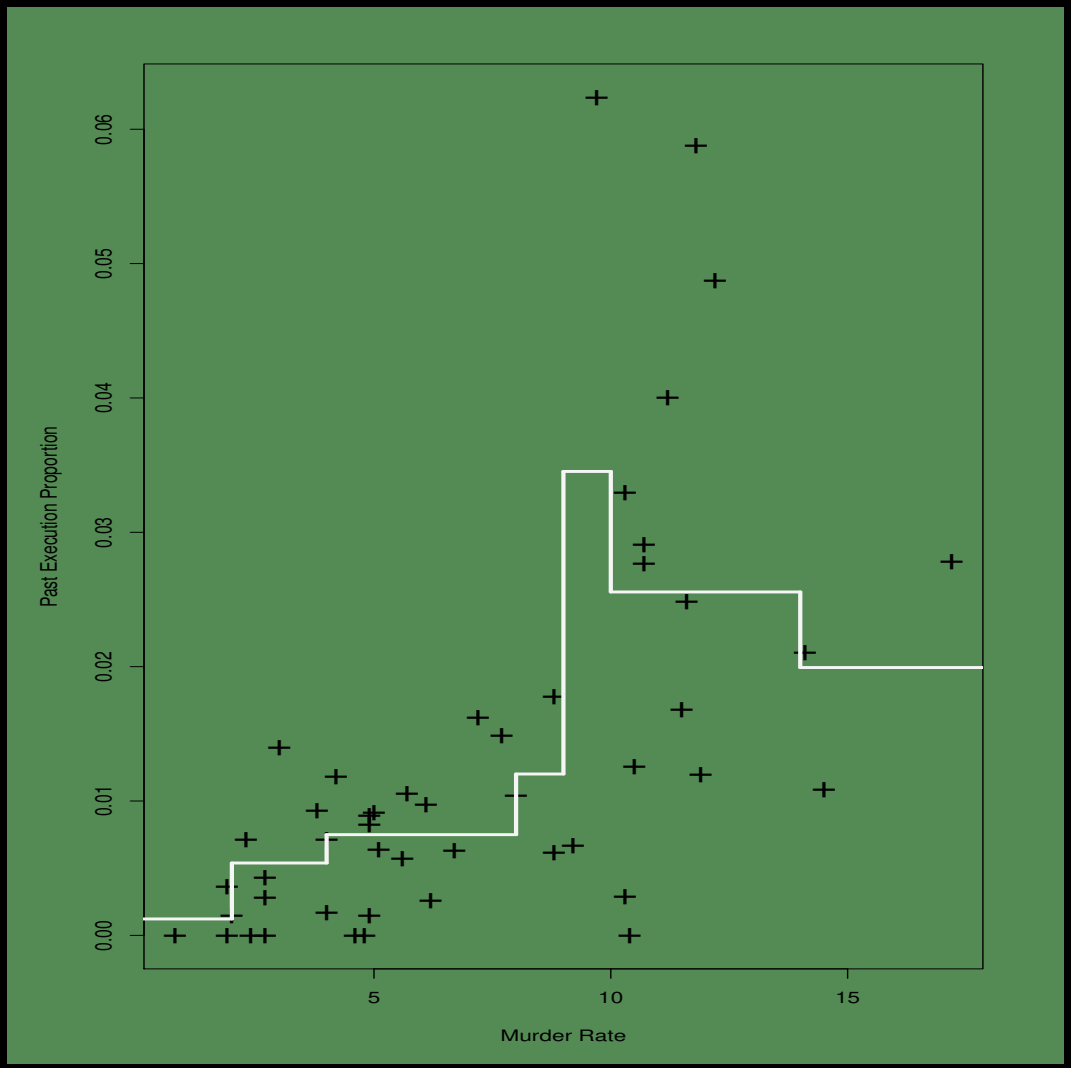
- Window widths can be equal (default) or customized.
- Two issues: it looks "choppy" and sometimes picking a good value for J.

Bin Smoother

- Code:

```
postscript("Class.Stat.Comp/cognitive2e.ps")
par(mfrow=c(1,1),mar=c(5,5,3,3),bg="palegreen4")
x1 <- seq(0.25,10,length=40)
y1 <- tan(x1)/x1^2 + rt(length(x1),10)
plot(x1,y1,pch="+",lwd=3,xlab="",ylab="",xlim=c(0,10))

x1.cuts <- c(0,2,4,6,8,10)
y1.means <- rep(NA,(length(x1.cuts)-1))
for (i in 1:length(y1.means))
    y1.means[i] <- mean(y1[x1 > x1.cuts[i] & x1 < x1.cuts[i+1]])

for (i in 1:length(y1.means))  {
    segments(x1.cuts[i],y1.means[i],x1.cuts[i+1],y1.means[i],lwd=2)
    if (i != 1) segments(x1.cuts[i],y1.means[i-1],x1.cuts[i],y1.means[i],lwd=2)
}
dev.off()
```

Bin Smoother



Running Mean Smoothers

- General Idea: for each point in the data, pick k points to the left and k points to the right on the x-axis and take the mean on the y-axis of these points.
- This defines a neighborhood for each point.
- Small values of k produce rough plots and large values of k produce smooth plots.
- Also called the "symmetric nearest neighborhood" smoother or "simple neighborhood" smoother.

Running Mean Smoothers

- Steps:
  - $N(x_i)$ is the neighborhood of point i determined by w, which is the window size/span defined as the proportion of other points included. Note that $0 < w < 1$ is constant.
  - More formally, assume n points (odd number for notational convenience), and define:

$$N(x_i) = \left[\max\left(\frac{\lfloor wn \rfloor - 1}{2}, 1 \right), \ldots, i-1, i, i+1, \ldots, \min\left(\frac{\lfloor wn \rfloor - 1}{2}, n \right) \right]$$

    where $\lfloor wn \rfloor$ is the "floor" of wn, i.e. the lower integer component ($\lfloor 3.14 \rfloor = 3$).
  - So $N(x_i)$ gives indices of x's to include that denote x_i and $\frac{\lfloor wn \rfloor - 1}{2}$ points on either side (smaller near the endpoints though).
  - Now apply the smoother of choice, $\hat{f}(x_i)$ inside each neighborhood to get values for each x_i.

Running Mean Smoothers

- Running-mean type:

$$\hat{f}(x_i) = AVE(y_j | j \in N(x_i)) = AVE_{j \in N(x_i)}(y_i)$$

- Running-line (OLS fit of points in $N(x_i)$) type:

$$\hat{f}(x_i) = \hat{\alpha} + \hat{\beta} x_i$$

Running Mean Smoothers

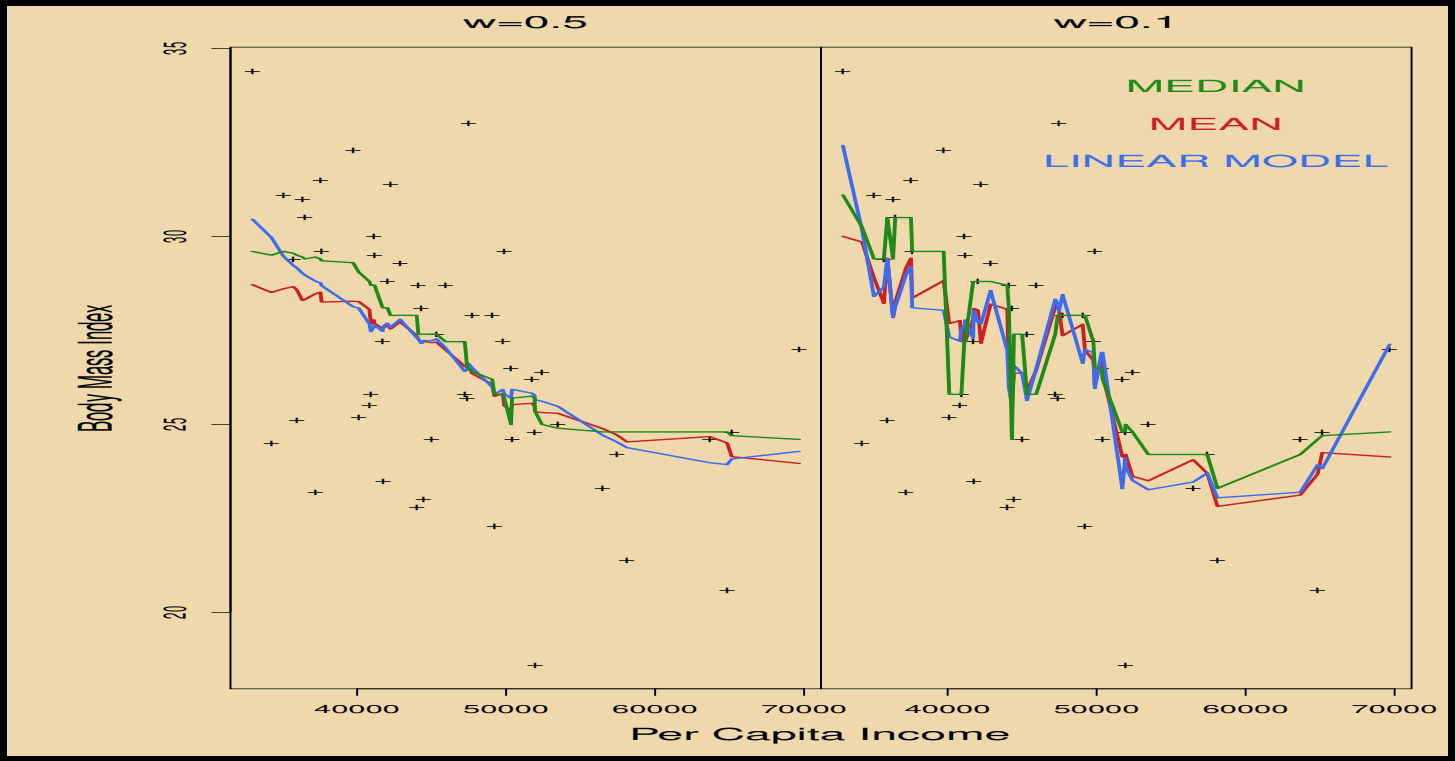
- R function is `smooth`, but we can write our own:

```
x0 <- seq(1,25,length=101); n <- length(x0)
y0 <- exp(2/(pi*x0))^(1.5) + rnorm(n,0,1/10)
f.rm <- f.rl <- f.rd <- rep(NA,length=n)
w <- 0.3

for (i in 1:n)  {
    lower <- max(i-floor((w*n-1)/2),1);  upper <- min(i+floor((w*n-1)/2),n)
    neighborhood <- lower:upper
    f.rm[i] <- mean(y0[neighborhood])
    f.rd[i] <- median(y0[neighborhood])
    lin.fit <- lm(y0[neighborhood] ~ x0[neighborhood])
    f.rl[i] <- lin.fit$coefficients %*% c(1,x0[i])
}
```

Running Mean Smoothers

```
# postscript("Class.Stat.Comp/cognitive2f.ps")
par(mfrow=c(1,1),mar=c(5,5,5,5),bg="wheat2")
plot(x0,y0,pch="+",lwd=1,xlab="",ylab="")
lines(x0,f.rm,col="firebrick3")
lines(x0,f.rl,col="royalblue2")
lines(x0,f.rd,col="forest green")
mtext(side=3,cex=1.3,line=2,"Comparison of Neighborhood Smoothers")
# dev.off()
```

Running Mean Smoothers



Running Line Smoother From R

- The R `smooth` function is actually Tukey's running median smoother, which is not so useful.
- Bratton and Van De Walle (1997) make available data on regime transition for 47 sub-Saharan countries over the period from each country's colonial independence to 1989, with some additional variables collected for the period 1990 to 1994. Included are 99 variables describing governmental, economic, and social conditions for the 47 cases. Also provided are data from 106 presidential and 185 parliamentary elections, including information about parties, turnout, and political openness.
- Focus on two variables: economic debt and growth.
- Let's look at Friedman's "super smoother" instead.

```
supsmu(x, y, wt, span = "cv", periodic = FALSE, bass = 0)
```

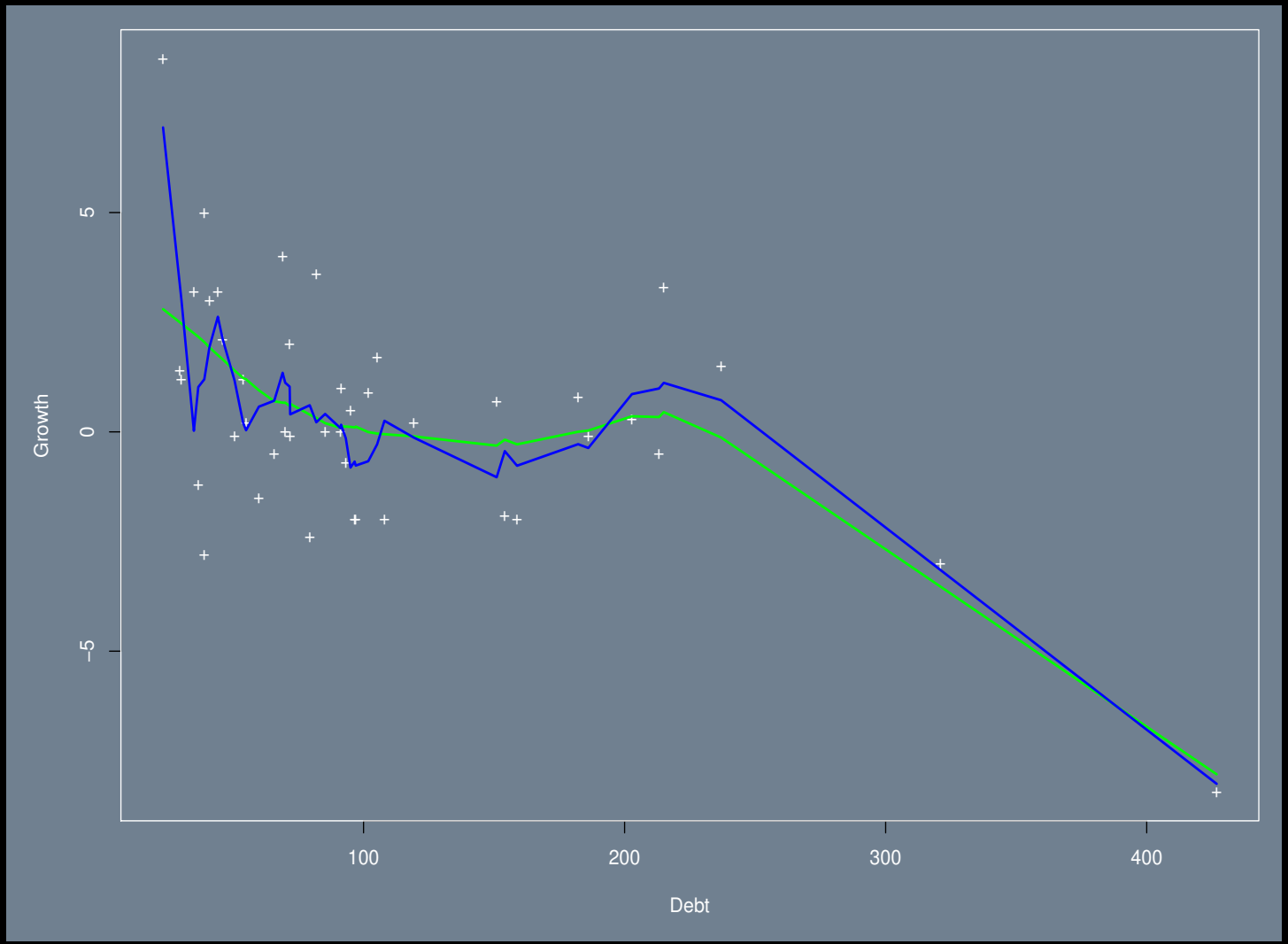
where:

`wt`	case weights, equal by default
`span`	the fraction of observations in the span of the running lines smoother, use "cv" to choose this by cross-validation (default).
`periodic`	if TRUE, x values assumed to be in [0, 1] and of period 1.
`bass`	smoothness control: values of up to 10 indicate increasing smoothness.

Running Line Smoother From R

```
africa <- read.table("http://jgill.wustl.edu/data/africa.dat",
    row.names=1,header=TRUE)
africa2 <- cbind(africa$DEBT,africa$GROWTH)[!is.na(apply(africa2,1,sum)),]
postscript("Class.Stat.Comp/africa.smooth.ps")
par(mar=c(5,5,1,1),col.axis="white",col.lab="white",col.sub="white",
    col="white",bg="slategray")
plot(africa2,pch="+",xlab="Debt",ylab="Growth")
super <- supsmu(africa2[,1],africa2[,2])
lines(super$x,super$y,col="green",lwd=2)
super <- supsmu(africa2[,1],africa2[,2],span=0.01,bass=2)
lines(super$x,super$y,col="blue",lwd=2)
dev.off()
```

Running Line Smoother From R



Kernel Smoothers

- An extension of the running-line smoother where explicit weights are now included.
- Standard idea: weight the points closer to x_i more than remote points.
- Again, pick k points to the left and k points to the right of the x_i point, producing a neighborhood size of $2k+1$.
- Define the jth weight for the ith point as the function:

$$\omega_{ij} = cd\left(\left|\frac{x_i - x_j}{\lambda}\right|\right) \qquad \text{for } j \in N_{2k+1},\ 0 \text{ otherwise}$$

where:

  - c is a normalizing constant such that $\int s(u)du = 1$:

$$c = \left[\sum_{i=1}^{2k+1} d\left(\left|\frac{x_i - x_j}{\lambda}\right|\right)\right]^{-1}$$

  - λ is the window width: $2k+1$,
  - and $d(t)$ is a decreasing function in $t = |x_i - x_j|/\lambda$.

Kernel Smoothers

- So

$$\hat{y}_i = s(y_i|\mathbf{X}) = \sum_{j=1}^{2k+1} \omega_{ij} y_i.$$

- Common forms:

$$d(t) = \phi(t) \qquad \text{[Gaussian]}$$

$$d(t) = \begin{cases} \frac{3}{4}(1-t^2), & |t| \leq 1 \\ 0, & \text{otherwise} \end{cases} \qquad \text{[Epanechnikov]}$$

$$d(t) = \begin{cases} \frac{3}{8}(3-5t^2), & |t| \leq 1 \\ 0, & \text{otherwise} \end{cases} \qquad \text{[minimum variance]}$$

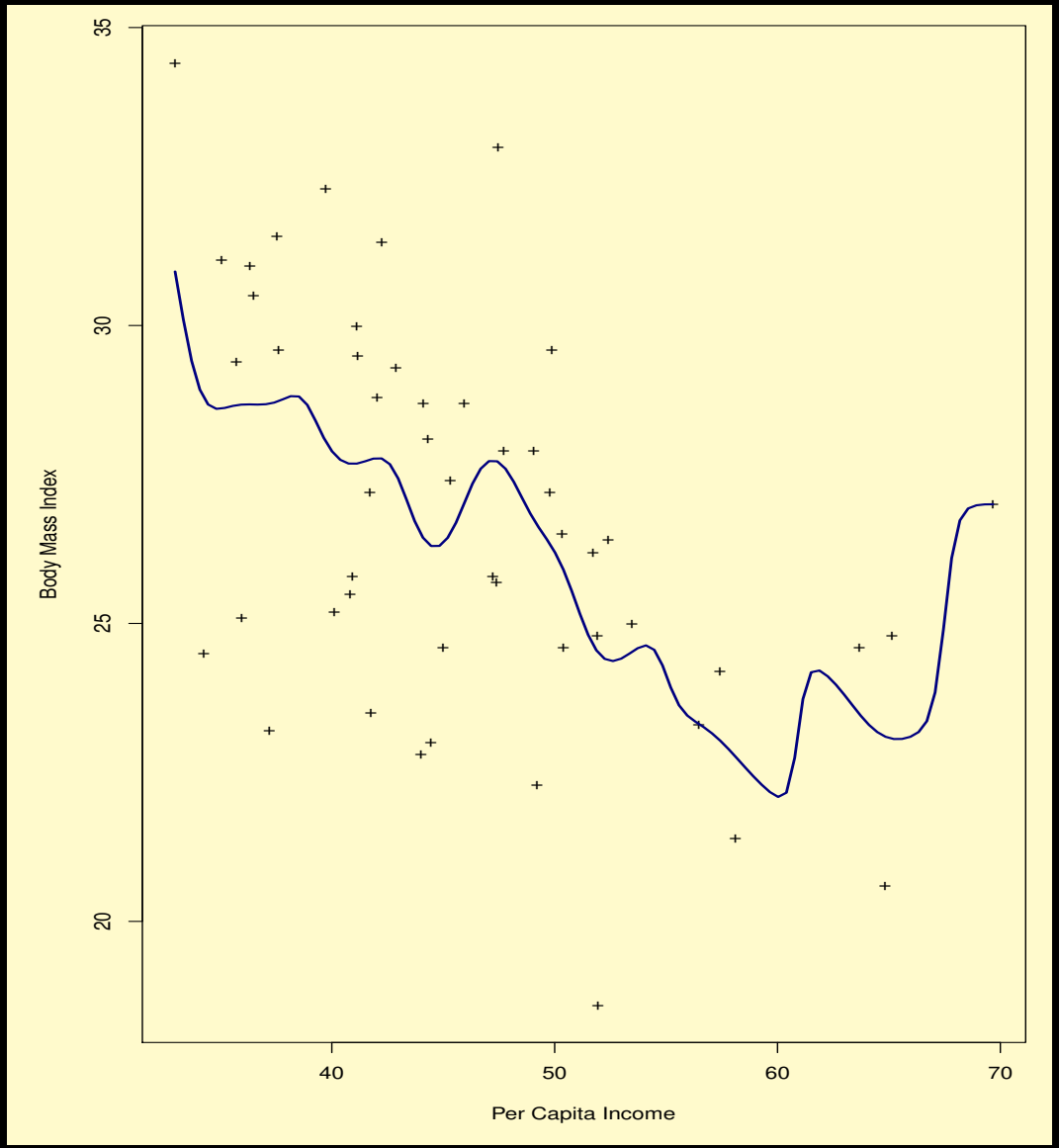
- Others: box, triangle, Parzen, miscellaneous polynomials.

Kernel Smoothers

- Using the R function ksmooth:

```
postscript("Class.Stat.Comp/cognitive2g.ps")
par(mfrow=c(1,1),mar=c(5,5,5,5),bg="lemonchiffon")
plot(x0,y0,pch="+",lwd=1,xlab="",ylab="")
lines(ksmooth(x0,y0,kern="normal",bandwidth=2),col="navy")
mtext(side=3,cex=1.3,line=2,"Gaussian Kernel Smoother")
dev.off()
```

Kernel Smoothers



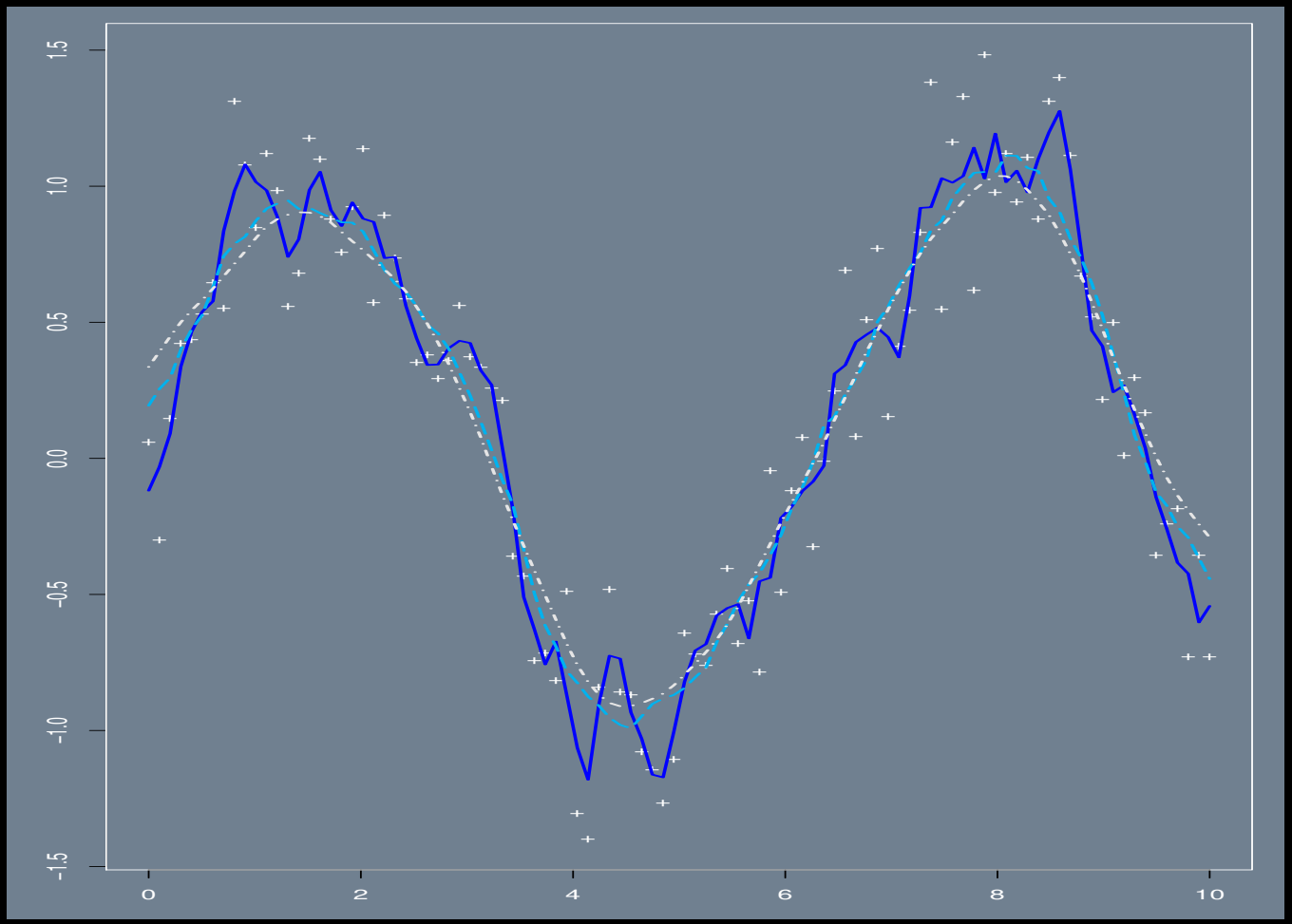
Kernel Smoothers, Rolling Our Own

```
epan <- function(x) ifelse(abs(x) <= 1, 0.75*(1-x^2), 0)

k.sm <- function(x,y,k)  {
    s.y <- y
    for (i in 1:length(x))  {
        lo <- ifelse(i-k >= 1, i-k, 1)
        hi <- ifelse(i+k <= length(x), i+k, length(x))
        w <- epan(x[i] - x[lo:hi])/sum(epan(x[i] - x[lo:hi]))
        s.y[i] <- y[lo:hi] %*% w
    }
    s.y
}
```

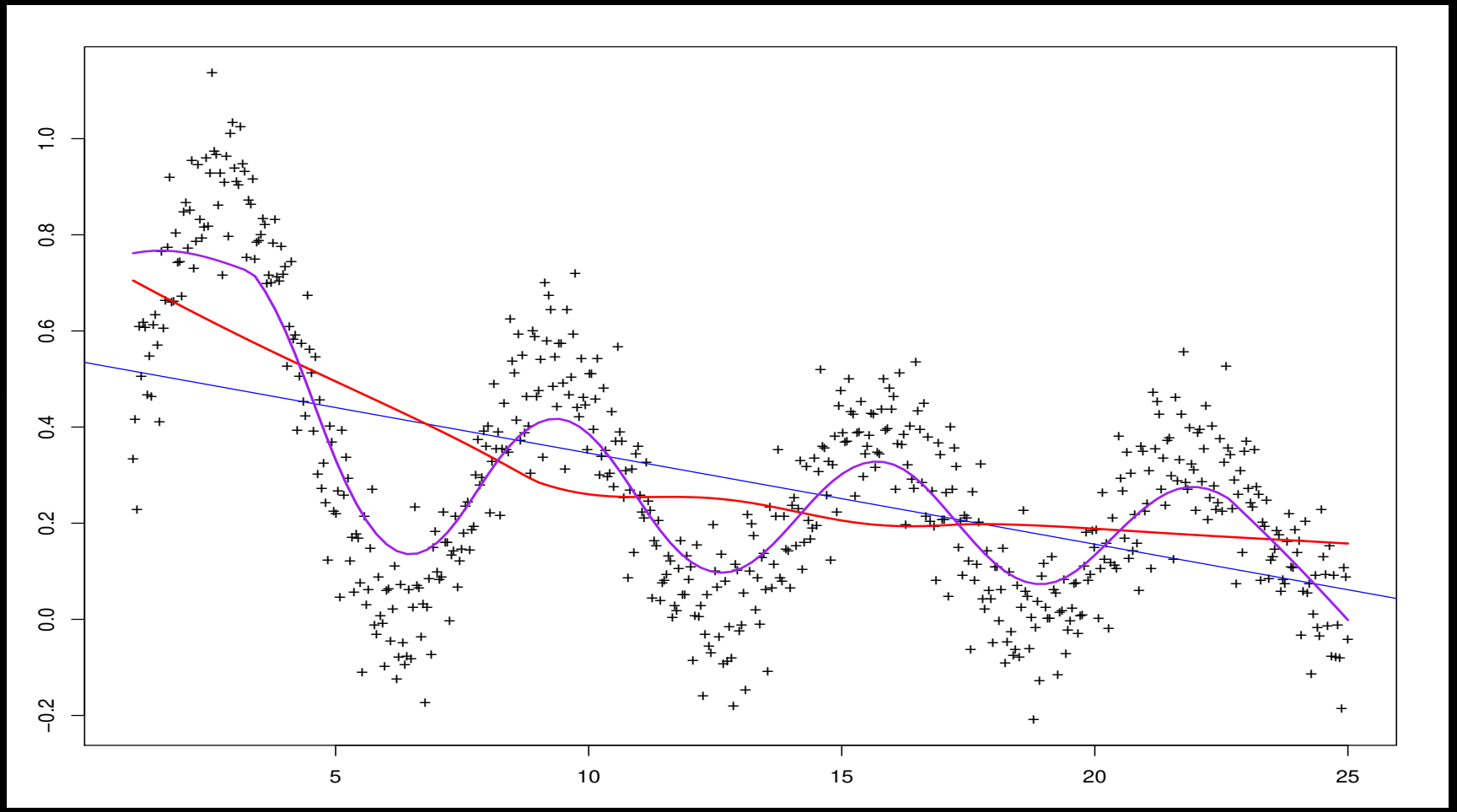
Kernel Smoothers, Rolling Our Own

```
X <- seq(0,10,length=100)
Y <- sin(X) + rnorm(length(X),0,0.3)
postscript("Class.Stat.Comp/kernel.smooth.ps")
par(mar=c(3,3,1,1),col.axis="white",col.lab="white",col.sub="white",
    col="white",bg="slategray")
plot(X,Y,pch="+")
for (j in c(1,5,10)) lines(X,k.sm(X,Y,j),col=colors()[3+24*j],lty=j,lwd=2)
dev.off()
```

Kernel Smoothers, Rolling Our Own



Recall the Lowess Smoother. . .



Lowess Smoother

- Lowess Smoother, Locally-weighted running line smoother (Cleveland 1979). Steps, for each point:

1. Denote k nearest neighbors of x_i as $N(x_i)$. Based on distance, not symmetry.
2. Determine the furthest neighbor distance:

$$\boldsymbol{\delta}(x_i) = \max_{N(x_i)} |x_i - x|, \qquad \forall x \in N(x_i).$$

3. Calculate weights for each jth point in $N(x_i)$ using the **tri-cube weighting function**:

$$u = u_{ij} = \frac{|x_i - x_j|}{\boldsymbol{\delta}(x_i)}$$

$$w(u_{ij}) = \begin{cases} (1-u^3)^3 & \text{for } 0 \leq u \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

4. Fit with a weighted running line smoother using these calculated weights:

$$\hat{f}(x_i) = \hat{\alpha}_N + \hat{\beta}_N x_i = \frac{\sum_{j=1}^{k} w(u_{ij}) Y_{ij}}{\sum_{j=1}^{k} w(u_{ij})}$$

Note: weights are all positive and decreasing with increasing distance. They also decrease with increasing window width.

Lowess Smoother

- For each data-point we are producing a neighborhood definition with weights and new Y points from the fit:

$$\forall x_i, \begin{bmatrix} x_1, & x_2, & \ldots, & x_k \\ w_1, & w_2, & \ldots, & w_k \\ \hat{y}_1, & \hat{y}_2, & \ldots, & \hat{y}_k \end{bmatrix}$$

- Actually two "flavors" of Lowess available:
  - $\lambda = 1$:

$$\min \sum_{j=1}^{k} w(u_{ij})(y_j - \alpha - \beta x_j)^2$$

  - $\lambda = 2$:

$$\min \sum_{j=1}^{k} w(u_{ij})(y_j - \alpha - \beta x_j - \gamma x_j^2)^2$$

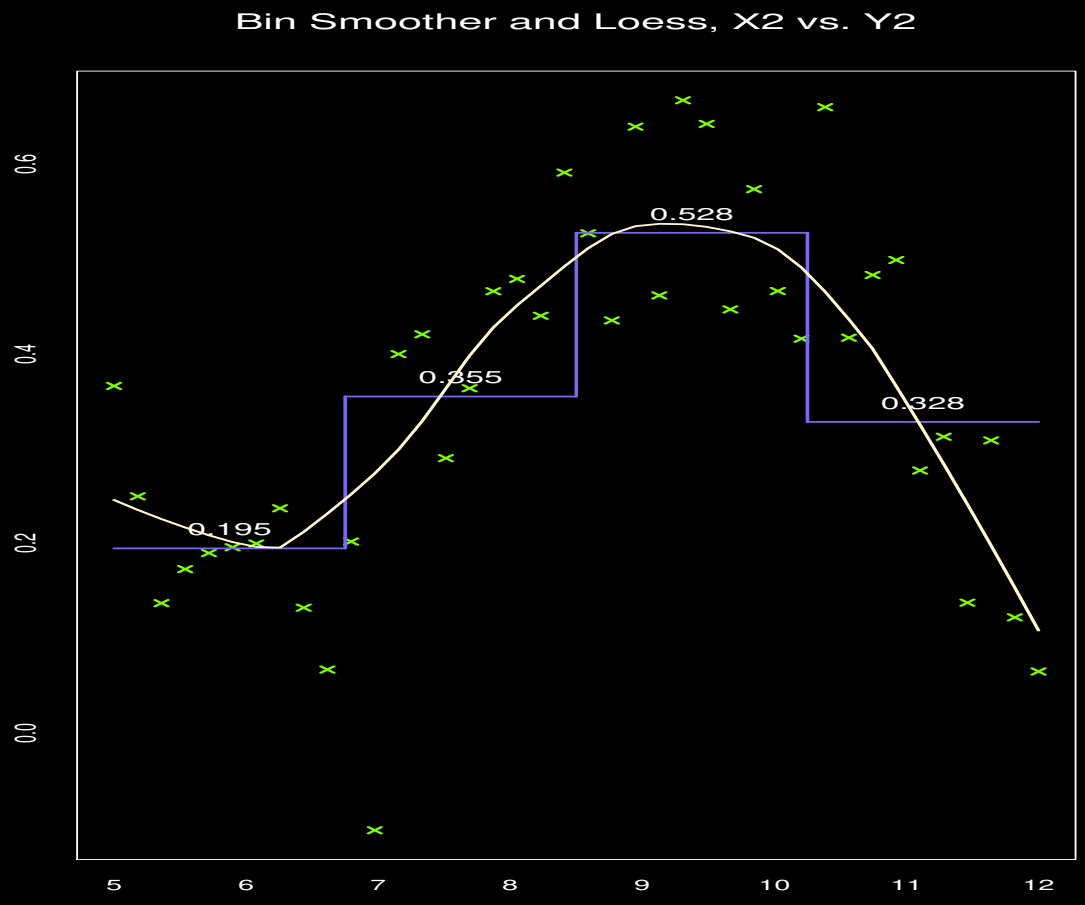
Lowess Smoother In R

```
x2 <- seq(5,12,length=40)
y2 <- (2/(pi*x2))^(0.5)*(1-cos(x2)) + rnorm(length(x2),0,1/10) + 0.1
postscript("Class.Stat.Comp/cognitive2j.ps")
par(mfrow=c(1,1),mar=c(2,2,2,2),oma=c(3,3,3,3),col.axis="white",col.lab="black",
    col.sub="white",col="white",bg="black")
plot(x2,y2,pch=4,col="chartreuse1",lwd=2)
```

Continued...

```
# Do a regressogram first
x2.cuts <- quantile(x2,seq(0,1,length=5))
y2.bins <- matrix(y2,ncol=4)
y2.means <- apply(y2.bins,2,mean)
for (i in 1:(length(x2.cuts)-1))  {
    segments(x2.cuts[i],y2.means[i],x2.cuts[i+1],y2.means[i],
        col="mediumslateblue",lwd=2)
    segments(x2.cuts[i+1],y2.means[i],x2.cuts[i+1],y2.means[i+1],
        col="mediumslateblue",lwd=2)
    text((x2.cuts[i]+x2.cuts[i+1])/2,y2.means[i]+0.02,cex=1.2,
        round(y2.means[i],3))
}
lines(lowess(x2,y2,f=0.4),col="lemonchiffon",lwd=2)
mtext(outer=TRUE,side=3,cex=1.5,line=0.25,"Bin Smoother and Loess, X2 vs. Y2")
dev.off()
```

Lowess Smoother



Regression Splines

- Choose breakpoints (also called knots).
  - This is the key tradeoff: more knots mean more flexibility, but can be more computationally intensive and sometimes too wavy.
  - Strategies:
    1. cardinal knots: uniform over range of X data,
    2. at quantiles,
    3. adaptive (complex),
    4. at selected X_i.
  - Effects:
    1. bad choices can be dramatic,
    2. bad choices can miss important features.
  - Setup: interior knots given by

$$\xi_1 < \xi_2 < \cdots < \xi_S$$

    over the range $(X_{(1)}, X_{(n)})$, along with the boundary knots:

$$\xi_0 < X_{(1)}, \qquad X_{(n)} < \xi_S$$

Regression Splines

- We need to choose a basis function which must join smoothly at knots.
- Truncated Power Series, with S knots:

$$s(x) = \delta_0 + \delta_1 x + \delta_2 x^2 + \delta_3 x^3 + \sum_{i=1}^{S} \delta_{3+i}(x - \xi_i)_+^3$$

Where $(x - \xi_i)_+^3$ means we include only positive terms, else zero:

$$(x - \xi_i)_+^3 = \max[0, (x - \xi_i)^3]$$

- So $s(x)$ is a linear weighted combination of the $S + 4$ functions:

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ P_2(x) &= x^2 \\ P_3(x) &= x^3 \\ P_{S_1}(x) = (x - \xi_1)_+^3, &\qquad P_{S_2}(x) = (x - \xi_2)_+^3, \qquad P_{S_3}(x) = (x - \xi_3)_+^3 \end{aligned}$$

meaning that the the function is linear in these $S + 4$ parameters and can be estimated with OLS.

Cubic Regression Splines

- Cubic Splines add the condition that at the boundary knots, $s(x)''$ and $s(x)'''$ are both zero, so $s(x)$ is linear (but not necessarily flat) in the intervals:

$$[\xi_0, \xi_1], \qquad [\xi_S, \xi_{S+1}].$$

- Now estimate by applying the function:

1. regress y_i on $f(x_i)$, say for $S = 3$:

$$\begin{aligned}\hat{y}_i &= f(x_i) \\ &= \delta_0 + \delta_1 x_i + \delta_2 x_i^2 + \delta_3 x_i^3 \\ &\quad + \delta_4 (x_i - \xi_1)_+^3 \\ &\quad + \delta_5 (x_i - \xi_2)_+^3 \\ &\quad + \delta_6 (x_i - \xi_3)_+^3\end{aligned}$$

   using OLS.

2. Obtains 7 coefficient estimates: $\hat{\delta}_i, \ i = 1, \ldots, 7$.

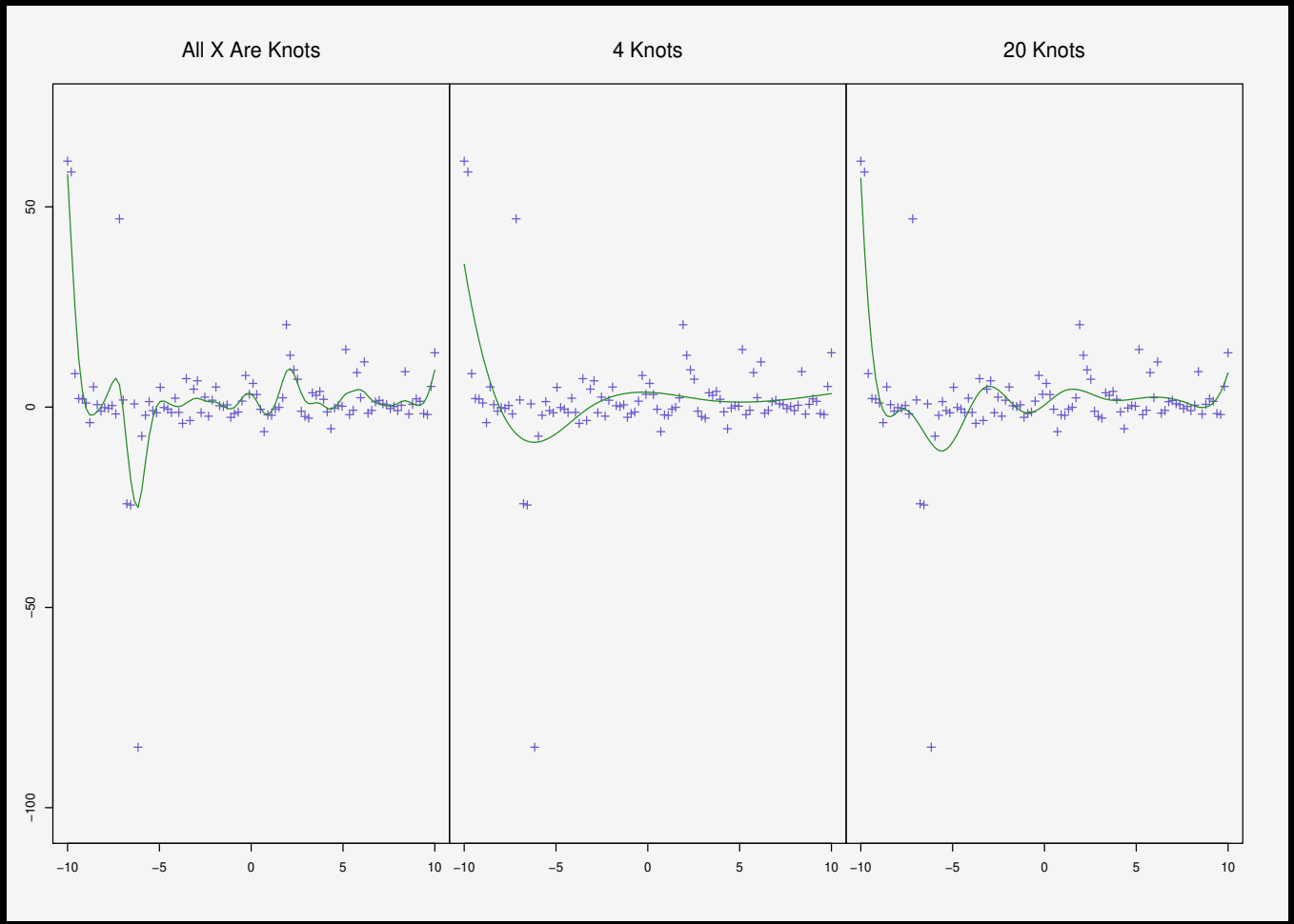
- Extension: B-Splines, different parameterizations,...

Cubic Regression Splines

- The R function for Cubic Splines is smooth.spline:

```
x1 <- seq(-10,10,length=100);   y1 <- cos(x1)/(rt(length(x1),50)*0.5)
postscript("Class.Stat.Comp/cognitive2h.ps")
par(mfrow=c(1,3),oma=c(3,3,3,3),mar=c(2,0,2,0),bg="whitesmoke")

plot(x1,y1,pch=3,ylim=range(y1)*1.2,col="slateblue")
spline.out <- smooth.spline(x1,y1,all.knots=TRUE)
lines(spline.out$x,spline.out$y,col="forest green")
mtext("All X Are Knots",outer=FALSE,side=3,cex=1.1,line=1.5)

plot(x1,y1,pch=3,ylim=range(y1)*1.2,yaxt="n",col="slateblue")
spline.out <- smooth.spline(x1,y1,all.knots=FALSE,nknots=4)
lines(spline.out$x,spline.out$y,col="forest green")
mtext("4 Knots",outer=FALSE,side=3,cex=1.1,line=1.5)

plot(x1,y1,pch=3,ylim=range(y1)*1.2,yaxt="n",col="slateblue")
spline.out <- smooth.spline(x1,y1,all.knots=FALSE,nknots=10)
lines(spline.out$x,spline.out$y,col="forest green")
mtext("10 Knots",outer=FALSE,side=3,cex=1.1,line=1.5);     dev.off()
```

Cubic Regression Splines



Penalized Splines

- The goal is:

$$\min \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_a^b (f''(t))^2 dt$$

  where λ is a fixed constant, and $a < x_{(1)}$, $x_{(n)} < b$.

- The idea is that the first term promotes fit and the second term penalizes overfitting.
- The function $f(x)$ has an explicit and unique form that minimizes the cubic spline with knots at all of the x_i.
- If $\lambda = \infty$, then we get interpolation.
- As λ approaches 0, we get closer linear regression.

Penalized Splines Code

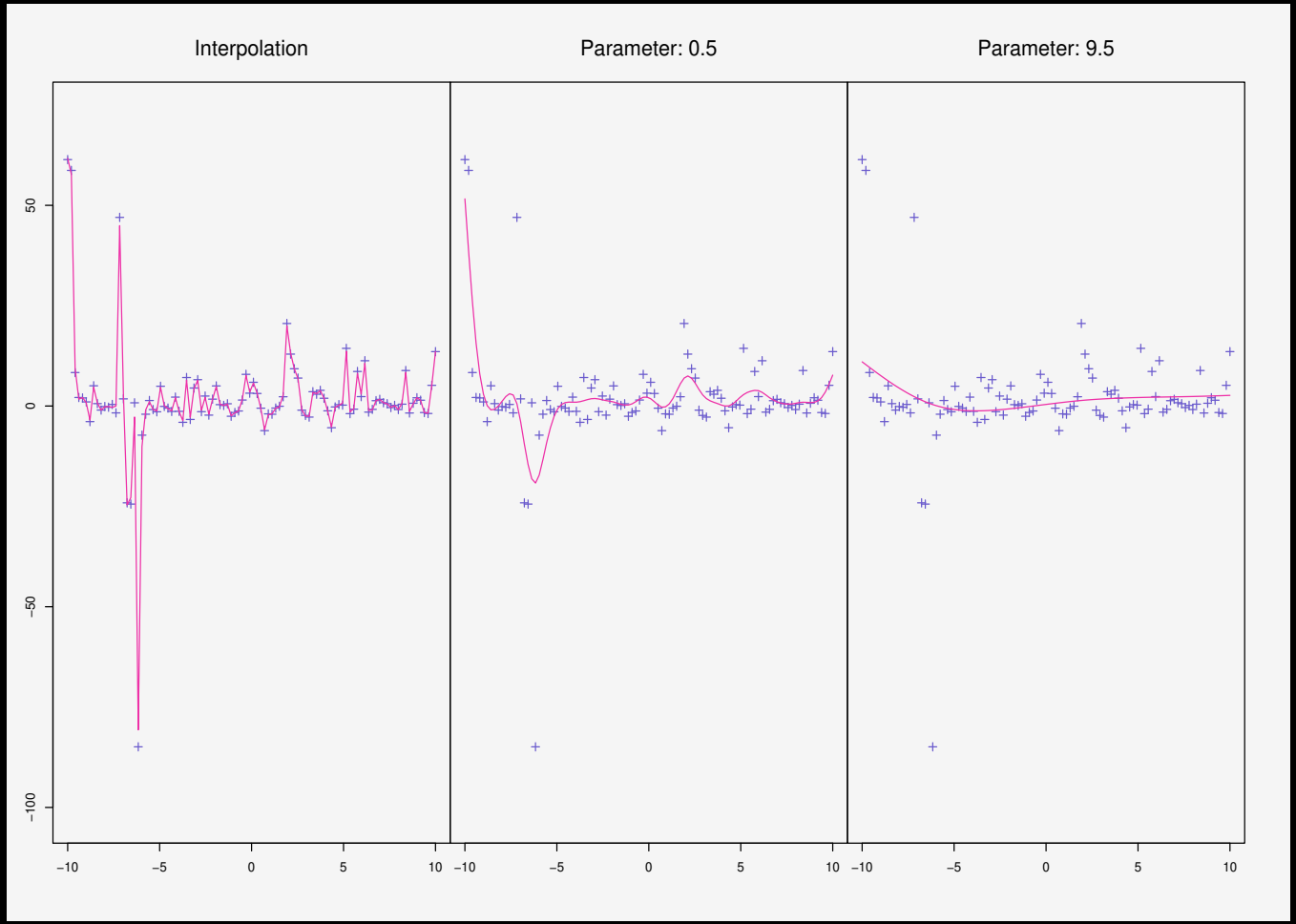
- The R function for Penalized Splines is also smooth.spline:

```
nonlin.mat <- read.table("http://jgill.wustl.edu/data/gam.test.dat",header=TRUE)
attach(nonlin.mat)
postscript("Class.Stat.Comp/cognitive2i.ps")
par(mfrow=c(1,3),oma=c(3,3,3,3),mar=c(2,0,2,0),bg="whitesmoke")

plot(x1,y1,pch=3,ylim=range(y1)*1.2,col="slateblue")
spline.out <- smooth.spline(x1,y1,spar=0.1,all.knots=TRUE)
lines(spline.out$x,spline.out$y,col="maroon2")
mtext("Interpolation",outer=FALSE,side=3,cex=1.1,line=1.5)
```

Penalized Splines

```
plot(x1,y1,pch=3,ylim=range(y1)*1.2,yaxt="n",col="slateblue")
spline.out <- smooth.spline(x1,y1,spar=0.5)
lines(spline.out$x,spline.out$y,col="maroon2")
mtext("Parameter: 0.5",outer=FALSE,side=3,cex=1.1,line=1.5)

plot(x1,y1,pch=3,ylim=range(y1)*1.2,yaxt="n",col="slateblue")
spline.out <- smooth.spline(x1,y1,spar=0.95)
lines(spline.out$x,spline.out$y,col="maroon2")
mtext("Parameter: 9.5",outer=FALSE,side=3,cex=1.1,line=1.5)
dev.off()
detach(nonlin.mat)
```

Penalized Splines



Thin Plate Splines

- Some disadvantages of standard spline approaches: user must stipulated knots, bases given for only one variable, criteria for bases unclear.
- We want: knot-free spline bases over any number of explanatory variables that have optimal properties.
- Thin plate splines (Duchon 1977; Wahba 1990, Green & Silverman 1994, Wood 2006, Chapter 4) are a good solution to this problem.
- General Strategy: penalize with derivative functions, to produce a smooth function according to

$$y_i = g(\mathbf{x}_i) + \boldsymbol{\epsilon}_i$$

  where $\boldsymbol{\epsilon}_i$ is a random error vector with good properties and $\mathbf{x}_i$ is a d-length explanatory variable vector.
- It automatically calculates how much weight to give the conflicting goals of following the data and making the fit as smooth as possible by putting the tradeoff into an explicit function.

Thin Plate Splines

- Objective: find a function $\mathbf{f}$ that minimizes the vector norm:

$$||\mathbf{y} - \mathbf{f}||^2 = \lambda J_{md}(f)$$

  where:

  - $\mathbf{y}$ is the n-length outcome variable vector,
  - $\mathbf{f} = |f(x_1), f(x_2), \ldots, f(x_n)|$,
  - λ is a smoothing parameter,
  - J_{md} is a penalty term based on the curviness of the smooth.

- This is very much in line with the spline technology that we have been studying except that there will be more "automatic" rather than human decisions.

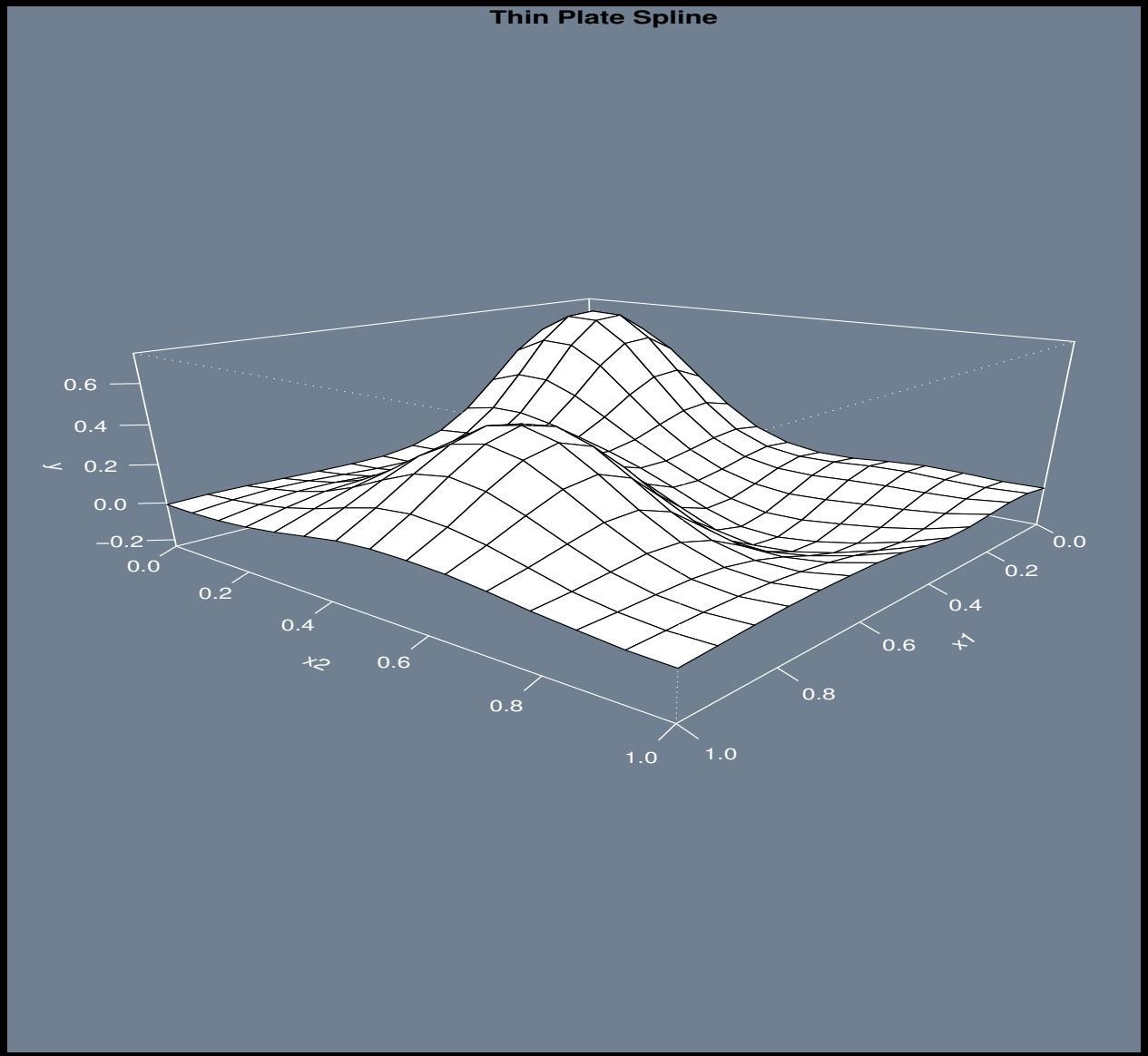
Thin Plate Splines in R

```
library(rgcvpack)

# DEFINE A THREE-DIMENSIONAL FUNCTION (2 IN, 1 OUT)
f <- function(x, y)  {
          0.75*exp( -((10*x-1)^2 + (10*y-1)^2)/5 ) +
          0.50*exp( -((10*x-7)^2 + (10*y-5)^2)/5 ) -
          0.25*exp( -((10*x-4)^2 + (10*y-7)^2)/5 )
}

# CREATE A FAKE DATASET USING THIS FUNCTION
set.seed(pi);   n <- 15;
x2 <- x1 <- seq(0,1,length=n)
y <- outer(x1, x2, f); y <- y + rnorm(n^2,0,0.05*max(abs(y)))

# THE FUNCTION NEEDS THESE AS VECTORS
x1.vec <- rep(x1, n); x2.vec <- rep(x2, rep(n,n)); y.vec <- as.vector(y)
```

Thin Plate Splines in R

```
# RUN THE THIN PLATE SPLINE WITH ALL DATA POINTS AS KNOTS, ORDER 3
thinpl.out <- fitTps(cbind(x1.vec,x2.vec), y.vec, m=3)

# GRAPH
postscript("Class.Stat.Comp/thin.plate.ps")
x2 <- x1 <- seq(0,1,length=length(y.vec))
par(mar=c(3,3,1,1),col.axis="white",col.lab="white",col.sub="white",
    col="white",bg="slategray")
persp(x1, x2, matrix(predict(thinpl.out),n,n), theta=130, phi=20,
      expand=0.50, xlab="x1", ylab="x2", zlab="y", xlim=c(0,1),
      ylim=c(0,1),zlim=range(y), ticktype="detailed", scale=FALSE,
      main="Thin Plate Spline")
dev.off()
```

Thin Plate Splines in R



Tensor Product Smoothing

- For a two-variable problem, start with the marginal smoothing functions:

$$f_x(x) = \sum_{i=1}^{I} \alpha_i a_i(x) \qquad f_y(y) = \sum_{\ell=1}^{L} \delta_\ell d_\ell(y),$$

  where α_i, δ_ℓ are parameters, and $a_i(x)$, $d_\ell(y)$ are known basis functions.

- We want $f_x()$ to vary smoothly with y, which can be done by requiring the α_i to vary smoothly with y, which means conditioning on some function of y.

- A convenient way to do this is to use:

$$\alpha_i(y) = \sum_{\ell=1}^{L} \delta_{i,\ell} d_\ell(y),$$

  which explicitly accounts for the smoothed value of y at each i in the smooth of x.

- Therefore from the x side the tensor *product* smooth is the function:

$$f_{x,y}(x,y) = \sum_{i=1}^{I} \sum_{\ell=1}^{L} \delta_{i,\ell} d_\ell(y) a_i(x).$$

Generalized Additive Models

- Big Picture: just like a GLM except we will do component-wise smoothing of some right-hand side variables.
- More computationally intensive that GLM estimation with many more model-fitting choices to make.
- Results are often given graphically for smoothed parameters, especially if there are many.
- Definitive citations:
  - Hastie and Tibshirani (1986), "Generalized Additive Models" (with discussion). *Statistical Science* 1, 297-318.
  - Wood (2006), *Generalized Additive Models: An Introduction with* `R`. Chapman & Hall/CRC.
  - Hastie (1993), in Chambers and Hastie, *Statistical Models in* `S`. Chapman & Hall.
  - Hastie and Tibshirani (1990), Generalized Additive Models. Chapman & Hall.

Generalized Additive Models

- Structure:

$$Y = \alpha + \sum_{j=1}^{n} f_j(x_j) + \epsilon$$

$$E[\epsilon] = 0$$

$$\text{cor}(\epsilon_i, x_j) = 0$$

$$\text{Var}(\epsilon) = \sigma^2$$

- Solved by an algorithm called "backfitting."
- Typically we think of f_j's as univariate and smooth, but they don't have to be either: $f(x_{j1}, x_{j2})$ like an interaction or other single dimension mapping, or categorical specifications.

Generalized Additive Models

- To avoid a plethora of free constants in each of the $f_j()$, it is common to assume $E[f_j(x_j)] = 0$, which can be achieved by centering if necessary.
- *Big point*: unlike a GLM, each term is represented additively and therefore we can use the same marginal interpretation as linear models (but without the linear assumption obviously). Two consequences:
  1. The variation of the fitted response surface holding all but one explanatory variable constant does not depend on the values of the other explanatory values.
  2. Plots of the fits separately are very useful.

Botanical Example

- Here is a *standard* example concerning cherry trees, which are less linear than one would think.
- The simple model of interest is:

$$\log(\texttt{Volume}_i) = f_1(\texttt{Height}_i) + f_2(\texttt{Girth}_i) + \epsilon_i.$$

- Start with:

```
library(mgcv)
data(trees)
t(trees)
```

```
       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18]
Girth   8.3  8.6  8.8 10.5 10.7 10.8 11.0 11.0 11.1  11.2  11.3  11.4  11.4  11.7  12.0  12.9  12.9  13.3
Height 70.0 65.0 63.0 72.0 81.0 83.0 66.0 75.0 80.0  75.0  79.0  76.0  76.0  69.0  75.0  74.0  85.0  86.0
Volume 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6  19.9  24.2  21.0  21.4  21.3  19.1  22.2  33.8  27.4

       [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26] [,27] [,28] [,29] [,30] [,31]
Girth   13.7  13.8  14.0  14.2  14.5  16.0  16.3  17.3  17.5  17.9  18.0    18  20.6
Height  71.0  64.0  78.0  80.0  74.0  72.0  77.0  81.0  82.0  80.0  80.0    80  87.0
Volume  25.7  24.9  34.5  31.7  36.3  38.3  42.6  55.4  55.7  58.3  51.5    51  77.0
```

Botanical Example

- Volume must be positive, so apply a Gamma link function:

```
tree.gam.1 <- gam(Volume ~ s(Height) + s(Girth),
                  family=Gamma(link=log), data=trees)
```

  where the log function is just for stability.

- When we type `tree.gam.1` we get:

```
Family: Gamma
Link function: log

Formula:
Volume ~ s(Height) + s(Girth)

Estimated degrees of freedom:              GCV score: 0.0080824
1.0000 2.4222  total = 4.4223
```

- The EDFs are for: `Height`, `Girth`, and the total is the sum of these plus one for the intercept.
- EDF of one means essentially a straight line and therefore not worth smoothing.

Botanical Example

- So use:

```
summary(tree.gam.1)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.276      0.015     219   <2e-16

Approximate significance of smooth terms:
           edf Ref.df     F p-value
s(Height) 1.00   1.00  31.2 6.7e-06
s(Girth)  2.42   2.42 268.9 < 2e-16

R-sq.(adj) =  0.973   Deviance explained = 97.8%
GCV score = 0.0080824  Scale est. = 0.0069294  n = 31
```

What These Quantities Mean

- The standard intercept term:

```
Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.276      0.015     219   <2e-16
```

- The smoothed terms:

```
Approximate significance of smooth terms:
           edf Ref.df     F p-value
s(Height) 1.00   1.00  31.2 6.7e-06
s(Girth)  2.42   2.42 268.9 < 2e-16
```

where:

  - `edf` gives the effective degrees of freedom (the trace of the **A** matrix). `1.00` means essentially a straight line.
  - `Ref.df`, uses an alternative estimate of edf. Useful for testing
  - `F p-value` give a Wald test of $\beta_j = 0$.

What These Quantities Mean

- `R-sq.(adj) = 0.973`, approximately the square of the correlation between observed and fitted values, adjusted for the degrees of freedom.
- `Deviance explained = 97.8`, model deviance in the GLM sense (not penalized deviance).
- `GCV score = 0.0080824`, minimized GCV score.
- `Scale est.  = 0.0069294`, estimated (or given) scale parameter σ^2.
- `n = 31`, data size without any adjustment.

Some Other Quantities of Interest

```
tree.gam.1$null.deviance
[1] 8.32

tree.gam.1$df.residual
[1] 26.6

tree.gam.1$hat
 [1] 0.2909 0.2502 0.2513 0.0744 0.1279 0.1613 0.1458 0.0615 0.0961 0.0590
[11] 0.0796 0.0593 0.0593 0.1056 0.0596 0.0727 0.1611 0.1837 0.1124 0.2674
[21] 0.0822 0.0961 0.0965 0.1443 0.1052 0.1148 0.1222 0.1301 0.1350 0.1350
[31] 0.5818
```

Formal Model Comparison

```
tree.gam.0 <- gam(Volume ~ s(Height),
                  family=Gamma(link=log), data=trees)

anova(tree.gam.0, tree.gam.1, test = "F")

Analysis of Deviance Table

Model 1: Volume ~ s(Height)
Model 2: Volume ~ s(Height) + s(Girth)
  Resid. Df Resid. Dev   Df Deviance   F Pr(>F)
1      29.0       4.77
2      26.6       0.18 2.42     4.59 273 <2e-16
```

Botanical Example

- There are also other quantities in the output:

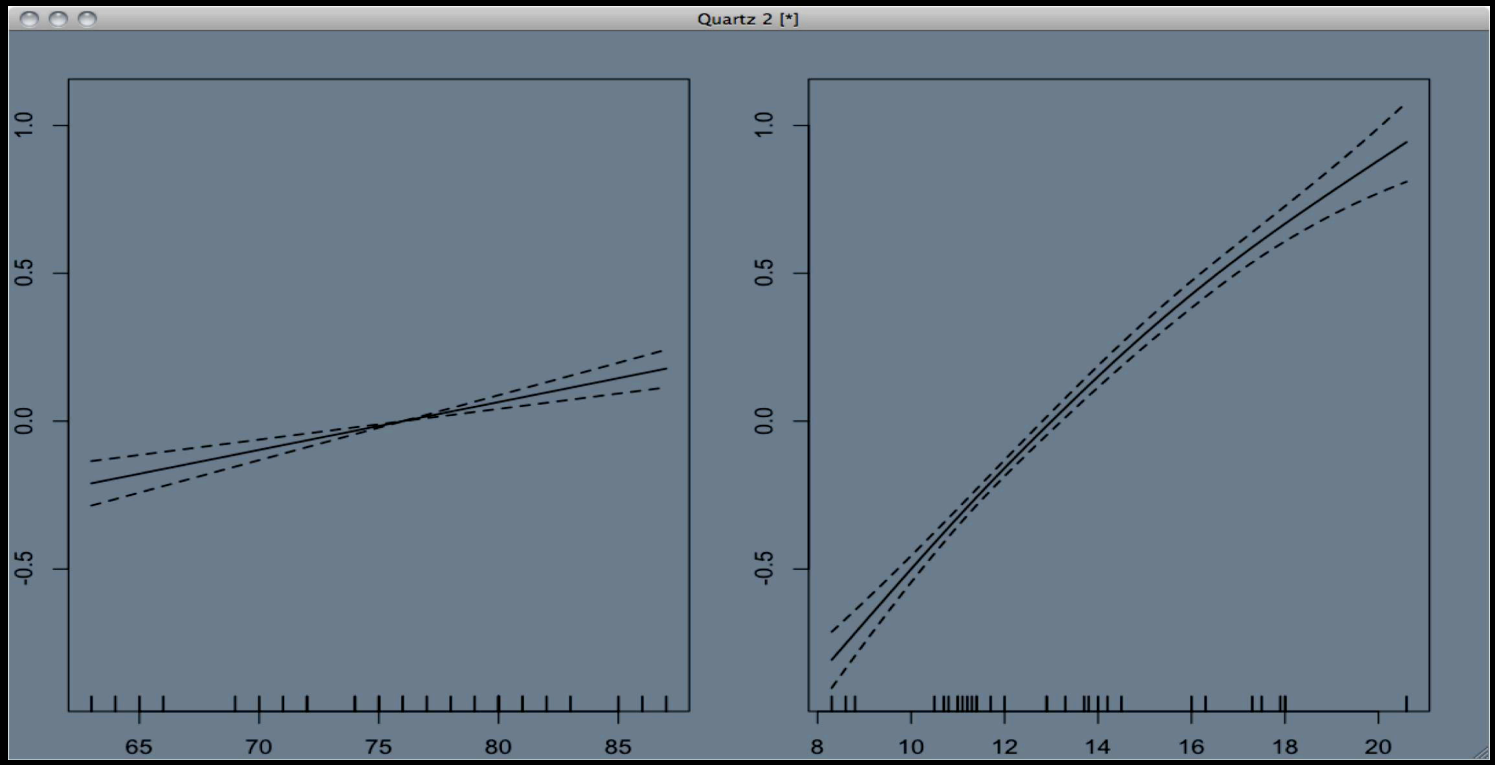
```
 names(tree.gam.1)
 [1] "coefficients"     "residuals"        "fitted.values"    "family"
 [6] "deviance"         "null.deviance"    "iter"             "weights"
[11] "df.null"          "y"                "converged"        "boundary"
[16] "reml.scale"       "aic"              "rank"             "K"
[21] "gcv.ubre"         "outer.info"       "scale"            "Vp"
[26] "Ve"               "edf"              "nsdf"             "sig2"
[31] "method"           "smooth"           "formula"          "var.summary"
[36] "model"            "control"          "terms"            "pterms"
[41] "offset"           "df.residual"      "min.edf"          "optimizer"
```

but most of these you do not need to use.

- Graphing GAM output always helps:

```
postscript("Class.Stat.Comp/tree.fig1.ps")
par(mfrow=c(1,2),mar=c(4.5,4.5,2,2),cex.axis=1,cex.lab=1.1,bg="slategray")
plot(tree.gam.1,lwd=1.5)
dev.off()
```

Botanical Example



Botanical Example

- We used the default smoother: thin plate regression splines, order of penalty equal to two and the dimension of the basis equal to 10.
- Now change this to a penalized cubic regression spline for Girth:

```
tree.gam.2 <- gam(Volume ~ s(Height) + s(Girth,bs="cr",k=20),
                  family=Gamma(link=log), data=trees)
summary(tree.gam.2)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.276      0.015     219   <2e-16

Approximate significance of smooth terms:
           edf Ref.df     F p-value
s(Height) 1.00   1.00  31.2 6.7e-06
s(Girth)  2.42   2.42 266.4 < 2e-16

R-sq.(adj) =  0.973   Deviance explained = 97.8%
GCV score = 0.008083  Scale est. = 0.0069294  n = 31
```

Botanical Example

- A parameter, γ, is used to adjust the fit by multiplying the effective degrees of freedom.
- The default value for γ is 1, and higher values give smoother fits.
- Sometimes GCV gives overly rough fits (to some tastes), so Kim & Gu suggest using 1.4:

```
tree.gam.3 <- gam(Volume ~ s(Height) + s(Girth,bs="cr",k=20),
                  family=Gamma(link=log), data=trees, gamma=1.4)
summary(tree.gam.3)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.2758     0.0151     218   <2e-16

Approximate significance of smooth terms:
           edf Ref.df     F p-value
s(Height) 1.00   1.00  31.4 6.1e-06
s(Girth)  2.17   2.17 294.2 < 2e-16

R-sq.(adj) =  0.972   Deviance explained = 97.7%
GCV score = 0.0092292  Scale est. = 0.0070251  n = 31
```

Details on GAM Model Specification

- The R formula for `gam` is just like `glm` except we have new smoother terms: `s` and `te`.
- The notation `s(X1)`, gives a spline based smooth for the `X1` explanatory variable.
- The notation `te(X2)` gives a tensor product based smooth for `X2` explanatory variable.
- It is common to "mix" smoothed and unsmoothed terms in a model:

```
Y ~ X1 + s(X2) + te(X3)
```

- There can be nested smoothing specifications:

```
Y ~ s(X1) + s(X2) + s(X1,X2)
Y ~ s(X1,X2) + s(X2,X3)
```

- We can also control the smooth with parameter vectors, for instance:

```
Y ~ te(X1,X2, bs=c("tp","tp"), m=c(3,4), k=(5,6))
```

  which gives a tensor product smooths of `X1` and `X2` with bases of dimension 3 for `X1` and 4 for `X2`, and marginal penalties of 5 for `X1` and 6 for `X2`.

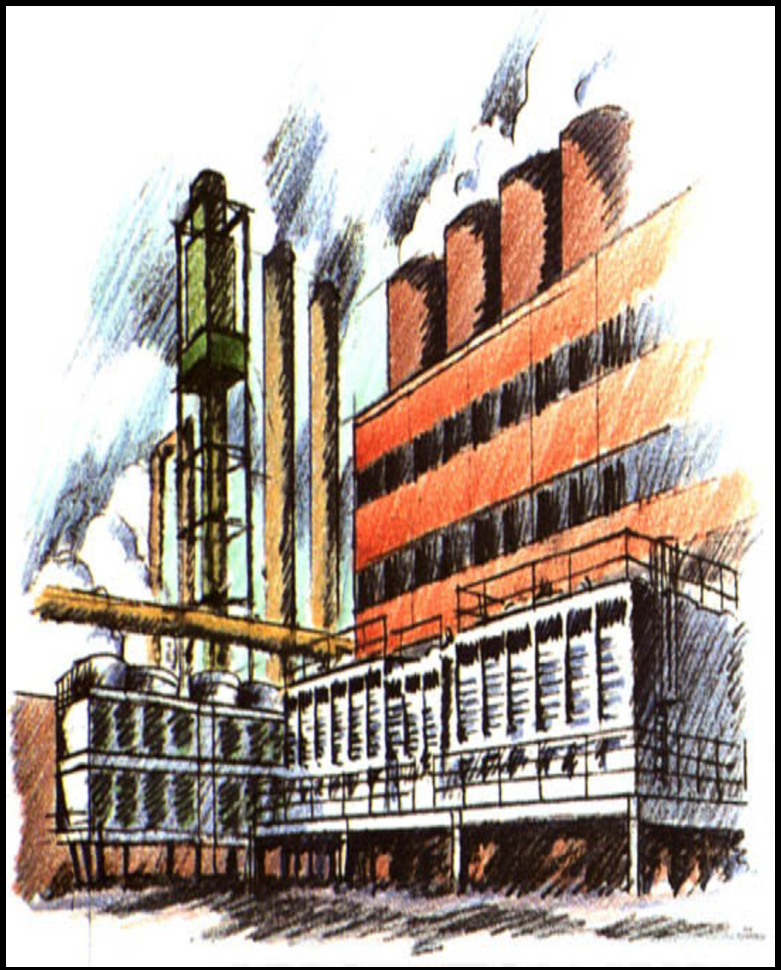
Chronic Bronchitis and Dust Concentration Study

- The file contains data from a study of the Deutsche Forschungsgemeinschaft. The data were recorded during the years 1960 and 1977 in a Munich plant (1246 workers).
- Objective: dose response model for cbr with covariates dust, expo and smoking, and assessment of threshold limiting value under which dust has no influence on cbr.
- Description of the variables:

cbr	Chronic Bronchitis Reaction
	1 : Yes
	0 : No
dust	dust concentration at working place (in mg/m)
smoking	does worker smoke?
	1 : Yes
	0 : No
expo	duration of exposure in years

Chronic Bronchitis and Dust Concentration Study

- Sources:
  Gossl, C. / Kuchenhoff, H. (2001): Bayesian analysis of logistic regression with an unknown change point and covariate measurement error. Statistics in Medicine, 20, 3109-3121.
  Kuchenhoff, H. / Carroll, R.J. (1997): Segmented regression with errors in predictors: semiparametric and parametric methods. Statistics in Medicine, 16, 169-188.



Chronic Bronchitis and Dust Concentration Study, Read Data and Run a GLM

dust.df ¡- read.table("http://jgill.wustl.edu/data/dust.asc",header=TRUE)

```
dust.df <- read.table("/Users/jgill/Class.GLM/dust.asc", header=TRUE)
summary(dust.df)
```

```
      cbr              dust            smoking           expo
 Min.   :0.0000   Min.   : 0.2000   Min.   :0.0000   Min.   : 3.00
 1st Qu.:0.0000   1st Qu.: 0.4925   1st Qu.:0.0000   1st Qu.:16.00
 Median :0.0000   Median : 1.4050   Median :1.0000   Median :25.00
 Mean   :0.2343   Mean   : 2.8154   Mean   :0.7392   Mean   :25.06
 3rd Qu.:0.0000   3rd Qu.: 5.2475   3rd Qu.:1.0000   3rd Qu.:33.00
 Max.   :1.0000   Max.   :24.0000   Max.   :1.0000   Max.   :66.00
```

```
dust.glm <- glm(cbr ~ dust+smoking+expo, family = binomial(link = logit),
        data=dust.df);
```

Chronic Bronchitis and Dust Concentration Study, GLM Results

```
summary.glm(dust.glm)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.3675  -0.7798  -0.5906  -0.3813   2.3022

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.04785    0.24813 -12.283  < 2e-16
dust         0.09189    0.02323   3.956 7.63e-05
smoking      0.67683    0.17407   3.888 0.000101
expo         0.04016    0.00620   6.476 9.40e-11
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1356.8  on 1245  degrees of freedom
Residual deviance: 1278.3  on 1242  degrees of freedom
AIC: 1286.3
```

Chronic Bronchitis and Dust Concentration Study, Run a GAM

```
library(mgcv)      # DOWNLOAD FROM CRAN IF NECESSARY, INCLUDES TENSOR SMOOTH
                   # WRITTEN BY SIMON WOOD
# library(gam)     # TREVOR HASTIE'S OLDER PACKAGE
```

- To get started we will just use the default smoother: thin place regression splines, order of penalty equal to two.

```
dust.gam <- gam(cbr ~ s(dust,k=32) + smoking + s(expo,k=32),
                family = binomial(link = logit), data=dust.df)
```

- Note the use of `s()` here to denote "spline"
- Here 32 is the dimension of the basis used to represent the smooth term in both cases.
- The GAM penalized likelihood maximization problem is solved by *Penalized Iteratively Reweighted Least Squares.*

Chronic Bronchitis and Dust Concentration Study, GAM Results

```
summary(dust.gam)
Family: binomial
Link function: logit

Formula:
cbr ~ s(dust, k = 32) + smoking + s(expo, k = 32)

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.410      1.555   -1.55  0.12114
smoking        0.673      0.179    3.77  0.00016

Approximate significance of smooth terms:
         edf Ref.df Chi.sq p-value
s(dust) 24.3   24.3   31.7    0.14
s(expo)  3.9    3.9   54.4 3.7e-11

R-sq.(adj) =  0.121   Deviance explained = 13.3%
UBRE score = -0.0074567  Scale est. = 1         n = 1246
```

Chronic Bronchitis and Dust Concentration Study, Explaining GAM OUTPUT

- `Parametric coefficients`: read like normal GLM output.
- `edf`: coefficient's estimated degrees of freedom (penalization means that many of these are less than 1)
- `Ref.df`: the same for us. Note also:

```
dust.gam$df.residual
[1] 1215.768
dust.gam$df.null
[1] 1245
sum(dust.gam$edf)
[1] 30.23168
dust.gam$min.edf
[1] 6
```

- `R-sq.(adj)`: no need to pay attention to this
- `Deviance explained`: equivalent to $(D_n - D_m)/D_n$, i.e.:

```
1-dust.gam$deviance/dust.gam$null.deviance
[1] 0.1331007
```

Chronic Bronchitis and Dust Concentration Study, Explaining GAM OUTPUT

- `UBRE score`: the UnBiased Risk Estimator estimated by $D/n + 2s(DoF)/(n-s)$, where D is the deviance, n is the number of cases, s the scale parameter and DoF is the effective degrees of freedom of the model. UBRE is the AIC only rescaled, and should be used only when s is known.
- Using `dust.gam$aic` we could get the AIC but it is misleading since we maximize the penalized likelihood rather than the regular likelihood, and these have different degrees of freedom.

Contrasting GLM and GAM Output

- Results from the GLM:

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.04785    0.24813 -12.283  < 2e-16
dust         0.09189    0.02323   3.956 7.63e-05
smoking      0.67683    0.17407   3.888 0.000101
expo         0.04016    0.00620   6.476 9.40e-11
```

- Results from the GAM:

```
Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.410      1.555   -1.55  0.12114
smoking        0.673      0.179    3.77  0.00016

Approximate significance of smooth terms:
         edf Ref.df Chi.sq p-value
s(dust) 24.3   24.3   31.7    0.14
s(expo)  3.9    3.9   54.4 3.7e-11
```

Predictions From GAM Output

- First build a design matrix for non-smokers that has values over the range of `dust` and `expo` from their maximum to their minimum.

```
attach(dust.df);     logit <- function(Xb)  1/(1+exp(-Xb))
( predict.dust.df <- data.frame(dust=seq(min(dust),max(dust),length=20),
     smoking=rep(0,length=20),expo=seq(min(expo),max(expo),length=20)) )

predict.dust.df
         dust smoking      expo
1    0.200000       0  3.000000
2    1.452632       0  6.315789
3    2.705263       0  9.631579
4    3.957895       0 12.947368
5    5.210526       0 16.263158
6    6.463158       0 19.578947
7    7.715789       0 22.894737
8    8.968421       0 26.210526
9   10.221053       0 29.526316
10 11.473684       0 32.842105
:   :              : :
```

Predictions From GAM Output

```
:  :                :  :
11 12.726316        0 36.157895
12 13.978947        0 39.473684
13 15.231579        0 42.789474
14 16.484211        0 46.105263
15 17.736842        0 49.421053
16 18.989474        0 52.736842
17 20.242105        0 56.052632
18 21.494737        0 59.368421
19 22.747368        0 62.684211
20 24.000000        0 66.000000
```

```
predict.dust.dens <- matrix(NA,20,20)
for (i in 1:20)  {
    predict.dust.df.temp <- data.frame(dust=rep(predict.dust.df$dust[i],length=20),
        smoking=rep(0,length=20),expo=seq(min(expo),max(expo),length=20))
    predict.dust.dens[i,] <-
        logit(predict.gam(dust.gam,newdata=predict.dust.df.temp,se.fit=F,plot.call=F))
}
```

Predictions From GAM Output

- Now build a design matrix for smokers that has also values over the range of `dust` and `expo` from their maximum to their minimum.

```
( predict.dust.df2 <- data.frame(dust=seq(min(dust),max(dust),length=20),
    smoking=rep(1,length=20),expo=seq(min(expo),max(expo),length=20)) )
```

```
predict.dust.df2
         dust smoking      expo
1    0.200000       1  3.000000
2    1.452632       1  6.315789
3    2.705263       1  9.631579
4    3.957895       1 12.947368
5    5.210526       1 16.263158
6    6.463158       1 19.578947
7    7.715789       1 22.894737
8    8.968421       1 26.210526
9   10.221053       1 29.526316
10 11.473684       1 32.842105
:   :            :  :
```

Predictions From GAM Output

```
:  :               : :
11 12.726316       1 36.157895
12 13.978947       1 39.473684
13 15.231579       1 42.789474
14 16.484211       1 46.105263
15 17.736842       1 49.421053
16 18.989474       1 52.736842
17 20.242105       1 56.052632
18 21.494737       1 59.368421
19 22.747368       1 62.684211
20 24.000000       1 66.000000

predict.dust.dens2 <- matrix(NA,20,20)
for (i in 1:20)  {
    predict.dust.df2.temp <- data.frame(dust=rep(predict.dust.df2$dust[i],length=20),
        smoking=rep(1,length=20),expo=seq(min(expo),max(expo),length=20))
    predict.dust.dens2[i,] <-
        logit(predict.gam(dust.gam,newdata=predict.dust.df2.temp,se.fit=F,plot.call=F))
}
```

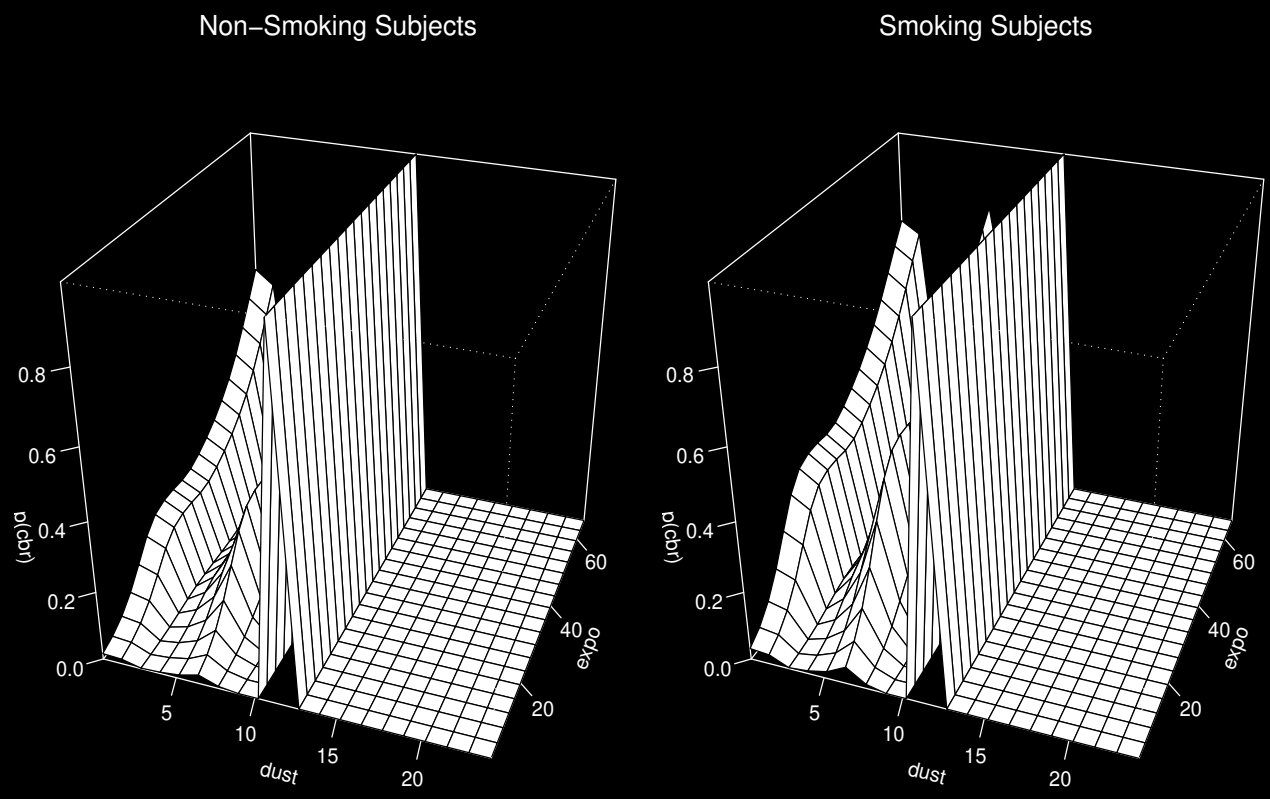
Predictions From GAM Output, Perspective Plot

- Graph with a perspective plot...
- The surface is then viewed by looking at the origin from a direction defined by `theta` and `phi`.
- If `theta` and `phi` are both zero the viewing direction is directly down the negative y axis.
- Changing `theta` will vary the *azimuth* and changing `phi` the *colatitude*.
- The term `r` is the distance of the eyepoint from the center of the plotting box.

Predictions From GAM Output, Perspective Plot

```
postscript("Class.Stat.Comp/dust.gam1.ps")
par(mfrow=c(1,2),mar=c(1,1,1,1),oma=c(1,1,1,1),col.axis="white",col.lab="black",
    col.sub="white",col="white",bg="black")

persp(predict.dust.df$dust,predict.dust.df$expo,predict.dust.dens,
    theta=20,phi=30,r=5,ticktype="detailed",xlab="dust",ylab="expo",zlab="p(cbr)")

mtext(side=3,outer=F,cex=1.3,"Non-Smoking Subjects",line=-1)

persp(predict.dust.df2$dust,predict.dust.df2$expo,predict.dust.dens2,
    theta=20,phi=30,r=5,ticktype="detailed",xlab="dust",ylab="expo",zlab="p(cbr)")

mtext(side=3,outer=F,cex=1.3,"Smoking Subjects",line=-1)

dev.off()
```

Predictions From GAM Output, Perspective Plot



Terrorism Data Analysis

- This example is about comparing different GAM fits.
- Source: The International Policy Institute for Counter-Terrorism, Herzlia, Israel.
- Provided on an online database with details of attacks in Israel since September, 2000.
- Subsetted by Markison to give 103 suicide attacks over a three-year period from November 6, 2000 to November 3, 2003 when there was a steep drop.
- Information provided: date and place of the attack, attack type, the type of target and device employed, organizational affiliation of the attacker, and the number of casualties, along with a written description of the attack.
- Casualties are given personal attributes such as name, age, sex, nationality, and religion.

ison3.txt",header=TRUE)

Terrorism Data

```
harr <- read.table("http://jgill.wustl.edu/data/harrison4.txt",header=TRUE)
apply(harr[,-1],2,table)

$NumberKilled
 0  1  2  3  5  6  7  8  9 11 15 17 19 21 23 24 30
44 13  9  8  3  2  3  2  2  3  4  3  1  3  1  1  1

$NumberInjured
  0   1   2   3   4   5   6   8   9  11  13  14  16  17  20  21  22  26  27  30
 28   1   5   4   4   3   1   2   2   2   1   1   1   1   3   1   1   1   1   5
 40  42  47  50  52  57  58  59  60  65  69  86  90 100 102 120 130 150 188
  3   1   1   7   1   1   1   2   5   1   1   1   1   3   1   1   1   2   1

$TotalCasualties
  0   1   2   3   4   5   6   8   9  10  12  13  15  17  20  21  26  27  29  30
 22   5   6   4   3   3   2   2   1   1   2   2   1   1   2   1   1   2   1   1
 31  32  35  38  45  49  50  51  52  53  57  58  59  61  62  63  65  67  71  75
  1   1   1   1   1   2   1   1   1   1   2   1   1   2   1   1   2   2   3   2
 81  91  93 105 106 123 126 141 145 151 180 199
  1   1   1   1   1   1   1   1   1   1   1   1
```

Terrorism Data

```
$ResponsibleHamas
 0  1
59 44

$ResponsibleisPIJ
 0  1
79 24

$TargetisMilitary
 0  1
76 10

$TargetisBus
 0  1
89 14

$TargetisCheckpoint
 0  1
87 16

$ResponsibleisMartyrs
 0  1
78 25

$ResponsibleisOther
 0  1
99  4

$TargetisCivilian
 0  1
10 76

$TargetisCafe
 0  1
89 14

$TargetisResidence
  0   1
102   1
```

Terrorism Data

```
$TargetisOffshore
  0   1
101   2

$TargetisStore
  0  1
 96  7

$TargetisStreet
 0  1
71 32

$TargetisTravelstop
  0  1
 88 15

$DeviceisCar
 0  1
89 14

$DeviceisBoat
  0   1
101   2

$AttackisPrevented
  0   1
101   2

$AttackerisChallenged
 0  1
63 40

$FirstAttackerisMale
 0  1
 7 92

$FirstAttackerisFemale
 0  1
92  7
```

Terrorism Data

```
$AgeofFirstAttacker
16 17 18 19 20 21 22 23 24 25 26 27 29 31 43 45 48
 1  8  7 10 15 11 10 12  2  3  2  1  3  1  1  1  1
```

- Data Notes:
  - measurement is very nongranular,
  - some dichotomous variables very skewed,
  - and real motivations, planning, and training are not observed.

Terrorism Data Analysis

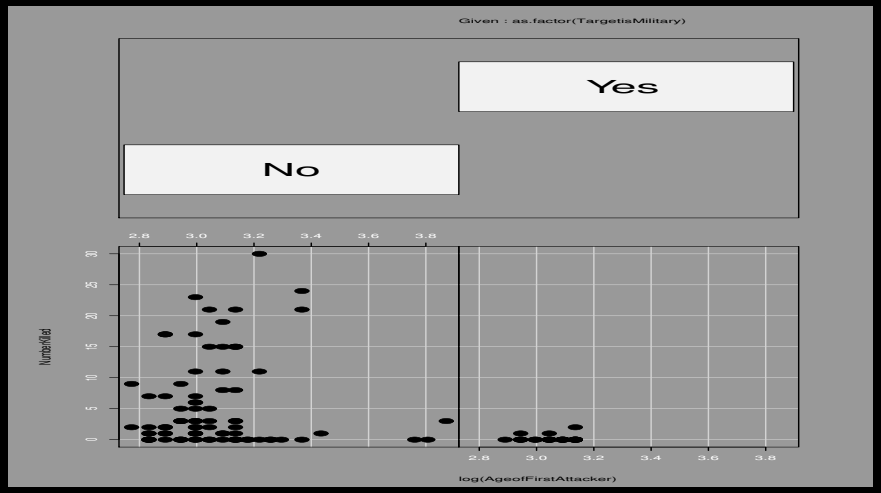
```
postscript("Class.Stat.Comp/coplot1.ps")
par(mfrow=c(1,1),mar=c(6,6,6,2),col.axis="white",col.lab="white", col.sub="white",col="black",bg="grey60", cex.lab=.001)
coplot2(NumberKilled ~ log(AgeofFirstAttacker) | as.factor(AttackerisChallenged), data = harr, cex=2, pch=19)
mtext(side=1,line=10,"log(Age of First Attacker)",cex=2)
mtext(side=2,line=10,"Number Killed",cex=2)
mtext(side=3,line=10,"Attacker Is Challenged",cex=2)
dev.off()

postscript("Class.Stat.Comp/coplot2.ps")
par(mfrow=c(1,1),mar=c(6,6,6,2),col.axis="white",col.lab="white", col.sub="white",col="black",bg="grey60", cex.lab=.001)
coplot2(NumberKilled ~ log(AgeofFirstAttacker) | as.factor(DeviceisCar), data = harr,cex=2,pch=19)
mtext(side=1,line=10,"log(Age of First Attacker)",cex=2)
mtext(side=2,line=10,"Number Killed",cex=2)
mtext(side=3,line=10,"Attacker Is Challenged",cex=2)
dev.off()

postscript("Class.Stat.Comp/coplot3.ps")
par(mfrow=c(1,1),mar=c(6,6,6,2),col.axis="white",col.lab="white", col.sub="white",col="black",bg="grey60", cex.lab=.001)
coplot2(NumberKilled ~ log(AgeofFirstAttacker) | as.factor(TargetisMilitary), data = harr,cex=2,pch=19)
mtext(side=1,line=10,"log(Age of First Attacker)",cex=2)
mtext(side=2,line=10,"Number Killed",cex=2)
mtext(side=3,line=10,"Attacker Is Challenged",cex=2)
dev.off()

postscript("Class.Stat.Comp/coplot4.ps")
par(mfrow=c(1,1),mar=c(6,6,6,2),col.axis="white",col.lab="white", col.sub="white",col="black",bg="grey60", cex.lab=.001)
coplot2(NumberKilled ~ log(AgeofFirstAttacker) | as.factor(ResponsibleHamas), data = harr,cex=2,pch=19)
mtext(side=1,line=10,"log(Age of First Attacker)",cex=2)
mtext(side=2,line=10,"Number Killed",cex=2)
mtext(side=3,line=10,"Attacker Is Challenged",cex=2)
dev.off()
```

Terrorism Data Analysis

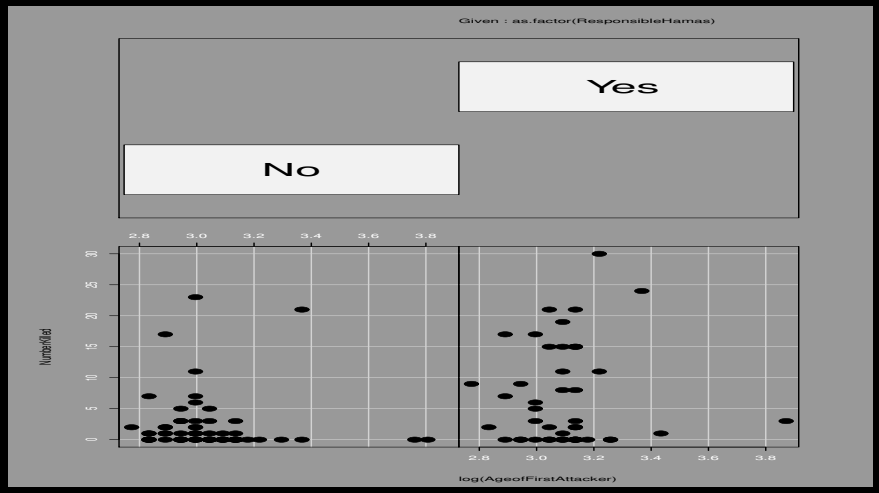
Attacker is Challenged

Device is Car

Target is Military



Hamas Responsible



Terrorism Data Analysis

```
harr.gam1 <- gam(NumberKilled ~ s(log(AgeofFirstAttacker),bs="tp") + log(Date) +
                   AttackerisChallenged + FirstAttackerisFemale +
                   DeviceisCar + TargetisCafe + TargetisMilitary +
                   ResponsibleHamas, data=harr)

harr.gam2 <- gam(NumberKilled ~ s(log(AgeofFirstAttacker),bs="tp") +
                   s(log(Date),bs="cr",k=5) +
                   AttackerisChallenged + FirstAttackerisFemale +
                   DeviceisCar + TargetisCafe + TargetisMilitary +
                   ResponsibleHamas, data=harr)
```

Terrorism Data Analysis

```
summary(harr.gam1)

Parametric coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)            -5.1905     2.4608  -2.109 0.037602
log(Date)               2.5481     0.6210   4.103 8.72e-05
AttackerisChallenged   -3.7087     1.1381  -3.259 0.001563
FirstAttackerisFemale   2.0103     2.1901   0.918 0.361043
DeviceisCar             0.8684     1.6705   0.520 0.604415
TargetisCafe            4.0685     1.6358   2.487 0.014656
TargetisMilitary       -4.5712     1.4032  -3.258 0.001568
ResponsibleHamas        4.0489     1.1692   3.463 0.000809

Approximate significance of smooth terms:
                            edf Ref.df     F p-value
s(log(AgeofFirstAttacker)) 1.893  1.893 1.159   0.316

R-sq.(adj) =  0.391   Deviance explained = 44.4%
GCV score = 30.657  Scale est. = 27.712    n = 103
```

Terrorism Data Analysis

```
summary(harr.gam2)

Parametric coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)              3.879      1.065   3.643 0.000446
AttackerisChallenged    -3.673      1.135  -3.237 0.001684
FirstAttackerisFemale    2.096      2.185   0.959 0.339952
DeviceisCar              1.185      1.699   0.698 0.487152
TargetisCafe             4.100      1.627   2.520 0.013481
TargetisMilitary        -4.599      1.402  -3.280 0.001467
ResponsibleHamas         4.544      1.225   3.709 0.000356

Approximate significance of smooth terms:
                            edf Ref.df     F  p-value
s(log(AgeofFirstAttacker)) 1.811  1.811 1.465 0.236896
s(log(Date))               2.503  2.503 6.978 0.000633

R-sq.(adj) =  0.396   Deviance explained = 45.7%
GCV score = 30.911  Scale est. = 27.515    n = 103
```

Terrorism Data Analysis

- It is also possible to do simultaneous multivariate smoothing:

```
harr.gam3 <- gam(NumberKilled ~ te(log(AgeofFirstAttacker),log(Date),k=3) +
                 AttackerisChallenged + FirstAttackerisFemale +
                 DeviceisCar + TargetisCafe + TargetisMilitary +
                 ResponsibleHamas, data=harr)
```

- This fits a bivariate surface for `log(AgeofFirstAttacker)` and `log(Date)` at the same time using a *tensor product smooth*.
- In this case it is a slightly better fit...

Terrorism Data Analysis

```
summary(harr.gam3)

Parametric coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)              4.377      1.070   4.091  9.3e-05
AttackerisChallenged    -4.059      1.144  -3.549 0.000616
FirstAttackerisFemale    1.286      2.255   0.571 0.569737
DeviceisCar              1.204      1.677   0.718 0.474763
TargetisCafe             3.824      1.638   2.335 0.021752
TargetisMilitary        -4.772      1.384  -3.448 0.000860
ResponsibleHamas         4.027      1.184   3.400 0.001003

Approximate significance of smooth terms:
                                       edf Ref.df     F p-value
te(log(AgeofFirstAttacker),log(Date)) 5.613  5.613 3.794 0.00255

R-sq.(adj) =  0.412   Deviance explained = 47.9%
GCV score = 30.527  Scale est. = 26.789    n = 103
```

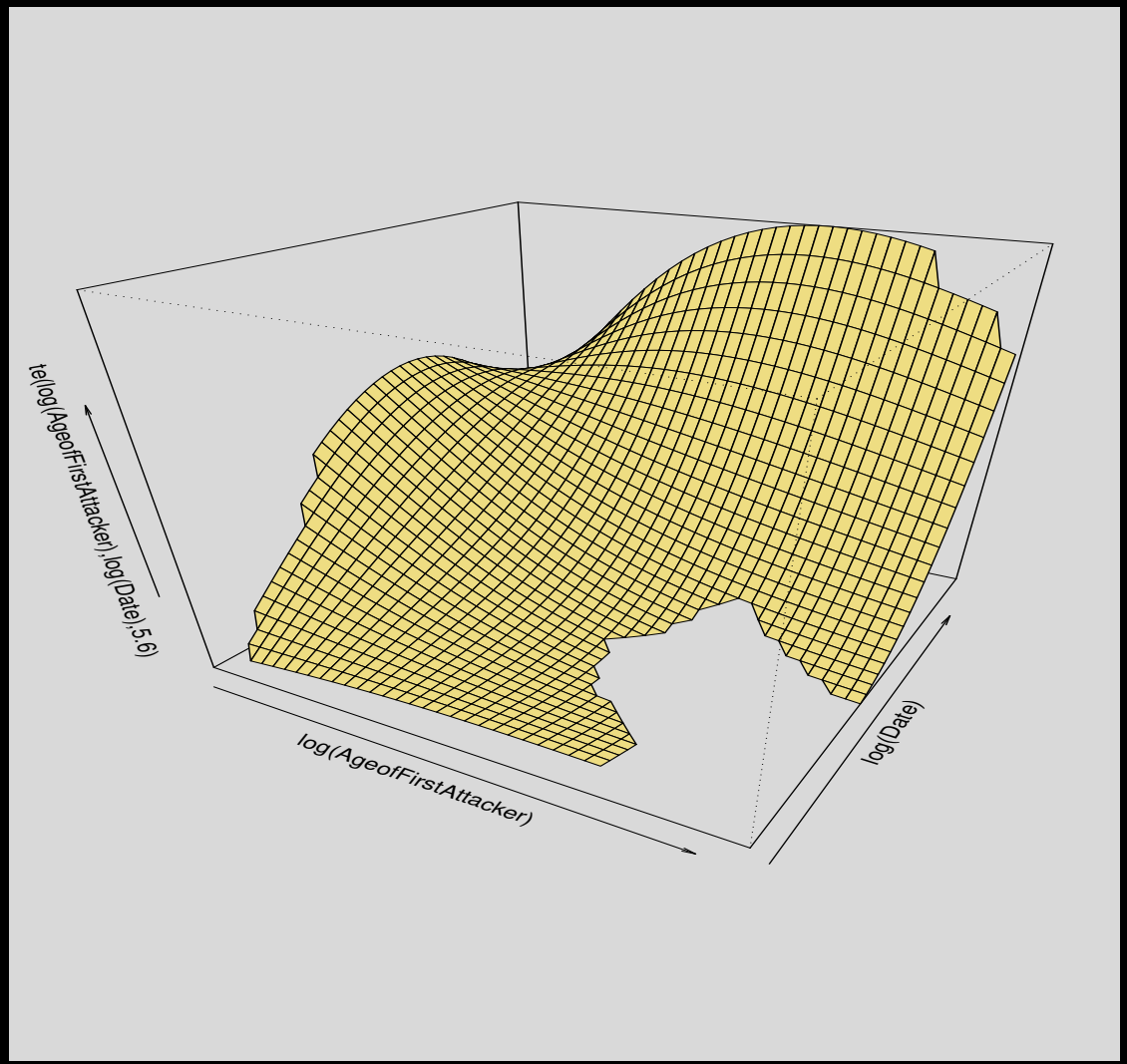
Graphing the Terrorism Data Analysis

- There is a handy but slightly confusing plot routine for GAMs:

```
postscript("Class.Stat.Comp/harr.persp1.ps")
par(mfrow=c(1,1),mar=c(1,1,0,1),oma=c(0,0,0,0),col.axis="white",col.lab="white",
     col.sub="white",col="black",bg="grey85",cex.lab=3)
plot.gam(harr.gam3,too.far=0.25,lwd=1,pers=TRUE,col="lightgoldenrod",cex.lab=1.3)
dev.off()
postscript("Class.Stat.Comp/harr.persp2.ps")
par(mfrow=c(1,1),mar=c(1,2,0,1),oma=c(0,0,0,0),col.axis="white",col.lab="white",
     col.sub="white",col="black",bg="grey85",cex.lab=3)
plot.gam(harr.gam3,too.far=0.25,lwd=1,pers=TRUE,col="lightgoldenrod",cex.lab=1.3,
         theta=285,phi=5)
dev.off()
```

- This option gives the perspective plot.

Viewing the Nonparametric Results



Viewing the Nonparametric Results